Package 'RAS'

November 2, 2026

Title Regional Association Score for Genome-Wide Association Studies

Version 1.1.3

Description Implements the Regional Association Score (RAS) method for genome-wide association studies (GWAS). For each single nucleotide polymorphism (SNP), RAS quantifies the strength of association within its surrounding genomic region, arranges these regional scores along the chromosome into a signal profile, and locates association regions on that profile with one of two detectors: the original changepoint detector, or a box-scan region detector that also delimits broad plateau-shaped regions. Genotypes can be streamed from a chunked on-disk format through compiled code so that peak memory no longer grows with chromosome size, and the regional weights can be taken from an independent external GWAS (harmonised summary statistics) instead of a within-sample split. The method is described in Jiang and Zhang (2025) <doi:10.1073/pnas.2419721122>.

License MIT + file LICENSE

URL https://github.com/hepingzhangyale/RAS

BugReports https://github.com/hepingzhangyale/RAS/issues

Encoding UTF-8

Language en-GB

Imports grDevices, graphics, parallel, segmented, stats, tools, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Jiahe Jin [aut],
Yiran Jiang [aut],
Heping Zhang [aut, cre]

Maintainer Heping Zhang <heping.zhang@yale.edu>

Repository CRAN

Date/Publication 2026-11-02 21:00:53 UTC

Contents

RAS-package . . . 3
bed_to_rasbin . . . 5
compute_gwas_weights . . . 6
compute_gwas_weights_original . . . 7
compute_pgs_matrix . . . 9
geno_to_rasbin . . . 11
get_break_points . . . 12
get_local_maximum . . . 13
plot.ras . . . 14
print.ras . . . 16
ras . . . 16
ras-aliases . . . 21
rasbin_header . . . 22
rasbin_read_chunk . . . 22
ras_box_calibrate . . . 23
ras_box_detect . . . 24
ras_box_stat . . . 26
ras_detect . . . 28
ras_detect_original . . . 29
ras_harmonize_sumstats . . . 32
ras_liftover_sumstats . . . 34
ras_map_ids_to_rsid . . . 35
ras_memory . . . 36
ras_original . . . 37
ras_read_variant_dictionary . . . 42
ras_scan . . . 43
ras_scan_external . . . 45
ras_scan_external_original . . . 46
ras_scan_original . . . 47
ras_sumstats_report . . . 50
ras_validate . . . 51
ras_weights_from_sumstats . . . 53
release_memory . . . 56
screen_forward_max_region . . . 56
screen_forward_max_region_original . . . 58
slope_test . . . 61

Index **63**

RAS-package *RAS: Regional Association Score for Genome-Wide Association Studies*

Description

The RAS package implements the Regional Association Score method for genome-wide association studies. It converts per-SNP effect sizes into a genomic $-\log_{10}(p)$ profile and locates association regions on that profile with one of two detectors. The method supports both continuous and binary traits.

Option A, one call (recommended)

`ras` runs the complete pipeline and returns a `"ras"` object. Its first argument is either an in-memory genotype matrix or the path to a chunked on-disk `.rasbin` file (`geno_to_rasbin`, `bed_to_rasbin`); genotypes are streamed from the file in compiled code, so peak memory is set by `chunk_snps` rather than by the chromosome. `ras_original` is the pure-R in-memory implementation of 1.0.x, kept for reference. Use `plot.ras` to visualise results.

```
result <- ras(geno, phenotype, covariates,
              covariate_cols = c("age", "sex", paste0("pc", 1:10)),
              is_continuous  = TRUE,
              chrom = 1, save_dir = "results/")

bed_to_rasbin("cohort_chr1.bed", "cohort_chr1.rasbin")
result <- ras("cohort_chr1.rasbin", phenotype, covariates,
              covariate_cols = c("age", "sex", paste0("pc", 1:10)),
              is_continuous  = TRUE,
              chrom = 1, save_dir = "results/")

print(result)             # detected positions / regions
plot(result)              # full-chromosome scan profile
plot(result, zoom = TRUE) # zoomed view around each detection
```

Option B, step by step (advanced)

`ras_scan` → `ras_detect` → `ras_validate` → `plot.ras`, or `ras_scan` → `ras_box_detect` → `plot.ras`.

```
## Step 1: averaged -log10(p) profile
scan     <- ras_scan(geno, phenotype, covariates,
                     covariate_cols = c("age", "sex", paste0("pc", 1:10)),
                     is_continuous = TRUE, chrom = 1, save_dir = "results/")

## Step 2 + 3, changepoint detector
detected <- ras_detect(scan$x, scan$y,
                       window_size = 3000,
```

```
                        slope.p.values.threshold.left  = 1e-10,
                        slope.p.values.threshold.right = 1e-20)
final    <- ras_validate(detected, x = scan$x, y = scan$y,
                         this.skip = 10, p.value.threshold = 1e-10)

## Step 2 + 3, box-scan detector (threshold from ras_box_calibrate())
final    <- ras_box_detect(scan$x, scan$y, calibration = cal)

## Step 4: plot
result   <- structure(
  list(scan = scan, detection = final, chrom = 1, save_dir = "results/"),
  class = "ras")
plot(result)
plot(result, zoom = TRUE)
```

Stage 1, scan

ras_scan performs **num_rep** independent 50/50 train/hold-out splits. In each repetition it fits per-SNP regressions on the training half (compute_gwas_weights) and runs a forward scan on the hold-out half (screen_forward_max_region) that slides an expanding window and records the minimum p-value at each grid position, streaming chunk_snps SNP columns at a time from the .rasbin file. The $-\log_{10}(p)$ vectors are averaged across repetitions. For continuous traits, covariates are residualised from the training phenotype each repetition (no hold-out leakage); for binary traits, a Rao score test is used in the scan step. The pure-R in-memory implementation (ras_scan_original, compute_gwas_weights_original, compute_pgs_matrix, screen_forward_max_region_origin computes the same profile and additionally offers the per-window logistic regression (scan_test = "glm") for binary traits.

When an independent GWAS of the same trait is available, its effect sizes can replace the training split: ras_harmonize_sumstats aligns the external summary statistics to the target variant map and ras_scan_external scans all samples once with those fixed weights.

Stage 2 and 3, changepoint detector (detector = "changepoint")

ras_detect (pure-R: ras_detect_original) slides a window of width **cp_window_size** across the profile. At each position get_break_points fits a segmented linear model and applies the Davies test. A candidate is retained only when three conditions all hold: Davies $p <$ **cp_p_threshold**, left-side slope significantly positive (slope_test, $p <$ **cp_slope_left**), and right-side slope significantly negative ($p <$ **cp_slope_right**). Each accepted candidate is snapped to the nearest local peak via get_local_maximum. ras_validate then re-tests each candidate with local Davies tests on both sides within a window of half-width **second_window_size**; a candidate is accepted if either side passes $p <$ **second_p_threshold**, and candidates with $-\log_{10}(p) \leq$ **min_signal** at the estimated position are discarded regardless.

Stage 2 and 3, box-scan detector (detector = "box")

ras_box_detect scores every window of a set of widths by how far its mean stands above both adjacent flanks and above the profile background, in the raw $-\log_{10}(p)$ units of the profile (ras_box_stat), and reports the intervals $[\tau_L, \tau_R]$ whose score clears a threshold. A window inside a plateau or on the

shoulder of a peak scores about zero, so broad regions are delimited at their edges and neighbouring peaks stay separate. The threshold is calibrated on null profiles with ras_box_calibrate.

Stage 4, plots

plot.ras with zoom = FALSE produces two files: the RAS scan profile with detected changepoints or intervals highlighted, and a dual-axis overlay with the Davies test significance (or the box-scan score) at every candidate. plot.ras with zoom = TRUE produces a multi-panel figure with one zoomed panel per detection.

Author(s)

Jiahe Jin <jiahe.jin@yale.edu>, Yiran Jiang <yiran.jiang@uky.edu>, Heping Zhang <heping.zhang@yale.edu> (maintainer)

See Also

- ras, ras_scan, ras_detect, ras_validate, ras_scan_external
- ras_original, ras_scan_original, ras_detect_original (pure-R, in-memory)
- ras_box_detect, ras_box_calibrate, ras_box_stat
- geno_to_rasbin, bed_to_rasbin, rasbin_header, rasbin_read_chunk
- ras_harmonize_sumstats, ras_weights_from_sumstats, ras_sumstats_report
- compute_gwas_weights_original, compute_pgs_matrix, screen_forward_max_region_original
- get_break_points, get_local_maximum, slope_test
- plot.ras, print.ras
- ras_memory, release_memory

---

bed_to_rasbin *Convert a PLINK 1 .bed/.bim/.fam Fileset to .rasbin*

Description

Converts a PLINK 1 binary fileset directly into the chunked .rasbin format, without ever materializing a dense genotype matrix in R. Both the read from .bed and the write to .rasbin are done in fixed-size SNP chunks in C (see src/plink_io.c), so peak memory is $O(\text{n_samples} \times \text{chunk_snps})$ regardless of how many samples or SNPs the fileset contains.

Usage

```
bed_to_rasbin(bed_path, out_path, chunk_snps = 2000)
```

Arguments

bed_path	Character. Path to the .bed file. The companion .bim and .fam files are expected alongside it with the same stem (standard PLINK convention, e.g. --bfile prefix).
out_path	Character. Destination path for the .rasbin file. A companion <out_path>.meta.rds is written alongside it holding sample and SNP identifiers, parsed from .fam/.bim.
chunk_snps	Integer. Number of SNPs read from .bed and written to .rasbin per chunk. Only affects peak memory and throughput, not the resulting file. Default 2000, matching geno_to_rasbin's chunk_write_size default.

Details

Only SNP-major .bed files are supported (the only mode any current PLINK version writes). Genotype dosage counts copies of the A1 allele (the fourth column of .bim), matching PLINK's own --recode A / plink2's --export A convention: 2 = homozygous A1, 1 = heterozygous, 0 = homozygous A2, NA = missing.

Unlike geno_to_rasbin (dtype 0, double, 8 bytes/genotype), this writes dtype 1 (int8, 1 byte/genotype) – 8x smaller on disk, though still ~4x larger than .bed's 2-bit packing (see src/rasbin.h for why byte-per-genotype was chosen over bit-packing). Both dtypes are read transparently by rasbin_read_chunk and every _fast pipeline function.

Value

Invisibly, the out_path that was written.

---

compute_gwas_weights *Per-SNP GWAS Effect Size Weights*

Description

Fits the per-SNP regressions of RAS Stage 1 on the training half of the sample: for each variant, phenotype1 ~ dosage for a continuous trait or phenotype1 ~ dosage + covariates for a binary trait. SNP columns are streamed from the .rasbin file in chunks and only the training rows are kept, so the full genotype matrix is never held in memory. The pure-R, in-memory implementation is compute_gwas_weights_original.

Usage

```
compute_gwas_weights(
  rasbin_path,
  phenotype1,
  this.sample,
  this.df,
  is_continuous,
  covariate_formula = NULL,
  chunk_snps = 5000
)
```

Arguments

`rasbin_path`	Character. Path to the `.rasbin` genotype file.
`phenotype1`	Numeric vector, one value per training sample: the covariate-adjusted residuals for a continuous trait, the raw 0/1 outcome for a binary trait.
`this.sample`	Integer vector. 1-based row indices of the training samples in the genotype file.
`this.df`	Data frame of covariates for all samples (indexed by `this.sample`); used for binary traits only.
`is_continuous`	Logical. `TRUE` for a quantitative trait.
`covariate_formula`	Character. Right-hand side of the binary-trait model, including the SNP term `this.x`, for example `"this.x + age + sex"`. Ignored for continuous traits.
`chunk_snps`	Integer. SNP columns read per disk chunk. Default `5000`.

Details

Binary traits are fitted by the Frisch-Waugh-Lovell identity: the covariate projection is computed once and each SNP costs one residual regression. SNPs with a missing dosage in the training rows are refitted on their complete rows by downdating the shared covariate cross-products in blocks, so post-QC sequencing data with sparse missingness stays fast.

Value

Numeric matrix with N rows (variants) and columns `Estimate`, `Std. Error`, `t value` and `Pr(>|t|)`. The `Estimate` column is the weight vector used by `screen_forward_max_region`.

See Also

`ras_scan`, which calls this function once per repetition; `compute_gwas_weights_original`.

`compute_gwas_weights_original` *Per-SNP GWAS Effect Size Weights (Pure-R, In-Memory)*

Description

This is the pure-R, in-memory implementation that RAS 1.0.x shipped under the plain name. Since 1.1.0 the plain name (`compute_gwas_weights`) is the compiled, disk-backed implementation; this function is kept for reference and gives the same results. Runs a per-SNP regression on the training split to obtain effect-size estimates used as polygenic score (PGS) weights in the forward scan.

Usage

```
compute_gwas_weights_original(
  geno,
  phenotype1,
  this.sample,
  this.df,
  is_continuous,
  covariate_formula = NULL
)
```

Arguments

geno	Matrix of genotype dosages (n samples $\times$ N variants). Only the rows indexed by this.sample are used.
phenotype1	Numeric vector of length |this.sample|. Training-split phenotype values. For continuous traits these should be covariate-residualised values; for binary traits, the raw 0/1 indicator.
this.sample	Integer vector. Row indices of the training split in geno.
this.df	Data frame of covariates with rows aligned to geno. Required for binary traits; used to subset and build the regression data frame. Ignored for continuous traits.
is_continuous	Logical. TRUE for quantitative traits, FALSE for binary (case/control) traits.
covariate_formula	Character. The right-hand side of the regression formula including the SNP term this.x. For example, "this.x + age + sex + pc1". If NULL, a default formula with sex, age, age-squared, age-sex interaction, and the first ten ancestry PCs is used.

Details

For **continuous traits** (is_continuous = TRUE) the function fits lm(phenotype1 ~ this.x) for each SNP, excluding samples with missing dosage via na.omit. The covariates are assumed to have already been residualised from phenotype1 before this function is called (as done in ras_scan_original).

For **binary traits** (is_continuous = FALSE) the function builds a data frame combining the training-split rows of this.df with the dosage column (this.x) and phenotype (phenotype1), removes incomplete cases with na.omit, and fits lm(phenotype1 ~ covariate_formula). Note that lm rather than glm(family = binomial) is used at the GWAS weight stage to obtain a linear probability approximation to the effect size; this is consistent with standard GWAS practice for weight derivation.

Both branches are vectorised: instead of fitting one lm per SNP, all NA-free SNPs are solved in closed form with a single BLAS cross-product (continuous) or a batched Frisch-Waugh-Lovell residualisation (binary). SNP columns that contain missing dosages are handled individually so the result matches lm + na.omit exactly. A single "Starting GWAS ..." message is printed.

Value

A numeric matrix with N rows and 4 columns:

`Estimate` Per-SNP effect-size estimate (slope of the dosage term). The first column is used as PGS weights by `compute_pgs_matrix`.

`Std. Error` Standard error of the estimate.

`t value` t-statistic.

`Pr(>|t|)` Two-sided p-value.

See Also

`compute_pgs_matrix` for the next step, which multiplies these weights by the hold-out genotype matrix. `ras_scan_original` for the recommended high-level entry point that calls this function automatically.

Examples

```
set.seed(1)
n_samp <- 60; n_snp <- 10
geno <- matrix(sample(0:2, n_samp * n_snp, replace = TRUE,
                      prob = c(0.6, 0.3, 0.1)),
               nrow = n_samp, ncol = n_snp)
pheno_train <- rnorm(n_samp)

coef_mat <- compute_gwas_weights_original(
  geno              = geno,
  phenotype1        = pheno_train,
  this.sample       = seq_len(n_samp),
  this.df           = data.frame(row.names = seq_len(n_samp)),
  is_continuous     = TRUE
)
dim(coef_mat)        # n_snp x 4
head(coef_mat, 3)
```

---

compute_pgs_matrix *Build the Per-Individual PGS Contribution Matrix*

Description

Multiplies each hold-out individual's genotype dosages by the per-SNP PGS weights to produce a matrix of weighted dosage contributions used in the forward scan.

Usage

```
compute_pgs_matrix(geno, this.leftout, pgs.weights)
```

Arguments

geno	Matrix of genotype dosages (n samples $\times$ N variants). NA values are replaced with 0 in-place before multiplication.
this.leftout	Integer vector. Row indices of the hold-out split in geno.
pgs.weights	Numeric vector of length N. Per-SNP effect-size weights, typically the Estimate column from compute_gwas_weights_original.

Details

Missing genotype values (NA) in geno are imputed to zero before computing the product. This is a simple mean-imputation equivalent under a centred dosage scale and is sufficient for the forward scan, where the primary goal is to aggregate regional signals rather than obtain individual-level accuracy.

The caller's geno object is **not** modified: the NA-replacement assignment triggers R's copy-on-modify semantics, so a local copy of geno is made inside the function. Callers working with very large genotype matrices should be aware that this temporarily doubles the memory footprint of geno during the call; use ras_memory to check feasibility before running.

Value

A numeric matrix of dimensions $|\texttt{this.leftout}| \times N$, where entry $[i, j]$ is the weighted dosage contribution of SNP j for hold-out individual i (i.e., geno[this.leftout[i], j] * pgs.weights[j]).

See Also

compute_gwas_weights_original for the step that produces pgs.weights. screen_forward_max_region_original for the step that consumes this matrix. ras_memory for pre-flight memory estimation.

Examples

```
set.seed(2)
geno    <- matrix(sample(0:2, 50 * 20, replace = TRUE), nrow = 50, ncol = 20)
weights <- rnorm(20)
leftout <- 1:15

pgs_mat <- compute_pgs_matrix(geno, leftout, weights)
dim(pgs_mat)   # 15 x 20

## Entry [i, j] equals geno[leftout[i], j] * weights[j]
stopifnot(pgs_mat[1, 1] == geno[leftout[1], 1] * weights[1])
```

geno_to_rasbin *Write a Genotype Matrix to the Chunked .rasbin Format*

Description

Converts an in-memory genotype dosage matrix (or a matrix stored in an `.rds`/`.csv`/`.tsv` file) into `.rasbin`: a flat, column-major binary file that supports reading an arbitrary SNP-column range with a single seek, without loading the whole matrix into memory.

Usage

```
geno_to_rasbin(geno, out_path, chunk_write_size = 2000)
```

Arguments

`geno`	Either a numeric matrix (n samples $\times$ N variants), or a character path to a `.rds`, `.csv`, or `.tsv`/`.txt` file containing one (first column treated as sample ID if present as rownames-style data). The source is read into memory exactly once, regardless of downstream chunked access.
`out_path`	Character. Destination path for the `.rasbin` file. A companion `<out_path>.meta.rds` is written alongside it holding sample and SNP identifiers.
`chunk_write_size`	Integer. Number of SNP columns written per `writeBin` call. Only affects write throughput, not the resulting file. Default `2000`.

Details

File layout (see `src/rasbin.h` for the authoritative spec):

- Header (32 bytes): 8-byte magic `"RASBIN01"`, 8-byte `n_samples` (int64), 8-byte `n_snps` (int64), 4-byte `dtype` (int32; `0` = double), 4 reserved bytes.
- Body: `n_samples * n_snps` doubles, column-major, so column `j` (0-based) occupies bytes `[32 + j*n_samples*8, 32 + (j+1)*n_samples*8)`.

`NA` dosages round-trip exactly: they are ordinary IEEE-754 doubles at the bit level, so a raw byte copy preserves R's `NA_real_` sentinel without any special-casing.

Value

Invisibly, the `out_path` that was written.

`get_break_points` *Detect a Single Changepoint via Segmented Regression*

Description

Fits a segmented linear model to detect one breakpoint in the relationship between `x` and `y` up to index `t`, and tests its significance using the Davies test.

Usage

```
get_break_points(x, y, t)
```

Arguments

`x`	Numeric vector. Predictor (e.g., SNP position indices).
`y`	Numeric vector. Response (e.g., $-\log_{10}(p)$-values from a scan). Must be the same length as `x`.
`t`	Integer. Number of observations to use from the start of `x` and `y`. Must satisfy `t >= 4` for segmented regression to be identifiable.

Details

The function first fits an ordinary linear model `y ~ x` over the first `t` observations, then calls `segmented` to estimate a single breakpoint. If the segmented fit succeeds, the Davies test (`davies.test`) is applied to the base linear model to assess whether the slope change is significant.

Both the `segmented` call and the Davies test are wrapped in `try(..., silent = TRUE)`: if either fails (e.g., due to collinear data or a degenerate window), the function returns `p.values = 1` and `NULL` for the breakpoint and slopes rather than propagating an error.

The breakpoint position is returned as an index into the full `x` vector (not just the first `t` elements), found by locating the element of `x[1:t]` closest to the estimated breakpoint coordinate.

Value

A named list with four elements:

`break.points` Integer. Index of the estimated breakpoint in `x[1:t]`, or `NULL` if no breakpoint was found.

`p.values` Numeric. Davies test p-value for the breakpoint, or `1` if the fit failed or no breakpoint was found.

`slope.left` Numeric. Estimated slope to the left of the breakpoint, or `NULL` if not found.

`slope.right` Numeric. Estimated slope to the right of the breakpoint, or `NULL` if not found.

See Also

`ras_detect_original` which calls this function repeatedly in a sliding-window loop. `slope_test` for the one-tailed slope verification step. `segmented`, `davies.test` for the underlying segmented-regression routines.

Examples

```
set.seed(1)
x <- 1:60
y <- c(seq(0, 6, length.out = 30),
       seq(6, 2, length.out = 30)) + rnorm(60, sd = 0.4)
result <- get_break_points(x, y, t = 60)
cat("Breakpoint index:", result$break.points, "\n")
cat("Davies p-value:  ", result$p.values,    "\n")
cat("Left slope:      ", result$slope.left,   "\n")
cat("Right slope:     ", result$slope.right,  "\n")
```

`get_local_maximum` *Find the Local Maximum Within a Window*

Description

Returns the index of the maximum value of `y` within a symmetric window of half-width `window.size` centred at `x0`.

Usage

```
get_local_maximum(y, x0, window.size = 50)
```

Arguments

`y`	Numeric vector. Values to search (e.g., $-\log_{10}(p)$ values from the RAS scan).
`x0`	Integer. Centre index of the search window.
`window.size`	Integer. Half-width of the search window. The search covers indices `max(1, x0 - window.size)` to `min(length(y), x0 + window.size)`. Default `50`.

Details

This function is used by `ras_detect` (and `ras_detect_original`) after the sliding-window loop to snap each accepted changepoint index to the nearest local peak in the $-\log_{10}(p)$ profile. Snapping to the peak ensures that reported positions correspond to the most significant SNP in the association region rather than to the mathematical breakpoint of the piecewise linear fit, which may be slightly offset.

Value

Integer. The index (into `y`) of the local maximum within the window centred at `x0`. If multiple positions tie for the maximum, `which.max` returns the first.

See Also

`ras_detect` which calls this function to refine detected changepoint positions.

Examples

```
y <- c(1, 3, 7, 5, 2, 8, 4, 1)

## Peak within a window of half-width 2 centred at index 3
get_local_maximum(y, x0 = 3, window.size = 2)  # returns 3 (value = 7)

## Peak within a window of half-width 3 centred at index 3
## covers indices 1:6; max is at index 6 (value = 8)
get_local_maximum(y, x0 = 3, window.size = 3)  # returns 6
```

`plot.ras` *Plot a RAS Result Object*

Description

S3 plot method for objects of class `"ras"` returned by `ras` or `ras_original`. When `zoom = FALSE` (default) produces the full-chromosome scan profile with detected changepoints marked. When `zoom = TRUE` produces the zoomed multi-panel figure around each detected changepoint. For a result obtained with `detector = "box"` the shaded areas are the detected intervals $[\tau_L, \tau_R]$, the vertical lines mark each interval's anchor, and the right-hand axis of the overlay plot shows the box-scan score instead of the Davies test significance.

Usage

```
## S3 method for class 'ras'
plot(
  x,
  zoom = FALSE,
  device = "pdf",
  p.threshold = 8,
  y_cap = NULL,
  min_display_p = 1,
  xlim = NULL,
  zoom_half_width = 3000,
  ncol = 3,
  min_signal = 2.5,
  ...
)
```

Arguments

`x`	Object of class `"ras"` as returned by `ras`.
`zoom`	Logical. `FALSE` (default) plots the full scan profile; `TRUE` plots zoomed panels around each detected changepoint.
`device`	Character. Output device: `"pdf"` (default), `"png"`, or `"screen"`.
`p.threshold`	Numeric. Significance reference line on the $-\log_{10}$ scale. Default `8`.

y_cap	Numeric or NULL. Cap the y-axis (scan plot only). Default NULL.
min_display_p	Numeric. Minimum all.p.values for a candidate to appear in the dual-axis overlay (scan plot only). Default 1.
xlim	Numeric vector of length 2, or NULL. Restrict plot to this genomic range (scan plot only). Default NULL.
zoom_half_width	Numeric. Half-width of each zoom panel in SNP index units (zoom plot only). Default 3000.
ncol	Integer. Columns in the zoom panel grid (zoom plot only). Default 3.
min_signal	Numeric. Low-signal colour boundary on the $-\log_{10}$ scale: scan values below this threshold are drawn in blue; values at or above it are drawn in yellow or higher. Should match the value passed to ras_validate. Default 2.5.
...	Currently unused.

Value

Invisibly returns x. Called for its side effect of writing plot files or rendering to the active graphics device.

See Also

ras for the function that produces the "ras" object. print.ras for the console summary.

Examples

```
## Build a minimal "ras" object by hand (see ras() for the full pipeline)
set.seed(7)
xg <- seq(1, 2000, by = 10)
yg <- c(seq(0, 9, length.out = length(xg) %/% 2),
        seq(9, 1, length.out = length(xg) - length(xg) %/% 2)) +
      rnorm(length(xg), sd = 0.4)
detection <- list(
  tau_hats         = xg[100],
  all.changepoints = xg[c(95, 100, 105)],
  all.p.values     = c(5, 12, 4),
  left.slopes      = 0.3,
  right.slopes     = -0.3
)
result <- structure(
  list(scan = list(x = xg, y = yg), detection = detection,
       chrom = 1, save_dir = tempdir()),
  class = "ras")

plot(result, device = "screen")
plot(result, zoom = TRUE, device = "screen")
```

print.ras *Print a RAS Result Object*

Description

Prints a one-line summary of a "ras" object showing the chromosome and detected changepoint positions.

Usage

```
## S3 method for class 'ras'
print(x, ...)
```

Arguments

x	Object of class "ras" as returned by ras.
...	Currently unused.

Value

Invisibly returns x.

See Also

ras, plot.ras.

ras *RAS: Regional Association Score Analysis*

Description

One-call entry point of the package. Runs the RAS pipeline on one chromosome: the Stage-1 scan (ras_scan), region detection on the resulting profile with the box-scan detector (ras_box_detect, default) or the changepoint detector (ras_detect followed by ras_validate), and the diagnostic plots (plot.ras). Genotypes are streamed from a .rasbin file; an in-memory genotype matrix is accepted as well.

Usage

```
ras(
  geno,
  phenotype,
  covariates,
  covariate_cols,
  is_continuous,
```

```
  num_rep = 5,
  skip1 = 10,
  skip2 = 20,
  chrom = 1,
  save_dir = file.path(tempdir(), "RAS"),
  min_window_size = 5,
  max_window_size = 100,
  scan_test = c("score", "glm"),
  chunk_snps = 5000,
  cores = 1,
  keep_reps = FALSE,
  cp_p_threshold = 0.01,
  cp_window_size = 3000,
  cp_min_length = 10,
  cp_slope_check_window = 30,
  cp_slope_left = 1e-10,
  cp_slope_right = 1e-20,
  second_window_size = 50,
  second_p_threshold = 1e-10,
  min_signal = 2.5,
  detector = c("box", "changepoint"),
  box_threshold = NULL,
  box_calibration = NULL,
  box_level0 = NA_real_,
  box_null = 100,
  box_alpha = 0.05,
  box_null_n = NULL,
  box_method = c("order", "gumbel"),
  run_plots = TRUE,
  plot_device = "pdf",
  plot_p_threshold = 8,
  plot_y_cap = NULL
)
```

Arguments

`geno`	Either the path to a `.rasbin` genotype file (written by `geno_to_rasbin` or `bed_to_rasbin`), or an in-memory numeric genotype matrix (n samples by N variants). A matrix is converted to a temporary `.rasbin` file for the run; convert once with `geno_to_rasbin` when running repeatedly.
`phenotype`	Numeric vector of length n, in the sample order of `geno`.
`covariates`	Data frame with n rows, in the sample order of `geno`.
`covariate_cols`	Character vector. Names of the columns of `covariates` to adjust for.
`is_continuous`	Logical. `TRUE` for a quantitative trait, `FALSE` for a binary (0/1) trait.
`num_rep`	Integer. Number of train/hold-out repetitions to average over. Default `5`.
`skip1`	Integer. Stride of the profile grid in SNPs: the profile has one value every `skip1` SNPs. Default `10`.

skip2 Integer. Step, in SNPs, by which the scan window grows from min_window_size to max_window_size at each grid position. Default 20.

chrom Integer or character. Chromosome label used in the output file names. Default 1.

save_dir Character or NULL. Directory that receives the per-repetition coefficient matrices and the averaged profile; NULL writes nothing. Default file.path(tempdir(), "RAS").

min_window_size, max_window_size
Integer. Smallest and largest scan window, in SNPs. Default 5 and 100.

scan_test Character. Per-window test for a binary trait: "score" (default, Rao score test in closed form) or "glm" (per-window logistic regression, the default of RAS 1.0.x). "glm" runs the in-memory implementation ras_scan_original and therefore needs a genotype matrix, not a .rasbin path. Ignored for continuous traits.

chunk_snps Integer. SNP columns held in memory at a time. Bounds peak memory at roughly n * chunk_snps * 8 bytes per worker. Default 5000.

cores Integer. Number of worker processes the repetitions are spread over (a PSOCK cluster). Default 1. Each worker draws its own random-number stream, so results with cores > 1 are not bit-identical to a serial run with the same seed, and peak memory grows roughly in proportion to cores.

keep_reps Logical. Also return (as $reps) and save the individual per-repetition profiles, which are needed to study the split-to-split variability that the average removes. Default FALSE.

cp_p_threshold Numeric. Davies test p-value threshold for a first-pass candidate changepoint. Default 0.01.

cp_window_size Integer. Width, in grid points, of the sliding window of the first pass. Default 3000.

cp_min_length Integer. Minimum number of grid points on each side of a candidate. Default 10.

cp_slope_check_window
Integer. Half-width, in grid points, of the local window in which the slopes on each side of a candidate are tested. Default 30.

cp_slope_left, cp_slope_right
Numeric. One-tailed p-value thresholds for the rising left slope and the falling right slope. Default 1e-10 and 1e-20.

second_window_size
Integer. Half-width, in grid points, of the second-pass local Davies tests. Default 50.

second_p_threshold
Numeric. Davies test p-value threshold of the second pass. Default 1e-10.

min_signal Numeric. Minimum profile value $-\log_{10}(p)$ at a changepoint for it to be kept; also the colour boundary in the plots. Default 2.5.

detector Character. "box" (default): the box-scan detector ras_box_detect, which reports intervals $[\tau_L, \tau_R]$ and also delimits broad plateau-shaped regions. "changepoint": the two-pass changepoint detector of RAS 1.0.x, ras_detect then ras_validate,

which reports single positions. The box_* arguments belong to "box", the cp_*, second_* and min_signal arguments to "changepoint".

box_threshold Numeric or NULL. Fixed score threshold for detector = "box". NULL (default) uses box_calibration, or calibrates on box_null permutations.

box_calibration
A ras_box_calibrate result or NULL (default). Supplies the threshold and level0 for detector = "box" and skips the permutation calibration.

box_level0 Numeric. Null-profile median for the box scan's level factor (see ras_box_stat). NA (default) takes it from box_calibration when given and otherwise disables the factor.

box_null Integer. Number of permuted-phenotype scans used to calibrate the box-scan threshold when neither box_threshold nor box_calibration is given. Default 100. Continuous phenotypes are permuted as residuals after the covariates (Freedman-Lane); binary phenotypes are permuted directly. 0 disables the calibration, in which case a threshold must be supplied.

box_alpha Numeric. Family-wise error rate of the calibrated threshold. Default 0.05.

box_null_n Integer or NULL. Number of samples, drawn at random once, on which the permuted-phenotype scans are run. NULL (default) uses all samples. The null distribution of the profile maximum depends on the linkage structure, not on the sample size, so a subset of a few tens of thousands of individuals calibrates a biobank-scale cohort at a fraction of the cost (on a 352-pig chromosome, subsets of a third and a sixth of the animals gave thresholds within 3 percent of the full-sample one).

box_method Character. How the calibrated threshold is read off the null maxima: "order" (default, exact order statistic, use with box_null of 100 or more) or "gumbel" (Gumbel fit, usable from box_null = 20 with a slightly liberal threshold). See ras_box_calibrate.

run_plots Logical. Save the diagnostic plots to save_dir. Default TRUE.

plot_device Character. "pdf" (default), "png" or "screen".

plot_p_threshold
Numeric. Significance reference line drawn on the plots, on the $-\log_{10}$ scale. Default 8.

plot_y_cap Numeric or NULL. Cap of the plotted y-axis. Default NULL.

Details

The box-scan threshold is calibrated on the data: unless box_calibration or box_threshold is supplied, the scan is repeated on box_null permuted phenotypes and the threshold is the order statistic of their maximum scores at level box_alpha (ras_box_calibrate), which controls the family-wise error rate without distributional assumptions. This multiplies the run time by about box_null; for large cohorts calibrate once with ras_box_calibrate and pass the result.

The pure-R, in-memory implementation that RAS 1.0.x shipped as ras() is available as ras_original; it defaults to the changepoint detector, and with detector = "changepoint" the two functions give the same scan profile to machine precision and the same detected positions.

Value

Invisibly, an object of class "ras": a list with scan (the ras_scan result), detection (for detector = "box" the ras_box_detect result, whose regions table has one row per interval; for detector = "changepoint" the ras_validate result, whose tau_hats are the detected positions), box_calibration (the calibration used, or NULL), chrom and save_dir. Use print() and plot() on it.

See Also

ras_scan, ras_detect, ras_validate, ras_box_detect, plot.ras; geno_to_rasbin and bed_to_rasbin to prepare the genotype file; ras_original for the pure-R implementation.

Examples

```
set.seed(3)
n_samp <- 120; n_snp <- 400
geno <- matrix(sample(0:2, n_samp * n_snp, replace = TRUE,
                      prob = c(0.6, 0.3, 0.1)), n_samp, n_snp)
causal <- 181:220                       # 40 causal SNPs, together explaining
pheno  <- 2 * as.numeric(scale(rowSums(geno[, causal]))) + rnorm(n_samp)  # ~80% of the variance
cov_df <- data.frame(age = rnorm(n_samp), sex = rbinom(n_samp, 1, 0.5))

## default: box-scan detector, threshold calibrated on permuted phenotypes
## (20 permutations with the Gumbel fit keep the example short; the
## default is the exact order statistic on 100 permutations)
res <- ras(geno, pheno, cov_df, covariate_cols = c("age", "sex"),
           is_continuous = TRUE, num_rep = 2, skip1 = 2, skip2 = 5,
           min_window_size = 2, max_window_size = 20,
           box_null = 20, box_method = "gumbel",
           save_dir = tempdir(), run_plots = FALSE)
res$detection$regions
res$box_calibration

## large data: convert once and pass the file; here with the changepoint
## detector of RAS 1.0.x (settings scaled down to the 200-point profile)
rb <- tempfile(fileext = ".rasbin")
geno_to_rasbin(geno, rb)
res_cp <- ras(rb, pheno, cov_df, covariate_cols = c("age", "sex"),
              is_continuous = TRUE, num_rep = 2, skip1 = 2, skip2 = 5,
              min_window_size = 2, max_window_size = 20,
              detector = "changepoint",
              cp_window_size = 100, cp_slope_check_window = 10,
              cp_slope_left = 1e-2, cp_slope_right = 1e-2,
              second_window_size = 20, second_p_threshold = 1e-2,
              save_dir = tempdir(), run_plots = FALSE)
res_cp$detection$tau_hats
unlink(c(rb, paste0(rb, ".meta.rds")))
```

ras-aliases *Aliases Kept from the Development Versions*

Description

In the development versions of this package the disk-backed C implementations carried a `_fast` suffix. Since RAS 1.1.0 they are the default implementations and carry the plain names, while the original pure-R, in-memory implementations carry an `_original` suffix. The `_fast` names are kept as thin aliases so that scripts written against the development versions keep working; new code should use the plain names.

Usage

```
ras_fast(...)

ras_scan_fast(...)

ras_detect_fast(...)

compute_gwas_weights_fast(...)

screen_forward_max_region_fast(...)

ras_scan_external_fast(...)
```

Arguments

`...`	Arguments passed on unchanged to the aliased function.

Value

Whatever the aliased function returns.

See Also

`ras`, `ras_scan`, `ras_detect`, `compute_gwas_weights`, `screen_forward_max_region`, `ras_scan_external`.

Examples

```
identical(formals(ras_fast), formals(ras))   # FALSE only because of `...`
```

rasbin_header *Read the .rasbin File Header*

Description

Read the .rasbin File Header

Usage

```
rasbin_header(path)
```

Arguments

path	Character. Path to a .rasbin file.

Value

Named numeric vector with elements n_samples, n_snps, dtype.

rasbin_read_chunk *Read a Column Range from a .rasbin File*

Description

Read a Column Range from a .rasbin File

Usage

```
rasbin_read_chunk(path, col_start, col_end)
```

Arguments

path	Character. Path to a .rasbin file.
col_start	Integer. First column to read (1-based, inclusive).
col_end	Integer. Last column to read (1-based, inclusive).

Value

An n_samples x (col_end - col_start + 1) numeric matrix.

ras_box_calibrate *Calibrate the Box-Scan Threshold on Null Profiles*

Description

Turns a set of null scan profiles into the threshold that ras_box_detect needs. Null profiles are obtained by running the same scan (ras_scan or ras_scan_external, with the same settings) on the same genotypes with the phenotype permuted, or with a phenotype simulated under the null.

Usage

```
ras_box_calibrate(
  null_profiles,
  alpha = 0.05,
  widths = .ras_box_default_widths(),
  gamma = 0.5,
  length_ratio = 1,
  calib = NULL,
  edge = c("open", "strict"),
  method = c("order", "gumbel")
)
```

Arguments

null_profiles	A list of numeric vectors, or a matrix with one null profile per row. All profiles should have the length of the profile that will be analysed, or the length that length_ratio refers to.
alpha	Numeric. Target family-wise error rate. Default 0.05.
widths	Integer vector. Window widths in grid points. Default c(5, 8, 11, 14, 17, 20, 24, 28, 34, 42, 55, 75, 100). Widths that do not fit the profile are dropped.
gamma	Numeric. Exponent of the level factor. Default 0.5.
length_ratio	Numeric. If the profile to be analysed is length_ratio times longer than the null profiles, the null maxima are fitted by a Gumbel law (method of moments) and the $(1-\alpha)^{1/length_ratio}$ quantile is returned as threshold_scaled. That number is an extrapolation past the data; null_exceed_scaled reports how many null maxima reach it, and a null of the full length is always preferable. Default 1.
calib	Integer vector. Indices of the profiles used to set the threshold; the remaining profiles, if any, give a held-out error rate (fwer_holdout). Default: all profiles.
edge	"open" (default) or "strict". With "strict" both flanks must lie inside the profile, so a region at a chromosome end can never be reported. With "open" a flank that runs off the profile is truncated, and dropped altogether when fewer than 3 points remain; the window then only has to stand above the side that exists (and above the background).

`method` Character. `"order"` (default): the threshold is the order statistic $k = \lceil (B+1)(1-\alpha) \rceil$ of the B null maxima, which bounds the family-wise error rate by α exactly but needs $B \geq 1/\alpha$ profiles and is noisy for small B. `"gumbel"`: the $1-\alpha$ quantile of a Gumbel law fitted to the null maxima by the method of moments; stable from about 20 profiles on, at the price of a slightly liberal threshold (in a permutation study on a 224-point profile, 20 to 50 profiles gave 97 to 98 percent of the 200-profile order-statistic threshold and a realised error rate of about 0.07 for $\alpha = 0.05$).

Details

`level0` is the median of the null profiles' medians. Both thresholds are always computed and returned (`threshold_order`, `threshold_gumbel`); `threshold` is the one selected by `method`.

Value

An object of class `"ras_box_calibration"`: a list with `level0`, `threshold`, `threshold_order`, `threshold_gumbel`, `threshold_scaled`, `method`, `length_ratio`, `maxT` (the maximum score of every null profile), `calib`, `alpha`, `gumbel` (fitted location and scale), `null_exceed_scaled`, `fwer_holdout` (`NA` when no profile was held out), `widths`, `gamma`, `edge`.

See Also

`ras_box_detect`, which consumes the result.

Examples

```
## 60 short synthetic null profiles (a real calibration uses several
## hundred profiles from the actual scan on permuted phenotypes)
set.seed(2)
nulls <- lapply(1:60, function(k) 0.7 + abs(rnorm(200, sd = 0.3)))
cal <- ras_box_calibrate(nulls, alpha = 0.05, calib = 1:40)
cal$threshold
cal$fwer_holdout           # error rate on the 20 held-out profiles

y <- nulls[[41]]; y[91:110] <- y[91:110] + 6
ras_box_detect(seq_along(y), y, calibration = cal)$regions
```

`ras_box_detect` *Detect Elevated Regions with the Box Scan*

Description

The box-scan region detector, the alternative to the changepoint detector (`ras_detect` + `ras_validate`) selected with `detector = "box"` in `ras`. It reports intervals $[\tau_L, \tau_R]$ of the scan profile whose mean stands above both adjacent flanks and above the profile background, which lets it delimit broad plateau-shaped association regions as well as sharp peaks.

Usage

```
ras_box_detect(
  x,
  y,
  threshold = NULL,
  calibration = NULL,
  scaled = FALSE,
  level0 = NULL,
  widths = NULL,
  gamma = NULL,
  edge = NULL,
  floor = 0.2,
  max_regions = Inf,
  keep_stat = FALSE
)
```

Arguments

x	Numeric vector. Grid positions of the profile (scan$x, the SNP index of each grid point).
y	Numeric vector. The scan profile (scan$y), same length as x.
threshold	Numeric. Keep regions with T_box >= threshold. Either threshold or calibration must be supplied.
calibration	A ras_box_calibrate result. Supplies threshold (or threshold_scaled when scaled = TRUE), level0, widths, gamma and edge; an explicitly supplied argument overrides the calibrated value.
scaled	Logical. With a calibration whose length_ratio > 1, use its threshold_scaled (the threshold extrapolated to a profile that many times longer than the null profiles) instead of threshold. Default FALSE.
level0, widths, gamma, edge	As in ras_box_stat; NULL (default) takes the value from calibration when one is given and the ras_box_stat default otherwise.
floor	Numeric. Windows with T below this value are never considered, whatever the threshold. Default 0.2.
max_regions	Integer. Stop after this many regions. Default Inf.
keep_stat	Logical. Also return the full ras_box_stat table as element stat. Default FALSE (the table has one row per window start and width, so it is large for a chromosome-length profile).

Details

Windows are scored with ras_box_stat and taken greedily by score; a window is dropped when it touches an already accepted one expanded by 10 + max(5, width/2) grid points, which removes side lobes. Each accepted window becomes one region; its anchor is the highest point of the profile inside it.

The threshold is a property of the null distribution of the maximum score along a profile and therefore depends on the profile's length and on the scan settings. Obtain it with ras_box_calibrate from null profiles produced by the same scan on permuted phenotypes; the order statistic it returns controls the family-wise error rate at the requested level without any distributional assumption.

Value

A list that plot.ras and print.ras accept as a detection element:

detector "box".

regions Data frame with one row per region, ordered by decreasing score: tau_L, tau_R (grid indices), pos_L, pos_R (the same boundaries in x units), width, T_box, contrast, height, anchor_idx, anchor_pos, anchor_y. NULL when nothing reaches the threshold.

tau_hats Numeric vector. Anchor positions in x units (the same role as in ras_validate's result).

all.changepoints, all.p.values Anchor positions and their T_box scores, so that the candidate overlay of plot.ras works.

left.slopes, right.slopes NULL; the box scan estimates no slopes.

threshold, level0, level, level_factor The threshold used, the null level, the median of y and the resulting level factor.

stat The ras_box_stat table when keep_stat = TRUE, otherwise NULL.

See Also

ras_box_calibrate for the threshold, ras_box_stat for the statistic, ras_detect for the changepoint detector, ras which calls this function when detector = "box".

Examples

```
set.seed(1)
x <- seq(1, 6000, by = 10)                # 600 grid points
y <- 0.7 + abs(rnorm(600, sd = 0.3))      # null-like background
y[301:342] <- y[301:342] + 8              # a 42-point plateau
det <- ras_box_detect(x, y, threshold = 2)
det$regions[, c("tau_L", "tau_R", "pos_L", "pos_R", "T_box")]

## a window inside the plateau scores about 0: only its edges are reported
S <- ras_box_stat(y)
max(S$T[S$w == 20 & S$i >= 305 & S$i + 19 <= 338])
```

---

ras_box_stat *Box-Scan Statistic for Every Window Start and Width*

Description

Computes the box-scan score $T(i, w)$ of every window of every width in widths along a RAS scan profile. This is the building block of ras_box_detect; most users will call that function instead.

Usage

```
ras_box_stat(
  y,
  widths = .ras_box_default_widths(),
  level0 = NA_real_,
  gamma = 0.5,
  edge = c("open", "strict")
)
```

Arguments

y	Numeric vector. The scan profile ($-\log_{10}(p)$ values from ras_scan or ras_scan_external). Must be free of NA.
widths	Integer vector. Window widths in grid points. Default c(5, 8, 11, 14, 17, 20, 24, 28, 34, 42, 55, 75, 100). Widths that do not fit the profile are dropped.
level0	Numeric. Typical median of a null profile (element level0 of a ras_box_calibrate result). NA (default) disables the level factor.
gamma	Numeric. Exponent of the level factor. Default 0.5.
edge	"open" (default) or "strict". With "strict" both flanks must lie inside the profile, so a region at a chromosome end can never be reported. With "open" a flank that runs off the profile is truncated, and dropped altogether when fewer than 3 points remain; the window then only has to stand above the side that exists (and above the background).

Details

For a window of w grid points starting at i, with flanks of $f = \max(5, w/2)$ points on each side,

$$contrast = mean(window) - max(mean(leftflank), mean(rightflank))$$

$$height = mean(window) - median(y)$$

$$T = min(contrast, height)/max(1, median(y)/level0)^{gamma}.$$

All quantities are in the raw units of the profile; no noise scale is estimated. A window inside a plateau, on the shoulder of a peak, or spanning a cluster of neighbouring peaks scores about 0 because at least one flank is as high as the window.

Value

A data frame with one row per (start, width) pair and columns i (window start, grid index), w (width), contrast, height and T.

See Also

ras_box_detect, ras_box_calibrate.

Examples

```
y <- rep(1, 400); y[201:230] <- 8
S <- ras_box_stat(y)
S[which.max(S$T), ]          # the 30-point window starting at 201
```

`ras_detect` *First-Pass Changepoint Detection via Sliding Window*

Description

Slides a window of `window_size` grid points across the scan profile. In each window a single-breakpoint segmented regression of `y` on `x` is fitted (Muggeo's algorithm with bootstrap restarts) and the Davies test for the existence of a breakpoint is applied. A breakpoint is kept as a candidate when the Davies p-value is below `p.values.threshold`, the fitted slope rises on its left and falls on its right, and one-tailed slope tests in a local window of half-width `slope_check_window_size` confirm both directions. Accepted candidates are snapped to the nearest local maximum of `y`. The result is passed to `ras_validate` for the second pass.

Usage

```
ras_detect(
  x,
  y,
  p.values.threshold = 0.01,
  min.length = 10,
  skip = 1,
  window_size = 3000,
  slope_check_window_size = 30,
  slope.p.values.threshold.left = 1e-10,
  slope.p.values.threshold.right = 1e-20
)
```

Arguments

`x`	Numeric vector. Grid positions of the profile (`scan$x`).
`y`	Numeric vector. Profile values (`scan$y`), same length as `x`.
`p.values.threshold`	Numeric. Davies test p-value threshold for a candidate. Default `0.01`.
`min.length`	Integer. Minimum number of grid points on each side of a candidate. Default `10`.
`skip`	Integer. Step, in grid points, between successive window starts. Default `1`.
`window_size`	Integer. Window width in grid points. Default `3000`; use a value below the profile length for short profiles.
`slope_check_window_size`	Integer. Half-width, in grid points, of the local window in which the left and right slopes are tested. Default `30`.

`slope.p.values.threshold.left`, `slope.p.values.threshold.right`
: Numeric. One-tailed p-value thresholds for the rising left slope and the falling right slope. Default `1e-10` and `1e-20`.

Details

The segmented fit and the Davies test are a port to compiled code of `segmented` and `davies.test`. The bootstrap restarts draw from R's random-number stream, so `set.seed()` makes a run reproducible, while results are statistically rather than bit-for-bit equal to the pure-R implementation `ras_detect_original`, which calls the `segmented` package.

Value

A list with `tau_hats` (accepted candidate positions, as indices into `x`), `p.values`, `slope.left`, `slope.right` and `slope.angle` for those candidates, `all.changepoints` and `all.p.values` for every window examined, and `previous_tau_hats` (a copy of `tau_hats` used by `ras_validate`).

See Also

`ras_validate` for the second pass; `ras_box_detect` for the alternative detector; `ras`, which runs both passes; `ras_detect_original`.

Examples

```
set.seed(42)
x <- 1:100
y <- c(seq(0, 8, length.out = 50), seq(8, 1, length.out = 50)) +
     rnorm(100, sd = 0.5)
cp <- ras_detect(x, y, window_size = 50, slope_check_window_size = 10,
                 slope.p.values.threshold.left  = 1e-3,
                 slope.p.values.threshold.right = 1e-3)
cp$tau_hats
```

---

ras_detect_original *First-Pass Changepoint Detection (Pure-R)*

Description

This is the pure-R, in-memory implementation that RAS 1.0.x shipped under the plain name. Since 1.1.0 the plain name (`ras_detect`) is the compiled, disk-backed implementation; this function is kept for reference and gives the same results. Scans a $-\log_{10}(p)$-value sequence using a sliding window to detect positions where the slope changes significantly from positive to negative.

Usage

```
ras_detect_original(
  x,
  y,
  p.values.threshold = 0.01,
  min.length = 10,
  skip = 1,
  window_size = 3000,
  slope_check_window_size = 30,
  slope.p.values.threshold = 1e-08,
  slope.p.values.threshold.left = 1e-10,
  slope.p.values.threshold.right = 1e-20
)
```

Arguments

`x`	Numeric vector. Predictor sequence (e.g., SNP position indices).
`y`	Numeric vector. Response sequence (e.g., $-\log_{10}(p)$ values from `ras_scan_original`). Must be the same length as `x`.
`p.values.threshold`	Numeric. Davies test p-value threshold for nominating a candidate changepoint. Default `0.01`.
`min.length`	Integer. Minimum number of observations required on each side of a candidate changepoint. Default `10`.
`skip`	Integer. Step size when iterating the sliding window start position. Default `1`.
`window_size`	Integer. Number of observations in each sliding window. Default `3000`.
`slope_check_window_size`	Integer. Half-width of the local region used to verify slope direction via `slope_test`. Default `30`.
`slope.p.values.threshold`	Numeric. Reserved combined slope threshold (currently unused in filtering). Default `1e-8`.
`slope.p.values.threshold.left`	Numeric. One-tailed p-value threshold for the left-side slope test. A candidate is accepted only if the slope to its left is significantly positive ($p <$ this value). Default `1e-10`.
`slope.p.values.threshold.right`	Numeric. One-tailed p-value threshold for the right-side slope test. A candidate is accepted only if the slope to its right is significantly negative ($p <$ this value). Default `1e-20`.

Details

At each window start position the function calls `get_break_points` on the window to estimate and test a single breakpoint. A candidate is retained when three conditions are all met:

1. The Davies test p-value is below **p.values.threshold**.

2. The left-side slope (estimated by segmented regression) is positive.
3. The right-side slope is negative.

Candidates passing these three filters are then subjected to one-tailed slope tests (slope_test) using centred sub-sequences of half-width **slope_check_window_size**. Only candidates that also pass both slope p-value thresholds are recorded.

After the sliding-window loop, each accepted changepoint is refined to the nearest local peak in **y** via get_local_maximum.

The function records all candidates examined (all.changepoints, all.p.values) in addition to the accepted ones, so that downstream functions can display the full candidate landscape.

Value

A named list with eight elements:

tau_hats Integer vector. Accepted changepoint indices (refined to local peaks).

p.values Numeric vector. Davies test p-values at accepted changepoints.

slope.left Numeric vector. Left-side slopes at accepted changepoints.

slope.right Numeric vector. Right-side slopes at accepted changepoints.

all.changepoints Integer vector. All candidate positions examined, including those that failed the slope tests.

all.p.values Numeric vector. Davies p-values for all examined candidates; set to 1 for candidates that failed the slope direction or slope significance filters.

slope.angle Numeric vector. Interior angle (degrees) at each accepted changepoint, computed from the left and right slope estimates.

previous_tau_hats Integer vector. Copy of tau_hats before any downstream modification; used by ras_validate.

See Also

ras_validate for the second-pass validation step. get_break_points for the per-window segmented regression. slope_test for the one-tailed slope verification. get_local_maximum for the peak-refinement step. ras_original for the recommended end-to-end entry point.

Examples

```
set.seed(42)
x <- 1:100
y <- c(seq(0, 8, length.out = 50),
       seq(8, 1, length.out = 50)) + rnorm(100, sd = 0.5)
result <- ras_detect_original(
  x, y,
  window_size             = 50,
  skip                    = 2,
  slope_check_window_size = 10,
  slope.p.values.threshold.left  = 1e-3,
  slope.p.values.threshold.right = 1e-3
)
```

```
cat("Detected changepoints:", result$tau_hats, "\n")
```

`ras_harmonize_sumstats` *Harmonise External GWAS Summary Statistics into RAS Weights*

Description

The entry point for summary-informed RAS. Takes a raw external summary-statistics file and a local genotype map, chooses the naming/build route from the data rather than from an assumption, and returns a verified weight vector together with a reproducible QC table.

Usage

```
ras_harmonize_sumstats(
  sumstats,
  map,
  dict = NULL,
  chain = NULL,
  chain_back = NULL,
  cup = NULL,
  converter = .liftover_converter_ucsc,
  check_reverse = TRUE,
  assume_same_build = FALSE,
  cols = list(),
  ...
)
```

Arguments

`sumstats`	Path to the external summary statistics, or a data frame.
`map`	Data frame with columns `SNP`, `CHR`, `BP`, `A1`, `A2`: the local genotype map, one row per genotype column, in order. `A1` must be the allele the local dosage counts.
`dict`	Optional variant dictionary ON THE MAP'S BUILD, from `ras_read_variant_dictionary`. Enables the dictionary route, which fixes a naming disagreement without touching any coordinate.
`chain, chain_back`	Optional liftOver chain files, enabling the coordinate-conversion route. `chain_back` is required unless `check_reverse = FALSE`.
`cup`	Optional data frame (`CHR, START, END`) of conversion-unstable intervals on the source build, passed to the liftOver route.
`converter`	Conversion function for the liftOver route; the default shells out to the UCSC `liftOver` binary.
`check_reverse`	Logical. Round-trip every liftOver conversion and drop what cannot return to its own position (default `TRUE`).

`assume_same_build`
: Logical. Assert, on the caller's authority, that both sides use the same genome build. Only then will positions be used without the build gate having verified them. Appropriate when the summary statistics are a GWAS Catalog harmonised (GRCh38) file and the map is GRCh38.

`cols`
: Optional column-name overrides, passed through.

`...`
: Passed to `ras_weights_from_sumstats`, e.g. `drop_ambiguous`, `match_min_prop`, `target_af`, `block_size`.

Details

The driver never guesses about the genome build. It inspects both sides' ID conventions and then either matches directly, translates the summary statistics through a dictionary, lifts their coordinates over, or refuses – listing exactly what would unblock it. Every path ends at `ras_weights_from_sumstats` with `build_check = TRUE` unless the caller has explicitly asserted the builds agree or a liftOver round trip has already established them.

Value

A list with

`weights`
: Numeric vector of length `nrow(map)`, ready to pass to `ras_scan_external`.

`qc`
: A data frame with one row per harmonisation stage (`item`, `n`, `denom`, `pct`): external variants read, dropped for a missing beta, dropped as duplicates, matched by rsID, matched by position, absent from the target, allele-aligned, allele-flipped, palindromic removed, and final usable weights. Written to be quoted directly in a manuscript.

`route`
: The route actually taken.

`trail`
: Human-readable audit trail of every decision.

`weights_result`
: The full `ras_weights_from_sumstats` result, including `detail`, `build` and `blocks`.

`sumstats_aligned`
: The summary statistics after any translation or conversion.

See Also

`ras_scan_external` to scan with these weights; `ras_sumstats_report` to inspect an alignment without committing to it.

`ras_liftover_sumstats` *LiftOver External Summary Statistics onto the Map's Build (Route B2)*

Description

Converts the summary statistics' coordinates to another genome build and verifies every conversion by round-tripping it (`from -> to -> from`); anything that cannot return to its own chromosome and position is dropped. This is the LAST-RESORT route. liftOver has real error rates (about 1.7% of positions discordant GRCh37 to GRCh38, 3.1% the other way), so prefer a dictionary lookup (`ras_read_variant_dictionary`) when one is available: a lookup can only fail to find a variant, it cannot find the wrong one.

Usage

```
ras_liftover_sumstats(
  ss,
  chain,
  chain_back = NULL,
  cols = list(),
  converter = .liftover_converter_ucsc,
  check_reverse = TRUE,
  cup = NULL,
  ...
)
```

Arguments

`ss`	Data frame of summary statistics, or a path to one.
`chain`	Path to the forward chain file, e.g. `hg19ToHg38.over.chain`.
`chain_back`	Path to the reverse chain file. Required unless `check_reverse = FALSE`.
`cols`	Optional column-name overrides, e.g. `list(chr = "CHROM")`.
`converter`	Function performing the conversion; the default shells out to UCSC `liftOver`. Replaceable for testing.
`check_reverse`	Logical. Verify each conversion by round-tripping it (default `TRUE`). Setting `FALSE` accepts unverified conversions.
`cup`	Optional data frame (`CHR`, `START`, `END`) of conversion-unstable intervals on the SOURCE build; overlapping variants are dropped first.
`...`	Passed to `converter`.

Details

Requires the UCSC `liftOver` binary on `PATH` unless a replacement `converter` is supplied.

Value

A list with `sumstats` (only successfully converted rows, with the pre-conversion position kept in `BP_source`) and `report` (a per-status count table).

See Also

`ras_harmonize_sumstats`

`ras_map_ids_to_rsid` *Rename a Local Genotype Map's Variants to rsIDs (Harmonisation Route B0)*

Description

Gives the LOCAL map rsIDs using a dictionary on the SAME genome build, so that afterwards both sides are keyed on rsID, the build gate becomes evaluable, and nothing about positions has to be trusted. Preferred when the map is ours – inside All of Us the dictionary is the Variant Annotation Table.

Usage

```
ras_map_ids_to_rsid(map, dict, update_name_file = NULL)
```

Arguments

`map`	Data frame with columns `SNP`, `CHR`, `BP`, `A1`, `A2`: the local genotype map, one row per genotype column, in order.
`dict`	A dictionary from `ras_read_variant_dictionary`, on the map's own build.
`update_name_file`	Optional path. When given, a two-column PLINK `--update-name` file (old ID, new ID; no header) is written there.

Details

Matching is chr:pos plus the unordered allele pair, so multi-allelic sites cannot be confused. Variants with no dictionary entry keep their original ID (they simply will not match the summary statistics afterwards).

Value

A list with `map` (the renamed map, carrying the original IDs in `SNP_original`), `report` (counts), and `update_name` (the rename pairs as a data frame).

See Also

`ras_harmonize_sumstats`

ras_memory *Estimate Memory Requirements and Check System Readiness*

Description

Detects the current machine's RAM and CPU configuration, computes per-stage peak memory estimates for the RAS pipeline, and issues a go / no-go verdict.

Usage

```
ras_memory(
  n_total,
  n_train,
  n_holdout,
  n_snps,
  bytes_per_element = 8,
  abort = FALSE
)
```

Arguments

n_total	Integer. Total number of samples (rows of geno).
n_train	Integer. Number of training-split samples (typically n_total / 2).
n_holdout	Integer. Number of hold-out samples (rows of pgs.mat; typically n_total - n_train).
n_snps	Integer. Number of variants (columns of geno).
bytes_per_element	Numeric. Bytes per matrix element. Default 8 (R double).
abort	Logical. If TRUE and estimated peak memory exceeds available RAM, calls stop so the pipeline cannot proceed. Default FALSE.

Details

Peak memory is estimated for three pipeline stages:

Stage 1: compute_gwas_weights_original Holds the full geno matrix plus a small $N \times 4$ coefficient matrix. Peak ≈ geno_mb + coefmat_mb.

Stage 2: compute_pgs_matrix Worst-case holds two copies of geno (R copy-on-modify triggered by geno[is.na(geno)] <- 0) plus the output pgs.mat. Peak ≈ 2 * geno_mb + pgsmat_mb.

Stage 3: screen_forward_max_region_original Holds geno and pgs.mat simultaneously. Peak ≈ geno_mb + pgsmat_mb.

The overall estimated peak is the maximum across the three stages.

Available RAM is queried via wmic on Windows and /proc/meminfo on Linux; the function degrades gracefully (prints a caution message and returns can_proceed = TRUE) if the query fails.

The function prints a formatted report to the console and invisibly returns the numeric estimates for programmatic use.

Value

Invisibly returns a named list with two elements:

`memory` Named numeric vector with elements `geno_mb`, `pgsmat_mb`, `coefmat_mb`, `stage1_peak_mb`, `stage2_peak_mb`, `stage3_peak_mb`, `overall_peak_mb`, `available_mb`, `total_mb`.

`system` Named list with elements `cores_physical`, `cores_logical`, `r_version`, `platform`, and `can_proceed` (logical: `TRUE` if RAM is sufficient or cannot be measured).

See Also

`ras_scan_original`, `ras_original` for the functions whose memory use is being estimated. `release_memory` to reclaim heap memory after each repetition.

Examples

```
## Check feasibility for 5,000 samples and 500,000 SNPs
ras_memory(
  n_total   = 5000,
  n_train   = 2500,
  n_holdout = 2500,
  n_snps    = 500000
)

## Abort if memory is insufficient (wrapped in try() so the example runs)
try(ras_memory(
  n_total   = 100000,
  n_train   = 50000,
  n_holdout = 50000,
  n_snps    = 1000000,
  abort     = TRUE
))
```

---

ras_original *RAS Pipeline (Pure-R, In-Memory)*

Description

This is the pure-R, in-memory implementation that RAS 1.0.x shipped under the plain name. Since 1.1.0 the plain name (`ras`) is the compiled, disk-backed implementation; this function is kept for reference and gives the same results. Detect significant association regions in GWAS data. Returns an object of class `"ras"`.

Usage

```
ras_original(
  geno,
  phenotype,
  covariates,
```

```
  covariate_cols,
  is_continuous,
  num_rep = 5,
  skip1 = 10,
  skip2 = 20,
  chrom = 1,
  save_dir = file.path(tempdir(), "RAS"),
  min_window_size = 5,
  max_window_size = 100,
  scan_test = c("glm", "score"),
  cp_p_threshold = 0.01,
  cp_window_size = 3000,
  cp_min_length = 10,
  cp_slope_check_window = 30,
  cp_slope_left = 1e-10,
  cp_slope_right = 1e-20,
  second_window_size = 50,
  second_p_threshold = 1e-10,
  min_signal = 2.5,
  detector = c("changepoint", "box"),
  box_threshold = NULL,
  box_calibration = NULL,
  box_level0 = NA_real_,
  run_plots = TRUE,
  plot_device = "pdf",
  plot_p_threshold = 8,
  plot_y_cap = NULL
)
```

Arguments

`geno`	Matrix of genotype dosages (n samples $\times$ N variants). Rows must be aligned with **phenotype** and **covariates**.
`phenotype`	Numeric vector of length n. Raw phenotype values. For continuous traits the function residualises covariates from the training phenotype internally each repetition; pass the original un-residualised values here.
`covariates`	Data frame with n rows aligned with **geno**. Must contain all columns named in **covariate_cols**; additional columns (e.g., an ID column) are silently ignored.
`covariate_cols`	Character vector of column names in **covariates** to include in the regression models.
`is_continuous`	Logical. `TRUE` for quantitative traits, `FALSE` for binary (case/control) traits.
`num_rep`	Integer. Number of independent 50/50 splits to average over. Default `5`.
`skip1`	Integer. Step size for the primary SNP position grid (`seq(1, N, by = skip1)`). Default `10`.
`skip2`	Integer. Sub-step used to grow the scan window at each grid position. Default `20`.

`chrom` Integer. Chromosome number used in saved file names. Default `1`.

`save_dir` Character. Directory for intermediate `.rds` output files; created recursively if it does not exist. Defaults to a per-session subdirectory of `tempdir()`; set an explicit path to keep the outputs.

`min_window_size`
Integer. Minimum scan window half-size passed to `screen_forward_max_region_original`. Default `5`.

`max_window_size`
Integer. Maximum scan window half-size passed to `screen_forward_max_region_original`. Default `100`.

`scan_test` Character. Per-window test for binary traits, passed to `screen_forward_max_region_original`. `"glm"` (default) fits a full logistic model per window (Wald p-value); `"score"` fits the null logistic model once and uses a Rao score test per window (asymptotically equivalent, much faster at scale). Has no effect for continuous traits.

`cp_p_threshold` Numeric. Davies test p-value threshold for first-pass candidate changepoints. Default `0.01`.

`cp_window_size` Integer. Sliding window width for first-pass detection. Default `3000`.

`cp_min_length` Integer. Minimum segment length before and after a candidate changepoint. Default `10`.

`cp_slope_check_window`
Integer. Half-width of the local window used to verify slope direction around each candidate. Default `30`.

`cp_slope_left` Numeric. One-tailed p-value threshold for the left-side slope test (tests that the slope to the left of the candidate is significantly positive). Default `1e-10`.

`cp_slope_right` Numeric. One-tailed p-value threshold for the right-side slope test (tests that the slope to the right is significantly negative). Default `1e-20`.

`second_window_size`
Integer. Half-window size for second-pass local Davies tests. Default `50`.

`second_p_threshold`
Numeric. Davies test p-value threshold for second-pass validation. Default `1e-10`.

`min_signal` Numeric. Minimum $-\log_{10}(p)$ scan value required to retain a validated changepoint (passed to `ras_validate`); also used as the low-signal colour boundary in all diagnostic plots. Default `2.5`.

`detector` Character. Which region detector to run on the scan profile: `"changepoint"` (default) is the original two-pass detector, `ras_detect_original` followed by `ras_validate`, which reports single changepoint positions; `"box"` is the box-scan region detector `ras_box_detect`, which reports intervals $[\tau_L, \tau_R]$ and also delimits broad plateau-shaped regions. The `cp_*`, `second_*` and `min_signal` arguments apply to `"changepoint"` only; the `box_*` arguments to `"box"` only.

`box_threshold` Numeric. Threshold on the box-scan score for `detector = "box"`. Required unless `box_calibration` is given; see `ras_box_calibrate` for how to obtain it.

`box_calibration`
A `ras_box_calibrate` result for `detector = "box"`; supplies the threshold and `level0`.

box_level0 Numeric. Typical median of a null profile for the box scan's level factor (see ras_box_stat). NA (default) takes it from box_calibration when one is given and disables the level factor otherwise.

run_plots Logical. If TRUE (default), saves diagnostic plots via plot_ras_scan and plot_ras_zoom_regions.

plot_device Character. Output device for plots: "pdf" (default), "png", or "screen".

plot_p_threshold Numeric. Significance reference line drawn on plots (on the $-\log_{10}$ scale). Default 8.

plot_y_cap Numeric or NULL. If provided, the y-axis is capped at this value and positions above the cap are annotated with arrows. Default NULL (no cap).

Details

For a stage-by-stage description of the pipeline see RAS.

Two region detectors are available for Stage 2. The default detector = "changepoint" reproduces the published method and the results of earlier versions of this package exactly. detector = "box" runs ras_box_detect instead; it needs a threshold, which is a property of the null distribution of the scan and is obtained with ras_box_calibrate from null profiles (the same scan on permuted phenotypes).

Value

Invisibly returns a named list with two elements:

scan A list with elements:

- x Integer vector. SNP position index grid seq(1, N, by = skip1).
- y Numeric vector (same length as x). Averaged $-\log_{10}(p)$-value profile across all repetitions.

detection For detector = "changepoint", the list returned by ras_validate with elements:

- tau_hats Integer vector. Validated changepoint positions (re-mapped to genomic coordinates via **this.start** and **this.skip**).
- all.changepoints Integer vector. All candidate positions examined during the first pass, re-mapped to genomic coordinates.
- all.p.values Numeric vector. $-\log_{10}(p)$ Davies test values for every candidate in all.changepoints.
- left.slopes Numeric vector. Estimated left-side slopes at each validated changepoint.
- right.slopes Numeric vector. Estimated right-side slopes at each validated changepoint.
- detector "changepoint".

For detector = "box", the list returned by ras_box_detect: regions (one row per interval with pos_L, pos_R, T_box, ...), tau_hats (the anchor of each interval) and the threshold used.

Plot files (if run_plots = TRUE) are written to save_dir with names chr-<chrom>-cp-plot.<ext>, chr-<chrom>-cp-p-values-plot.<ext>, and chr-<chrom>-zoom.<ext>.

Simple (recommended)

```
## one call does everything
result <- ras_original(geno, phenotype, covariates,
              covariate_cols = c("age", "sex", paste0("pc", 1:10)),
              is_continuous  = TRUE,
              chrom = 1, save_dir = "results/")

print(result)             # detected changepoint positions
plot(result)              # full-chromosome scan profile
plot(result, zoom = TRUE) # zoomed view around each changepoint
```

For step-by-step control see ras_scan_original, ras_detect_original, ras_validate.

See Also

ras_scan_original for the scan-only step (useful when tuning changepoint parameters separately). ras_detect_original, ras_validate for the changepoint detection steps. plot.ras for the plotting step. ras_memory to check memory requirements before loading large genotype data.

Examples

```
set.seed(42)
n_samp <- 80; n_snp <- 60
geno <- matrix(
  sample(0:2, n_samp * n_snp, replace = TRUE, prob = c(0.6, 0.3, 0.1)),
  nrow = n_samp, ncol = n_snp)
pheno <- rnorm(n_samp)
cov_df <- data.frame(age = rnorm(n_samp), sex = rbinom(n_samp, 1, 0.5))
cov_df$age_squared <- cov_df$age^2
cov_df$age_sex     <- cov_df$age * cov_df$sex
for (i in 1:10) cov_df[[paste0("pc", i)]] <- rnorm(n_samp)

result <- ras_original(
  geno              = geno,
  phenotype         = pheno,
  covariates        = cov_df,
  covariate_cols    = c("age", "sex", "age_squared", "age_sex",
                        paste0("pc", 1:10)),
  is_continuous     = TRUE,
  num_rep           = 2,
  skip1             = 1,        # every SNP on the grid: a 60-point profile
  skip2             = 2,
  min_window_size   = 2,
  max_window_size   = 10,
  # detector settings scaled down to the 60-point toy profile (the defaults
  # are tuned for genome-scale grids with tens of thousands of points)
  cp_window_size        = 30,
  cp_min_length         = 5,
  cp_slope_check_window = 5,
  cp_slope_left         = 1e-3,
  cp_slope_right        = 1e-3,
```

```
  second_window_size    = 10,
  second_p_threshold    = 1e-3,
  chrom             = 1,
  save_dir          = tempdir(),
  run_plots         = TRUE,
  plot_device       = "pdf"
)

cat("Detected changepoints:", result$detection$tau_hats, "\n")
plot(result$scan$x, result$scan$y, type = "l",
     xlab = "SNP index", ylab = expression(-log[10](p)),
     main = "RAS scan profile")

## The box-scan region detector on the same scan profile. In the one-call
## form this is ras_original(..., detector = "box", box_threshold = 2); the threshold
## would normally come from ras_box_calibrate() on null profiles.
result_box <- result
result_box$detection <- ras_box_detect(result$scan$x, result$scan$y,
                                       threshold = 2)
print(result_box)
```

`ras_read_variant_dictionary` *Read a Variant Dictionary for Summary-Statistics Harmonisation*

Description

Reads a table mapping rsID to (chromosome, position, REF, ALT) on ONE genome build, for use by `ras_harmonize_sumstats`'s dictionary route and by `ras_map_ids_to_rsid`. Sources include a dbSNP slice, the All of Us Variant Annotation Table, or any delimited file with those columns.

Usage

```
ras_read_variant_dictionary(src, cols = list(), build = NA_character_)
```

Arguments

`src`	data.frame, or a path to a delimited file / VCF.
`cols`	Optional named overrides for the auto-detected columns, e.g. `list(rsid = "dbsnp_rsid")`. Names are `rsid`, `chr`, `pos`, `ref`, `alt`.
`build`	Free-text label recorded on the result for provenance, e.g. `"GRCh38"`. Never inferred and never acted upon – it is an audit note, while the build gate does the actual checking.

Details

A plain VCF is accepted: lines beginning `##` are skipped and `#CHROM` is used as the header. An `ALT` field carrying several comma-separated alleles is expanded to one row per alternate allele, so multi-allelic sites resolve by their alleles rather than by row order.

Value

A data frame with columns RSID, CHR, BP, REF, ALT, KEY (the unordered allele-pair key), with the build label attached as an attribute.

See Also

ras_harmonize_sumstats, ras_map_ids_to_rsid

---

ras_scan *RAS Stage 1: Averaged Regional Association Profile*

Description

Computes the $-\log_{10}(p)$ profile that the RAS detectors work on. The sample is split at random into a training half and a hold-out half num_rep times; in each repetition per-SNP regression weights are fitted on the training half (compute_gwas_weights) and a forward regional scan is run on the hold-out half (screen_forward_max_region); the profiles are averaged over repetitions. Genotypes are streamed from a .rasbin file in chunks of chunk_snps SNP columns, so peak memory does not grow with the chromosome. For a genotype matrix small enough to sit in memory the pure-R implementation ras_scan_original gives the same profile.

Usage

```
ras_scan(
  geno,
  phenotype,
  covariates,
  covariate_cols,
  is_continuous,
  num_rep = 5,
  skip1 = 10,
  skip2 = 20,
  chrom = 1,
  save_dir = file.path(tempdir(), "RAS"),
  min_window_size = 5,
  max_window_size = 100,
  scan_test = c("score", "glm"),
  chunk_snps = 5000,
  cores = 1,
  keep_reps = FALSE,
  rows = NULL
)
```

Arguments

geno	Either the path to a .rasbin genotype file (written by geno_to_rasbin or bed_to_rasbin), or an in-memory numeric genotype matrix (n samples by N variants). A matrix is converted to a temporary .rasbin file for the run; convert once with geno_to_rasbin when running repeatedly.
phenotype	Numeric vector of length n, in the sample order of geno.
covariates	Data frame with n rows, in the sample order of geno.
covariate_cols	Character vector. Names of the columns of covariates to adjust for.
is_continuous	Logical. TRUE for a quantitative trait, FALSE for a binary (0/1) trait.
num_rep	Integer. Number of train/hold-out repetitions to average over. Default 5.
skip1	Integer. Stride of the profile grid in SNPs: the profile has one value every skip1 SNPs. Default 10.
skip2	Integer. Step, in SNPs, by which the scan window grows from min_window_size to max_window_size at each grid position. Default 20.
chrom	Integer or character. Chromosome label used in the output file names. Default 1.
save_dir	Character or NULL. Directory that receives the per-repetition coefficient matrices and the averaged profile; NULL writes nothing. Default file.path(tempdir(), "RAS").
min_window_size, max_window_size	Integer. Smallest and largest scan window, in SNPs. Default 5 and 100.
scan_test	Character. Per-window test for a binary trait: "score" (default, Rao score test in closed form) or "glm" (per-window logistic regression, the default of RAS 1.0.x). "glm" runs the in-memory implementation ras_scan_original and therefore needs a genotype matrix, not a .rasbin path. Ignored for continuous traits.
chunk_snps	Integer. SNP columns held in memory at a time. Bounds peak memory at roughly n * chunk_snps * 8 bytes per worker. Default 5000.
cores	Integer. Number of worker processes the repetitions are spread over (a PSOCK cluster). Default 1. Each worker draws its own random-number stream, so results with cores > 1 are not bit-identical to a serial run with the same seed, and peak memory grows roughly in proportion to cores.
keep_reps	Logical. Also return (as $reps) and save the individual per-repetition profiles, which are needed to study the split-to-split variability that the average removes. Default FALSE.
rows	Integer vector or NULL. Row indices of the samples to use; the train/hold-out splits are drawn within this set. NULL (default) uses all samples. Used by ras to calibrate the box-scan threshold on a random subset of a large cohort.

Value

Invisibly, a list with x (the SNP index of each grid point, seq(1, N, by = skip1)), y (the averaged $-\log_{10}(p)$ profile) and reps (a length(x) by num_rep matrix of per-repetition profiles when keep_reps = TRUE, otherwise NULL). The averaged profile is also written to save_dir as mean_p_values_chr<chrom>_reps1-<num_rep>.rds.

See Also

ras for the full pipeline; ras_detect and ras_box_detect for the detectors that consume the profile; ras_scan_external for a scan with external weights; ras_scan_original for the pure-R implementation.

`ras_scan_external` *RAS Scan with External Weights on All Samples*

Description

Runs the Stage-1 forward scan with weights taken from an independent external GWAS instead of a within-sample training split. Because the weights are independent of the target cohort, no 50/50 split is drawn and no repetitions are averaged: every sample is scanned once. Genotypes are streamed from a `.rasbin` file in chunks. The in-memory implementation is `ras_scan_external_original`.

Usage

```
ras_scan_external(
  geno,
  phenotype,
  covariates,
  covariate_cols,
  weights,
  is_continuous = TRUE,
  skip1 = 10,
  skip2 = 20,
  min_window_size = 5,
  max_window_size = 100,
  chunk_snps = 5000,
  chrom = 1,
  save_dir = NULL
)
```

Arguments

`geno`	Either the path to a `.rasbin` genotype file (see `geno_to_rasbin`, `bed_to_rasbin`) or an in-memory numeric genotype matrix, which is converted to a temporary `.rasbin` file for the run.
`phenotype`	Numeric vector of length n, in the sample order of `geno`.
`covariates`	Data frame with n rows, in the same sample order.
`covariate_cols`	Character vector of covariate column names.
`weights`	Numeric vector of length N, one weight per variant in the order of `geno`, typically `ras_harmonize_sumstats(...)$weights`. Non-finite entries count as zero.
`is_continuous`	Logical. `TRUE` for a quantitative trait.

`skip1` Integer. Stride of the profile grid in SNPs: the profile has one value every `skip1` SNPs. Default `10`.

`skip2` Integer. Step, in SNPs, by which the scan window grows from `min_window_size` to `max_window_size` at each grid position. Default `20`.

`min_window_size, max_window_size` Integer. Smallest and largest scan window, in SNPs. Default `5` and `100`.

`chunk_snps` Integer. SNP columns held in memory at a time. Bounds peak memory at roughly `n * chunk_snps * 8` bytes per worker. Default `5000`.

`chrom` Integer or character. Chromosome label used in the output file names. Default `1`.

`save_dir` Character or `NULL`. Directory in which to save the profile as `ext_scan_chr<chrom>.rds`; `NULL` (default) saves nothing.

Details

The thresholds of the changepoint detector were set on averaged profiles, whose noise floor is lower than that of a single-pass profile; calibrate before treating a detection from this scan as a finding (for the box-scan detector, `ras_box_calibrate` on permuted phenotypes).

Value

A list with `x` (the SNP index of each grid point) and `y` (the $-\log_{10}(p)$ profile).

See Also

`ras_harmonize_sumstats` to build `weights`; `ras_scan` for the split-based scan; `ras_box_detect` and `ras_detect` to detect regions on the profile.

---

ras_scan_external_original

Summary-Informed RAS Scan on All Samples (In-Memory)

Description

The in-memory counterpart of `ras_scan_external`, for cohorts small enough that a dense $n \times m$ genotype matrix and its PGS matrix fit in RAM. At biobank scale use `ras_scan_external` instead: at 453,698 samples by 332,690 variants the dense double alone is over a petabyte.

Usage

```
ras_scan_external_original(
  geno,
  phenotype,
  covariates,
  covariate_cols,
  weights,
```

```
  is_continuous = TRUE,
  skip1 = 10,
  skip2 = 20,
  min_window_size = 5,
  max_window_size = 100,
  scan_test = c("glm", "score"),
  chrom = 1,
  save_dir = NULL
)
```

Arguments

geno	Numeric $n \times m$ genotype dosage matrix.
phenotype	Numeric vector of length n.
covariates	Data frame with n rows containing covariate_cols.
covariate_cols	Character vector of covariate column names.
weights	Numeric vector of length m, aligned to the columns of geno.
is_continuous	Logical. TRUE for quantitative traits.
skip1, skip2, min_window_size, max_window_size	Scan geometry, as in ras_scan_original.
scan_test	"glm" (per-window Wald) or "score" (Rao score test).
chrom	Integer/character. Chromosome label used in output filenames.
save_dir	Character or NULL. Directory to save the profile in.

Value

A list with x (grid point variant indices) and y (the profile).

See Also

ras_scan_external, ras_harmonize_sumstats

---

ras_scan_original *RAS Stage 1 Scan (Pure-R, In-Memory)*

Description

This is the pure-R, in-memory implementation that RAS 1.0.x shipped under the plain name. Since 1.1.0 the plain name (ras_scan) is the compiled, disk-backed implementation; this function is kept for reference and gives the same results. Computes the averaged $-\log_{10}(p)$-value profile over **num_rep** independent 50/50 train/hold-out splits.

Usage

```
ras_scan_original(
  geno,
  phenotype,
  covariates,
  covariate_cols,
  is_continuous,
  num_rep = 5,
  skip1 = 10,
  skip2 = 20,
  chrom = 1,
  save_dir = file.path(tempdir(), "RAS"),
  min_window_size = 5,
  max_window_size = 100,
  scan_test = c("glm", "score")
)
```

Arguments

`geno`	Matrix of genotype dosages (n samples $\times$ N variants). Rows must be aligned with **phenotype** and **covariates**.
`phenotype`	Numeric vector of length n. Raw phenotype values. For continuous traits the function residualises covariates from the training phenotype internally each repetition; pass the original un-residualised values here.
`covariates`	Data frame with n rows aligned with **geno**. Must contain all columns named in **covariate_cols**; additional columns (e.g., an ID column) are silently ignored.
`covariate_cols`	Character vector of column names in **covariates** to include in the regression models.
`is_continuous`	Logical. `TRUE` for quantitative traits, `FALSE` for binary (case/control) traits.
`num_rep`	Integer. Number of independent 50/50 splits to average over. Default `5`.
`skip1`	Integer. Step size for the primary SNP position grid (`seq(1, N, by = skip1)`). Default `10`.
`skip2`	Integer. Sub-step used to grow the scan window at each grid position. Default `20`.
`chrom`	Integer. Chromosome number used in saved file names. Default `1`.
`save_dir`	Character. Directory for intermediate `.rds` output files; created recursively if it does not exist. Defaults to a per-session subdirectory of `tempdir()`; set an explicit path to keep the outputs.
`min_window_size`	Integer. Minimum scan window half-size passed to `screen_forward_max_region_original`. Default `5`.
`max_window_size`	Integer. Maximum scan window half-size passed to `screen_forward_max_region_original`. Default `100`.

`scan_test` Character. Per-window test for binary traits, passed to `screen_forward_max_region_original`. `"glm"` (default) fits a full logistic model per window (Wald p-value); `"score"` fits the null logistic model once and uses a Rao score test per window (asymptotically equivalent, much faster at scale). Has no effect for continuous traits.

Details

For each of the **num_rep** repetitions the function performs three steps:

1. **GWAS weights.** A fresh 50/50 random split of all n samples is drawn. `compute_gwas_weights_original` fits a per-SNP regression on the training half and returns an $N \times 4$ coefficient matrix; the first column (effect-size estimates) is used as PGS weights.
2. **PGS contribution matrix.** `compute_pgs_matrix` multiplies each hold-out individual's dosage vector by the per-SNP weights, producing an $n_{\text{holdout}} \times N$ matrix of weighted contributions.
3. **Forward scan.** `screen_forward_max_region_original` slides an expanding window across the genome. At each position it accumulates weighted dosages, regresses them against the hold-out phenotype, and records the minimum p-value over window sizes in `[min_window_size, max_window_size]`. The result is a vector of $-\log_{10}(p)$ values on the grid `seq(1, N, by = skip1)`.

The $-\log_{10}(p)$ vectors from all repetitions are summed and divided by **num_rep** to form the final profile.

For **continuous traits** (`is_continuous = TRUE`), covariates are residualised from the training phenotype using only training individuals before the GWAS step, so no hold-out information leaks into the effect-size estimates. The residualisation is repeated from the original `phenotype` vector each repetition. For **binary traits** (`is_continuous = FALSE`) the raw phenotype is passed directly to the GWAS step, and the scan step fits a logistic model via `glm(family = binomial())`.

After each repetition the large intermediate matrices (`coef.mat`, `pgs.mat`) are removed from the R session and `release_memory` is called to return free heap pages to the OS. On Linux/glibc this calls `malloc_trim(0)` and can substantially reduce RSS between repetitions.

Value

Invisibly returns a named list with two elements:

`x` Integer vector of length $\lceil N/\texttt{skip1} \rceil$. The SNP position index grid `seq(1, N, by = skip1)`.

`y` Numeric vector, same length as `x`. Averaged $-\log_{10}(p)$-value profile across all **num_rep** repetitions.

The following `.rds` files are written to `save_dir`:

`chr-<chrom>_coef_mat-<rep>.rds` The $N \times 4$ GWAS coefficient matrix from repetition `rep`.

`mean_p_values_chr<chrom>_reps1-<num_rep>.rds` The averaged $-\log_{10}(p)$ vector `y`.

See Also

`ras_original` for a single-call wrapper that also runs changepoint detection and plotting. `compute_gwas_weights_origina` `compute_pgs_matrix`, `screen_forward_max_region_original` for the constituent steps. `ras_detect_original`, `ras_validate` for downstream changepoint analysis. `ras_memory` to check memory requirements before loading large genotype data. `release_memory` for OS-level heap reclamation.

Examples

```
set.seed(42)
n_samp <- 80; n_snp <- 60
geno <- matrix(
  sample(0:2, n_samp * n_snp, replace = TRUE, prob = c(0.6, 0.3, 0.1)),
  nrow = n_samp, ncol = n_snp)
pheno <- rnorm(n_samp)
cov_df <- data.frame(age = rnorm(n_samp), sex = rbinom(n_samp, 1, 0.5))
cov_df$age_squared <- cov_df$age^2
cov_df$age_sex     <- cov_df$age * cov_df$sex
for (i in 1:10) cov_df[[paste0("pc", i)]] <- rnorm(n_samp)

scan <- ras_scan_original(
  geno            = geno,
  phenotype       = pheno,
  covariates      = cov_df,
  covariate_cols  = c("age", "sex", "age_squared", "age_sex",
                      paste0("pc", 1:10)),
  is_continuous   = TRUE,
  num_rep         = 2,
  skip1           = 5,
  skip2           = 5,
  min_window_size = 2,
  max_window_size = 10,
  chrom           = 1,
  save_dir        = tempdir()
)
plot(scan$x, scan$y, type = "l",
     xlab = "SNP index", ylab = expression(-log[10](p)),
     main = "RAS scan profile")
```

`ras_sumstats_report` *Inspect an External-Summary Alignment Without Committing to It*

Description

A non-fatal report on how an external summary-statistics file lines up with a local genotype map: which ID convention each side uses, the suggested route, how many variants overlap by ID, what the build gate would say, and how the overlap is distributed along the chromosome. Run this before `ras_harmonize_sumstats` to see what the gates will do without triggering them.

Usage

```
ras_sumstats_report(ss, map, cols = list(), block_size = 500L)
```

Arguments

ss	Data frame of summary statistics, or a path to one.
map	The local genotype map (SNP, and CHR/BP for the build and coverage lines).
cols	Optional column-name overrides.
block_size	Integer. Window, in variants, for the coverage profile.

Details

The coverage line matters more here than in a PRS setting: RAS's statistic is regional, so weights that are missing in one stretch of the chromosome remove the scan's ability to detect there, rather than merely weakening it overall.

Value

Invisibly, a list with plan, n_id, build and blocks. Called for the report it prints.

See Also

ras_harmonize_sumstats

---

ras_validate *Second-Pass Changepoint Validation*

Description

Validates candidate changepoints from ras_detect_original by re-running local Davies tests in windows around each candidate.

Usage

```
ras_validate(
  this.result,
  x,
  y,
  this.start = 1,
  this.skip = 30,
  second_window_size = 50,
  p.value.threshold = 1e-10,
  min_signal = 2.5
)
```

Arguments

this.result	List. Output from ras_detect_original.
x	Numeric vector. Predictor sequence used in the original scan.
y	Numeric vector. Response sequence used in the original scan.
this.start	Integer. Genomic start position for index re-mapping to chromosome coordinates. Default 1.
this.skip	Integer. Step size used in the original scan (skip1), needed for index re-mapping. Default 30.
second_window_size	Integer. Half-window size for local Davies test re-validation. Default 50.
p.value.threshold	Numeric. Davies test p-value threshold for second-pass acceptance. A candidate is retained if *either* the right-side or left-side local Davies test passes. Default 1e-10.
min_signal	Numeric. Minimum $-\log_{10}(p)$ scan value required to retain a validated changepoint. Candidates with y[tau_hat] <= min_signal are removed after the Davies filter. Default 2.5.

Details

For each candidate $\hat{\tau}$ in this.result$tau_hats the function fits two local linear models and applies the Davies test to each: one on the window $[\hat{\tau},\ \min(\hat{\tau}+\texttt{second_window_size},\ n)]$ and one on $[\max(\hat{\tau}-\texttt{second_window_size},\ 1),\ \hat{\tau}]$. If either Davies p-value is below **p.value.threshold**, the candidate is accepted; otherwise its all.p.values entry is set to zero to suppress it in downstream plots.

Windows with fewer than four observations are not tested (Davies test requires at least four points) and their p-value is set to 1.0.

After filtering, any remaining candidate with y[tau_hat] <= min_signal is removed, as such positions are below the minimum signal threshold.

Accepted changepoint indices and all candidate indices are re-mapped from the scan grid to genomic coordinates via this.start + (index − 1) × this.skip.

Value

A named list with five elements:

tau_hats Integer vector. Validated changepoint positions, re-mapped to genomic coordinates.

all.changepoints Integer vector. All candidate positions examined, re-mapped to genomic coordinates.

all.p.values Numeric vector. $-\log_{10}(p)$ Davies values for each candidate; set to 0 for rejected candidates.

left.slopes Numeric vector. Left-side slopes at accepted changepoints (carried over from first-pass output).

right.slopes Numeric vector. Right-side slopes at accepted changepoints.

See Also

ras_detect_original for the first-pass detection step whose output this function takes as input. plot.ras for visualising the validated changepoints. ras_original for the recommended end-to-end entry point.

Examples

```
set.seed(42)
x <- 1:100
y <- c(seq(0, 8, length.out = 50),
       seq(8, 1, length.out = 50)) + rnorm(100, sd = 0.5)

cp_result <- ras_detect_original(
  x, y,
  window_size             = 50,
  skip                    = 2,
  slope_check_window_size = 10,
  slope.p.values.threshold.left  = 1e-3,
  slope.p.values.threshold.right = 1e-3
)

final <- ras_validate(
  cp_result, x = x, y = y,
  this.skip          = 1,
  second_window_size = 20,
  p.value.threshold  = 1e-3
)
cat("Validated changepoints:", final$tau_hats, "\n")
```

---

ras_weights_from_sumstats

Align External GWAS Summary Statistics to a Local Genotype Map

---

Description

Turns an external GWAS's effect estimates into a weight vector aligned to a local genotype matrix, one weight per genotype column, in order. This is the matcher underneath ras_harmonize_sumstats; call that instead unless you have already settled the naming and build questions yourself.

Usage

```
ras_weights_from_sumstats(
  sumstats,
  map,
  cols = list(),
  drop_ambiguous = TRUE,
  on_or = TRUE,
```

```
  match_by = c("id", "pos"),
  match_min_prop = 0.2,
  build_check = TRUE,
  target_af = NULL,
  af_tol = 0.2,
  block_size = 500L,
  pos_fallback = NULL
)
```

Arguments

sumstats	Path to a whitespace/tab-delimited file, or a data frame. Column names are auto-detected across PLINK 1/2 and GWAS-SSF conventions; GWAS Catalog hm_* columns take precedence, being the authoritative harmonised values. A PLINK TEST column is filtered to ADD.
map	Data frame with columns SNP, A1, A2 (and CHR, BP when position matching or the build gate is wanted) describing the LOCAL genotype matrix, one row per column of the genotype data, in order. A1 must be the allele the local dosage COUNTS.
cols	Optional named list overriding auto-detected columns, e.g. list(snp = "MarkerName", a1 = "Allele1", beta = "Effect").
drop_ambiguous	Logical. Drop strand-ambiguous A/T and C/G variants (default TRUE).
on_or	Logical. If the file carries OR instead of BETA, take log(OR) (default TRUE).
match_by	"id", "pos", or c("id","pos") (the default) to try IDs first and retry the leftovers on chr:pos. Position matching REQUIRES the build gate to pass, or to be switched off deliberately. The two keys are not interchangeable and neither is redundant. An rsID survives a change of genome build but is NOT unique – multi-allelic sites share one, and a real WGS file has about 1.25\ chr:pos:alleles is far sharper but is meaningless across builds, and indel anchoring conventions make it fragile. Not every variant carries an rsID at all, which is the case position matching exists for. They also stand in a specific order of dependence: the build gate can only compare positions on variants BOTH sides name, so the ID-matched subset is what licenses the position matching that then covers the variants without rsIDs. Using "pos" alone leaves nothing able to show it matched the right variants.
match_min_prop	Numeric. Stop if fewer than this proportion of the smaller dataset ends up with a usable weight (default 0.2, the guard bigsnpr::snp_match() uses). A low match rate is the standard symptom of a wrong build or an ID-convention mismatch.
build_check	Logical. Verify that ID-matched variants agree on position (default TRUE). Set FALSE only when the builds have been confirmed by other means; doing so while match_by includes "pos" is the one configuration that can silently produce spatially truncated weights.
target_af	Optional numeric vector, length nrow(map), giving the frequency of map$A1 in the TARGET cohort. When supplied and the summary statistics carry an effect-allele frequency column, the two are compared after orientation and reported.

	Purely diagnostic: no variant is ever dropped on a frequency disagreement, because a real frequency difference between cohorts is expected and is not an error.
af_tol	Numeric. Absolute frequency difference above which a variant is counted as discordant in that report (default 0.20).
block_size	Integer. Window, in variants along the local map, for the match-rate profile returned in blocks.
pos_fallback	Deprecated, kept so old calls still run. TRUE maps to match_by = c("id","pos"), FALSE to match_by = "id".

Details

The alignment rule is deliberately conservative:

- $w_j = +\beta_j$ when the summary statistics' effect allele is the allele the local dosage counts (map$A1);
- $w_j = -\beta_j$ when it is the other allele (map$A2);
- $w_j = 0$ when the variant is absent, the alleles cannot be reconciled, or (with drop_ambiguous = TRUE) the variant is strand-ambiguous.

Strand-ambiguous (A/T, C/G) variants are removed rather than rescued. Rescuing them requires a frequency rule, which is a modelling decision this layer does not make.

Value

A list with

weights	Numeric vector of length nrow(map): the aligned weights, zero wherever no usable weight exists.
qc	The reproducible harmonisation QC table (see ras_harmonize_sumstats).
report	Per-status variant counts.
detail	One row per map variant: status, matching route, sign, weight.
matched_by	Counts matched by ID, by ID+alleles, and by position.
build	The build gate's verdict.
blocks	Match rate along the chromosome, in block_size windows.
af	The allele-frequency comparison, when target_af was given.

See Also

ras_harmonize_sumstats for the driver that chooses the naming/build route first; ras_scan_external to scan with the resulting weights.

release_memory *Release Memory Back to the OS*

Description

Runs full garbage collection and, on Linux with the GNU C library (glibc), calls `malloc_trim(0)` to return free heap pages to the operating system. Useful after large temporary matrices are removed in memory-heavy RAS pipeline stages.

Usage

```
release_memory(verbose = TRUE)
```

Arguments

`verbose`	Logical. If `TRUE` (default), prints a one-line message with the `malloc_trim` return value so RSS changes can be monitored in pipeline logs.

Details

`malloc_trim()` is a glibc extension. On every other platform, including Linux systems with another C library such as musl (Alpine Linux), Windows and macOS, the call is not compiled in and `NA` is returned silently; no error is raised.

Value

Invisibly returns the `malloc_trim(0)` result: `1` if heap pages were returned to the OS, `0` if nothing was returned, `NA_integer_` where `malloc_trim()` is unavailable (all platforms other than glibc Linux).

screen_forward_max_region *Forward Scan of the RAS Profile*

Description

The Stage-1 forward scan on the hold-out half of the sample. At every grid position (every `skip1` SNPs) windows of `min_window_size`, `min_window_size + skip2`, ... up to `max_window_size` SNPs are formed, each window's weighted dosage score is tested against the phenotype, and the smallest p-value is recorded as the profile value. SNP columns are streamed from the `.rasbin` file and the weighted scores are accumulated on the fly, so neither the genotype matrix nor a score matrix is held in memory. The pure-R, in-memory implementation is `screen_forward_max_region_original` together with `compute_pgs_matrix`.

Usage

```
screen_forward_max_region(
  rasbin_path,
  weights,
  this.leftout,
  this.df,
  is_continuous,
  covariate_formula,
  skip1 = 10,
  skip2 = 20,
  min_window_size = 5,
  max_window_size = 100,
  chunk_snps = 5000
)
```

Arguments

`rasbin_path`	Character. Path to the `.rasbin` genotype file.
`weights`	Numeric vector of length N (one weight per variant), typically the `Estimate` column of `compute_gwas_weights` or aligned external weights from `ras_harmonize_sumstats`.
`this.leftout`	Integer vector. 1-based row indices of the samples to scan (the hold-out half, or all samples for external weights).
`this.df`	Data frame with one row per element of `this.leftout`, holding the phenotype in a column named `phenotype2` and the covariates named in `covariate_formula`.
`is_continuous`	Logical. `TRUE`: exact Frisch-Waugh-Lovell regression of the window score on the phenotype after the covariates. `FALSE`: Rao score test of the window score in a logistic model with the covariates.
`covariate_formula`	Character. Right-hand side listing the covariates, for example `"age + sex"`.
`skip1`	Integer. Grid stride in SNPs. Default `10`.
`skip2`	Integer. Window growth step in SNPs. Default `20`.
`min_window_size, max_window_size`	Integer. Smallest and largest window in SNPs. Default `5` and `100`.
`chunk_snps`	Integer. SNP columns read per disk chunk. Default `5000`.

Value

Numeric vector with one $-\log_{10}(p)$ value per grid position, `ceiling(N / skip1)` values in total.

See Also

`ras_scan`, which averages this scan over repetitions; `ras_scan_external`; `screen_forward_max_region_original`.

`screen_forward_max_region_original` *Forward Scan of the RAS Profile (Pure-R, In-Memory)*

Description

This is the pure-R, in-memory implementation that RAS 1.0.x shipped under the plain name. Since 1.1.0 the plain name (`screen_forward_max_region`) is the compiled, disk-backed implementation; this function is kept for reference and gives the same results. Slides an expanding window across the genome, accumulating weighted genotype contributions and testing association with the hold-out phenotype at each position.

Usage

```
screen_forward_max_region_original(
  geno,
  pgs.mat,
  this.df,
  num_signals,
  start.point = 1,
  save.directory = tempdir(),
  this.chrome = 1,
  min_window_size = 5,
  max_window_size = 100,
  isSimulation = TRUE,
  this.repetition = 1,
  screening_round = 1,
  isPlot = FALSE,
  skip1 = 100,
  skip2 = 5,
  is_continuous,
  covariate_formula = NULL,
  scan_test = c("glm", "score"),
  signal.starts = NULL,
  signal.window.size = NULL
)
```

Arguments

`geno`	Matrix of genotype dosages (n samples $\times$ N variants).
`pgs.mat`	Matrix of pre-computed per-variant PGS contributions ($n_{\text{holdout}} \times N$), as returned by `compute_pgs_matrix`.
`this.df`	Data frame containing the hold-out phenotype in a column named `phenotype2` and any covariate columns named in `covariate_formula`. Rows must correspond to hold-out individuals.

num_signals Integer. Number of true signals; used only for simulation plots (isSimulation = TRUE) to draw reference lines. Pass -1 for real data.

start.point Integer. Starting SNP index for the scan. Default 1.

save.directory Character. Directory for optional PDF plots (only used when isPlot = TRUE). Defaults to tempdir() so nothing is written outside the session's temporary area unless the caller chooses an explicit path.

this.chrome Integer. Chromosome number for non-simulation plot filenames. Default 1.

min_window_size
Integer. Minimum scan window half-size. Default 5.

max_window_size
Integer. Maximum scan window half-size. Default 100.

isSimulation Logical. If TRUE, use simulation-mode plot filenames and draw true-signal reference lines. Default TRUE.

this.repetition
Integer. Repetition index used in plot filenames. Default 1.

screening_round
Integer. Number of forward screening rounds. Only a single round is implemented; values other than 1 raise an error. Default 1.

isPlot Logical. If TRUE, save a PDF of the scan profile to save.directory. Default FALSE (no file is written).

skip1 Integer. Step size for the primary SNP position grid. Default 100.

skip2 Integer. Step size for growing the window within each grid position. Default 5.

is_continuous Logical. If TRUE, the per-window PGS p-value is obtained from a linear model via an exact Frisch-Waugh-Lovell residualisation (numerically identical to lm(phenotype2 ~ this.pgs + covariates) but the covariate factorisation is done once instead of per window). If FALSE, a binary trait is used (see scan_test).

covariate_formula
Character. Right-hand side covariates (without the PGS term this.pgs) in the scan regression formula. If NULL, a default formula with sex, age, age-squared, age-sex interaction, and the first ten ancestry PCs is used.

scan_test Character. Test used for the binary branch (is_continuous = FALSE). "glm" (default) fits a full logistic regression per window and reports the Wald p-value of the PGS term, preserving the original behaviour. "score" fits the covariate-only null logistic model **once** and evaluates a Rao score test for the PGS term at each window, which is asymptotically equivalent but avoids per-window iterative fitting (typically 1-2 orders of magnitude faster at scale). Ignored when is_continuous = TRUE (the continuous branch always uses lm).

signal.starts Integer vector. True signal start positions for simulation reference lines. Default NULL.

signal.window.size
Integer. Width of each true signal window for simulation reference lines. Default NULL.

Details

At each grid position $j \in \{1, 1 + \texttt{skip1}, \ldots, N\}$ the function builds a polygenic score (PGS) by accumulating columns of `pgs.mat` from a growing symmetric window $[j - w + 1,\ j + w - 1]$ for $w \in \{0, \texttt{min_window_size}, \ldots, \texttt{max_window_size}\}$. The accumulation is incremental: only the newly added columns are summed at each step, avoiding redundant computation.

At window size $w = 0$, only column j of `pgs.mat` is used (the single-SNP PGS). For each window size the PGS is tested against `phenotype2` via `lm` (continuous) or `glm(family = binomial())` (binary). The minimum p-value over all window sizes is stored for position j.

The final return value is $-\log_{10}$ of these per-position minimum p-values.

Value

Numeric vector of length $\lceil N/\texttt{skip1} \rceil$. Each element is the $-\log_{10}(p)$-value at the corresponding grid position, where the p-value is the minimum over all tested window sizes.

See Also

`compute_pgs_matrix` for the step that produces `pgs.mat`. `ras_scan_original` for the recommended high-level entry point that calls this function automatically for each repetition.

Examples

```
set.seed(3)
n_samp <- 80; n_snp <- 50
geno <- matrix(sample(0:2, n_samp * n_snp, replace = TRUE,
                      prob = c(0.6, 0.3, 0.1)),
               nrow = n_samp, ncol = n_snp)
weights  <- rnorm(n_snp)
leftout  <- 31:80
pgs_mat  <- compute_pgs_matrix(geno, leftout, weights)

scan_df           <- data.frame(matrix(rnorm(50 * 12), 50, 12))
colnames(scan_df) <- c("age", "sex", "age_squared", "age_sex",
                       paste0("pc", 1:8))
scan_df$phenotype2 <- rnorm(50)

p_vec <- screen_forward_max_region_original(
  geno              = geno,
  pgs.mat           = pgs_mat,
  this.df           = scan_df,
  num_signals       = -1,
  is_continuous     = TRUE,
  covariate_formula = paste(c("age", "sex"), collapse = " + "),
  skip1             = 5,
  skip2             = 5,
  min_window_size   = 2,
  max_window_size   = 8,
  isPlot            = FALSE
)
plot(seq(1, n_snp, by = 5), p_vec, type = "l",
```

```
        xlab = "SNP index", ylab = expression(-log[10](p)))
```

`slope_test` *One-Tailed Slope Test Through the Origin*

Description

Fits a no-intercept linear model `y ~ x - 1` and performs a one-tailed t-test on the slope coefficient.

Usage

```
slope_test(x, y, lower.tail)
```

Arguments

`x`	Numeric vector. Predictor values, typically centred at the candidate changepoint so that the origin corresponds to the changepoint position.
`y`	Numeric vector. Response values, same length as `x`, typically centred at the changepoint y-value.
`lower.tail`	Logical. Passed to `pt`. Use `FALSE` to test $H_0 : \beta \geq 0$ against $H_1 : \beta < 0$ (left-slope test); use `TRUE` for the opposite direction (right-slope test).

Details

The model `lm(y ~ x - 1)` forces the regression line through the origin, which is appropriate after centring both `x` and `y` at the candidate changepoint. Under this parameterisation a positive left slope and a negative right slope correspond to the peak shape expected at a true association region.

The function is called by `ras_detect_original` on both sides of each Davies-significant candidate, using asymmetric thresholds (`cp_slope_left = 1e-10`, `cp_slope_right = 1e-20` by default) to require a steeper descending edge than ascending edge.

Value

Numeric scalar. The one-tailed p-value for the slope coefficient.

See Also

`get_break_points` for the Davies test step that precedes this slope check. `ras_detect_original` for the full detection workflow.

Examples

```
# Left-side slope: x goes from negative to 0, y should be rising (positive slope)
x_left <- -10:0
y_left <- x_left * 0.8 + rnorm(11, sd = 0.3)
slope_test(x_left, y_left, lower.tail = FALSE)  # expect small p (slope > 0)

# Right-side slope: x goes from 0 to positive, y should be falling (negative slope)
x_right <- 0:10
y_right <- x_right * (-0.8) + rnorm(11, sd = 0.3)
slope_test(x_right, y_right, lower.tail = TRUE)  # expect small p (slope < 0)
```

Index

bed_to_rasbin, *3*, *5*, 5, *17*, *20*, *44, 45*

compute_gwas_weights, *4*, 6, *7*, *21*, *43*, *57*
compute_gwas_weights_fast *(*ras-aliases*)*, 21
compute_gwas_weights_original, *4–7*, 7, *10*, *36*, *49*
compute_pgs_matrix, *4, 5*, *9*, 9, *36*, *49*, *56*, *58*, *60*

davies.test, *12*, *29*

geno_to_rasbin, *3*, *5, 6*, 11, *17*, *20*, *44, 45*
get_break_points, *4, 5*, 12, *30, 31*, *61*
get_local_maximum, *4, 5*, 13, *31*

plot.ras, *3*, *5*, 14, *16*, *20*, *26*, *41*, *53*
plot_ras_scan, *40*
plot_ras_zoom_regions, *40*
print.ras, *5*, *15*, 16, *26*
pt, *61*

RAS, *40*
RAS *(*RAS-package*)*, 3
ras, *3*, *5*, *14–16*, 16, *21*, *24*, *26*, *29*, *37*, *44, 45*
ras-aliases, 21
RAS-package, 3
ras_box_calibrate, *5*, *19*, 23, *25–27*, *39, 40*, *46*
ras_box_detect, *3–5*, *16*, *18*, *20*, *23, 24*, 24, *26, 27*, *29*, *39, 40*, *45, 46*
ras_box_stat, *4, 5*, *19*, *25, 26*, 26, *40*
ras_detect, *3–5*, *13*, *16*, *18*, *20, 21*, *24*, *26*, 28, *29*, *45, 46*
ras_detect_fast *(*ras-aliases*)*, 21
ras_detect_original, *4, 5*, *12, 13*, *29*, 29, *39*, *41*, *49*, *51–53*, *61*
ras_fast *(*ras-aliases*)*, 21
ras_harmonize_sumstats, *4, 5*, 32, *35*, *42, 43*, *46, 47*, *50, 51*, *53*, *55*, *57*
ras_liftover_sumstats, 34
ras_map_ids_to_rsid, 35, *42, 43*
ras_memory, *5*, *10*, 36, *41*, *49*
ras_original, *3*, *5*, *14*, *19, 20*, *31*, *37*, 37, *49*, *53*
ras_read_variant_dictionary, *32*, *34, 35*, 42
ras_scan, *3–5*, *7*, *16*, *20, 21*, *23*, *27*, 43, *46, 47*, *57*
ras_scan_external, *4, 5*, *21*, *23*, *27*, *33*, *45*, 45, *46, 47*, *55*, *57*
ras_scan_external_fast *(*ras-aliases*)*, 21
ras_scan_external_original, *45*, 46
ras_scan_fast *(*ras-aliases*)*, 21
ras_scan_original, *4, 5*, *8, 9*, *18*, *30*, *37*, *41*, *43–45*, *47*, 47, *60*
ras_sumstats_report, *5*, *33*, 50
ras_validate, *3–5*, *15, 16*, *18*, *20*, *24*, *26*, *28, 29*, *31*, *39–41*, *49*, 51
ras_weights_from_sumstats, *5*, *33*, 53
rasbin_header, *5*, 22
rasbin_read_chunk, *5, 6*, 22
release_memory, *5*, *37*, *49*, 56

screen_forward_max_region, *4*, *7*, *21*, *43*, 56, *58*
screen_forward_max_region_fast *(*ras-aliases*)*, 21
screen_forward_max_region_original, *4, 5*, *10*, *36*, *39*, *48, 49*, *56, 57*, 58
segmented, *12*, *29*
slope_test, *4, 5*, *12*, *30, 31*, 61
stop, *36*

tempdir, *39*, *48*, *59*

which.max, *13*