Package 'RAS'

July 24, 2026

Title Regional Association Score for Genome-Wide Association Studies

Version 1.0.3

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 applies changepoint detection to locate association regions, improving statistical power while controlling the false positive rate. 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

RoxygenNote 7.3.3

Imports grDevices, graphics, parallel, segmented, stats

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

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

Maintainer Jiahe Jin <jiahe.jin@yale.edu>

Repository CRAN

Date/Publication 2026-07-24 09:30:24 UTC

Contents

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 $-\log_{10}(p)$ time series and applies changepoint detection to locate peaks that mark significant association regions. The method supports both continuous and binary traits.

Option A — one call (recommended)

`ras` runs the complete pipeline and returns a `"ras"` object; 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/")

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

Option B — step-by-step (advanced)

`ras_scan` → `ras_detect` → `ras_validate` → `plot.ras`

```
## Step 1: compute 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: first-pass changepoint detection
detected <- ras_detect(scan$x, scan$y,
                       window_size = 3000,
                       slope.p.values.threshold.left  = 1e-10,
                       slope.p.values.threshold.right = 1e-20)

## Step 3: second-pass validation
final    <- ras_validate(detected, x = scan$x, y = scan$y,
                         this.skip = 10, p.value.threshold = 1e-10)

## 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`), builds a weighted dosage matrix for the hold-out half (`compute_pgs_matrix`), and runs a forward scan (`screen_forward_max_region`) that slides an expanding window and records the minimum p-value at each grid position. 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, logistic regression is used in the scan step.

Stage 2 — First-pass changepoint detection

`ras_detect` 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`.

Stage 3 — Second-pass validation

`ras_validate` re-tests each first-pass 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**. Candidates with $-\log_{10}(p) \leq$ **min_signal** at the estimated position are discarded regardless.

Stage 4 — Plots

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

Author(s)

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

See Also

- compute_gwas_weights
- compute_pgs_matrix
- get_break_points
- get_local_maximum
- plot.ras
- print.ras
- ras
- ras_detect
- ras_memory
- ras_scan
- ras_validate
- release_memory
- screen_forward_max_region
- slope_test

---

compute_gwas_weights *Compute Per-SNP GWAS Effect Size Weights*

Description

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(
  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).

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 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(
  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`.

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` for the step that produces `pgs.weights`. `screen_forward_max_region` 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])
```

---

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` 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 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. 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.

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

Detect significant association regions in GWAS data. Returns an object of class "ras".

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("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,
  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`. Default `5`.

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

`scan_test` Character. Per-window test for binary traits, passed to `screen_forward_max_region`. `"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`.

`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`.

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` A 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.

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(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`, `ras_detect`, `ras_validate`.

See Also

`ras_scan` for the scan-only step (useful when tuning changepoint parameters separately). `ras_detect`, `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(
  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(),
  run_plots         = TRUE,
  plot_device       = "png"
)

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")
```

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

Description

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(
  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`). 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 for the recommended end-to-end entry point.

Examples

```
set.seed(42)
x <- 1:300
y <- c(seq(0, 8, length.out = 150),
       seq(8, 1, length.out = 150)) + rnorm(300, sd = 0.5)
result <- ras_detect(
  x, y,
  window_size             = 150,
  slope_check_window_size = 20,
  slope.p.values.threshold.left  = 1e-3,
  slope.p.values.threshold.right = 1e-3
)
cat("Detected changepoints:", result$tau_hats, "\n")
```

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 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 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`, `ras` 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_scan *Run the RAS Scan (Repetition Screening Loop)*

Description

Computes the averaged $-\log_{10}(p)$-value profile over **num_rep** independent 50/50 train/hold-out splits.

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("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`. Default `5`.
max_window_size	Integer. Maximum scan window half-size passed to `screen_forward_max_region`. Default `100`.
scan_test	Character. Per-window test for binary traits, passed to `screen_forward_max_region`. `"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` 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` 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` for a single-call wrapper that also runs changepoint detection and plotting. `compute_gwas_weights`, `compute_pgs_matrix`, `screen_forward_max_region` for the constituent steps. `ras_detect`, `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(
  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_validate` *Second-Pass Changepoint Validation*

Description

Validates candidate changepoints from `ras_detect` 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`.

`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 $[\hat{\tau},\ \min(\hat{\tau}+$`second_window_size`$, n)]$ and one on $[\max(\hat{\tau} -$ `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) $\times$ `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` for the first-pass detection step whose output this function takes as input. `plot.ras` for visualising the validated changepoints. `ras` for the recommended end-to-end entry point.

Examples

```
set.seed(42)
x <- 1:300
y <- c(seq(0, 8, length.out = 150),
       seq(8, 1, length.out = 150)) + rnorm(300, sd = 0.5)

cp_result <- ras_detect(
  x, y,
  window_size             = 150,
  slope_check_window_size = 20,
  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 = 30,
  p.value.threshold  = 1e-3
)
cat("Validated changepoints:", final$tau_hats, "\n")
```

---

release_memory *Release Memory Back to the OS*

Description

Runs full garbage collection and, on Linux/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

On non-Linux platforms the C call is skipped 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_` on non-Linux platforms.

screen_forward_max_region
Forward Scan to Compute the RAS Profile

Description

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(
  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` 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(
  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` 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` 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