

Package: lantern (via r-universe)

October 7, 2026

Type Package

Title Ancestry-Specific Rare Variant Association Analysis

Version 0.99.0

Description Ancestry-specific rare variant burden testing for admixed populations. Splits genotype dosages by local ancestry (as inferred by 'RFMix') into per-ancestry dosages, using either phased haplotypes or unphased genotypes with shrinkage toward global ancestry, and writes them to 'SeqArray' GDS files. Ancestry-stratified gene-level tests are then run with the 'SMMAT' framework of Chen et al. (2019) [doi:10.1016/j.ajhg.2018.12.012](https://doi.org/10.1016/j.ajhg.2018.12.012) as implemented in 'GMMAT', and the per-ancestry p-values are combined with the Cauchy combination test of Liu and Xie (2020) [doi:10.1080/01621459.2018.1554485](https://doi.org/10.1080/01621459.2018.1554485).

URL <https://yuw444.github.io/lantern/>,
<https://github.com/yuw444/lantern>

BugReports <https://github.com/yuw444/lantern/issues>

SystemRequirements bcftools (>= 1.10) (optional; faster VCF reading and BCF input)

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.5.0)

Imports data.table, SeqArray, stats, utils

Suggests BiocStyle, GMMAT, knitr, rmarkdown, SeqVarTools, testthat (>= 3.0.0)

VignetteBuilder knitr

biocViews Software, Genetics, GenomeWideAssociation, SNP, VariantAnnotation

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs zlib1g-dev
Repository <https://biocstaging.r-universe.dev>
Date/Publication 2026-10-06 20:24:48 UTC
RemoteUrl <https://github.com/BiocStaging/lantern>
RemoteRef HEAD
RemoteSha fd0703b3db6b95aa7f1c2252b44981d70f423f2f

Contents

ancestry_smmat	2
ancestry_split	5
ancestry_split_dosage	6
ancestry_split_phased	8
cauchy_combine	9
read_bed_file	10
split_diploid	11
split_diploid_multi	12
split_haplotype	14
split_haplotype_multi	15
write_ancestry_gds	16
write_dosage_gds	17

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

ancestry_smmat	<i>Run ancestry-stratified SMMAT + Cauchy combination (Step 3)</i>
----------------	--

Description

Wraps GMMAT : : SMMAT() across all ancestry-split GDS files (as produced by [write_ancestry_gds](#)), computes per-gene ancestry weights from Step 1’s ancestry_counts (this replaces the retired src/step3_weight_finding.R script), and combines the per-gene p-values across populations with [cauchy_combine](#). This is the package’s Step 3 entry point; it does not run Step 1/2 for you (SMMAT runs can take hours, so this is always an explicit, separate call).

Usage

```
ancestry_smmat(  
  gds_paths,  
  pheno,  
  formula,  
  kinship,  
  gene_group_file,  
  ancestry_counts = NULL,  
  variant_info = NULL,  
  id_col = "id",
```

```

family = stats::gaussian(),
gene_col = "group",
p_col = "E.pval",
MAF.range = c(0, 0.5),
miss.cutoff = 1,
method = "davies",
ncores = 1,
verbose = TRUE,
...
)

```

Arguments

<code>gds_paths</code>	Named list/vector of GDS file paths, one per ancestry population, e.g. <code>c(AFR = "afr.gds", EUR = "eur.gds")</code> – as returned by write_ancestry_gds . Any $K \geq 1$ populations are supported; add an extra named entry (e.g. <code>OBSERVED = ...</code>) to include an additional (e.g. non-ancestry-split) GDS in the same SMMAT + Cauchy combination run.
<code>pheno</code>	<code>data.frame</code> with an <code>id_col</code> column plus the response and covariates referenced in formula.
<code>formula</code>	Model formula for the null model, e.g. <code>y ~ age + sex</code> .
<code>kinship</code>	Numeric matrix (or object coercible via <code>as.matrix</code>) with row/column names equal to sample IDs.
<code>gene_group_file</code>	Either a path to the <code>gene_group</code> file passed to <code>GMMAT::SMMAT()</code> (no header; columns <code>gene</code> , <code>chr</code> , <code>pos</code> , <code>ref</code> , <code>alt</code> , <code>weight</code>), or a <code>data.frame</code> /matrix with those same six columns. Either way, it is read, has any leading "chr" stripped from the <code>chr</code> column (matching <code>SeqArray::seqVCF2GDS()</code> 's convention – see the "chr column" note below), and written to a fresh temporary file that is deleted with <code>unlink()</code> when <code>ancestry_smmat()</code> returns; the caller's own path/ <code>data.frame</code> is never read-modified or deleted, and can safely be reused across repeated calls (e.g. one call per phenotype). chr column: pass either "19" or "chr19" – normalized automatically. Without this, a mismatch between the group file's chr convention and the GDS's (always bare, no "chr" prefix) would silently drop every affected variant, and the whole gene if all its variants are affected, since <code>GMMAT::SMMAT()</code> matches variants by exact <code>chr:pos:ref:alt</code> string equality with no fuzzy/positional alignment.
<code>ancestry_counts, variant_info</code>	Optional: pass <code>ancestry_split()\$ancestry_counts/\$variant_info</code> straight through to compute per-gene ancestry weights for the Cauchy combination. If omitted, equal weights are used.
<code>id_col</code>	Name of the sample ID column in <code>pheno</code> and formula's <code>id</code> argument to <code>glmmkin</code> (default "id").
<code>family</code>	A family object for <code>GMMAT::glmmkin()</code> (default <code>gaussian()</code>).
<code>gene_col</code>	Name of the gene/group ID column in <code>SMMAT()</code> 's output (default "group", matching <code>GMMAT::SMMAT()</code>).

<code>p_col</code>	Name of the p-value column in <code>SMMAT()</code> 's output to combine (default <code>"E.pval"</code> , matching <code>GMMAT::SMMAT()</code> 's default tests = <code>"E"</code> ; use <code>"O.pval"/"S.pval"/"B.pval"</code> if you pass a different tests via ...).
<code>MAF.range, miss.cutoff, method, ncores</code>	Passed straight through to <code>GMMAT::SMMAT()</code> – same names, same meaning as in a direct <code>SMMAT()</code> call.
<code>verbose</code>	Print this function's own step-by-step progress messages (Steps 1-6 below). Distinct from <code>GMMAT::SMMAT()</code> 's own internal verbose (its per-call progress bar), which is not exposed separately and stays at <code>SMMAT</code> 's default (<code>FALSE</code>); every other <code>SMMAT()</code> argument is available via
<code>...</code>	Any other <code>GMMAT::SMMAT()</code> argument, forwarded as-is – for example tests (<code>"B"</code> , <code>"S"</code> , <code>"O"</code> , or the default <code>"E"</code>), <code>rho</code> , <code>MAF.weights.beta</code> , <code>missing.method</code> , <code>use.minor.allele</code> , <code>auto.flip</code> , <code>Garbage.Collection</code> , <code>group.file.sep</code> , or <code>meta.file.prefix</code> . <code>null.obj</code> , <code>geno.file</code> , <code>group.file</code> , and <code>is.dosage</code> are always supplied internally by <code>ancestry_smmat()</code> and cannot be overridden this way. See <code>?GMMAT::SMMAT</code> for the full list and defaults.

Value

A list with two elements:

<code>results</code>	data.frame with one row per gene: <code>gene</code> , one <code>p_<POP></code> column per population, one <code>w_<POP></code> weight column per population (only if <code>ancestry_counts/variant_info</code> were supplied), and <code>p_cauchy</code> .
<code>smmat_results</code>	Named list (one entry per population, same names as <code>gds_paths</code>) of the raw <code>GMMAT::SMMAT()</code> result data.frames.

See Also

[ancestry_split](#), [write_ancestry_gds](#), [cauchy_combine](#)

Examples

```
if (requireNamespace("GMMAT", quietly = TRUE) &&
    requireNamespace("SeqVarTools", quietly = TRUE)) {
  ext <- system.file("extdata", package = "lantern")
  split <- ancestry_split(file.path(ext, "toy.vcf.gz"),
                          file.path(ext, "toy.msp.tsv"),
                          mode = "dosage", chrom = "chr19", verbose = FALSE)

  out_dir <- tempfile()
  gds <- write_ancestry_gds(split, out_dir, verbose = FALSE)

  pheno <- read.delim(file.path(ext, "toy.pheno.tsv"))
  kinship <- diag(nrow(pheno)) # unrelated samples
  dimnames(kinship) <- list(pheno$id, pheno$id)

  # gene_group_file: a path, or a data.frame with the same six columns
  out <- ancestry_smmat(gds, pheno, y ~ age + sex, kinship,
                       file.path(ext, "toy_genes.tsv"),
                       ancestry_counts = split$ancestry_counts,
```

```

        variant_info = split$variant_info,
        verbose = FALSE)
  out$results      # per-gene p-values + Cauchy combination
  head(out$smmat_results$AFR) # raw GMMAT::SMMAT() output for AFR
  unlink(out_dir, recursive = TRUE)
}

```

ancestry_split

Split a phased VCF by local ancestry (Step 1)

Description

Parse an RFMix MSP file and a phased VCF, intersect samples, map each variant to its ancestry tract, and split genotypes into per-population dosage matrices using either the proportional ("dosage") or deterministic ("haplotype") split algorithm. This is the package's Step 1 entry point: it only computes in-memory dosage matrices, with no file I/O. Pass the result to [write_ancestry_gds](#) (Step 2) to materialize GDS files, and then to [ancestry_smmat](#) (Step 3) to run SMMAT + Cauchy combination.

Usage

```

ancestry_split(
  vcf_path,
  msp_path,
  mode = c("dosage", "haplotype"),
  chrom = NULL,
  use_gla = TRUE,
  verbose = TRUE
)

```

Arguments

vcf_path	Path to phased VCF/BCF file (plain or gzipped). Read with bcftools if it is on PATH (fastest; required for BCF), otherwise with SeqArray . Set options(lantern.vcf_reader = "bcftools") or "seqarray" to force one.
msp_path	Path to RFMix MSP file (plain text or gzipped TSV).
mode	Split algorithm: "dosage" (proportional p1/p2 split via split_diploid_multi , unphased) or "haplotype" (deterministic per-haplotype split via split_haplotype_multi , phased). Either/or – call ancestry_split() twice if both are needed.
chrom	Chromosome to process (e.g., "chr19" or "19"). If NULL, all chromosomes present in the VCF are used. When supplied, vcf_path and msp_path may contain other chromosomes too: MSP tracts outside chrom are dropped up front, and bcftools query is restricted to chrom via -t when the VCF's declared or inferred contigs allow resolving the exact name to use (whichever of chrom, with, or without a leading "chr" matches) – other chromosomes' genotypes are never read into R. If the contig can't be resolved, all variants are queried and filtered in R instead (slower, more memory, same result).

use_gla	Logical, mode = "dosage" only. When TRUE (default), per-arm GLA (global local ancestry) shrinkage is computed and applied automatically (see the "Core Algorithm" section of the package's <code>CLAUDE.md</code> for the formula/rationale). When FALSE, GLA shrinkage is skipped entirely and the split reproduces the original pre-shrinkage $p[k]$ estimator exactly, including its 0.5/0.5 singleton fallback – equivalent to calling <code>split_diploid_multi</code> with <code>gla = NULL</code> . Ignored (no effect) when mode = "haplotype", which is deterministic and has no ambiguous ratio to shrink.
verbose	Print step-by-step progress messages.

Value

Invisibly, a list with one named numeric matrix (variants x samples) per population (named after the MSP population codes, e.g. \$AFR, \$EUR), plus:

variant_info	data.frame with columns chrom, pos, ref, alt.
sample_ids	Character vector of common sample IDs.
mode	The split mode used.
ancestry_counts	Numeric matrix (variants x populations): for each variant, the number of samples with pure ancestry in that population. Feeds <code>ancestry_smmat</code> 's per-gene weight calculation.
tract_info	data.frame of ancestry tracts from the MSP file.
overlap	List of intersection/filtering statistics.

See Also

[write_ancestry_gds](#), [ancestry_smmat](#)

Examples

```
vcf <- system.file("extdata", "toy.vcf.gz", package = "lantern")
msp <- system.file("extdata", "toy.msp.tsv", package = "lantern")
split <- ancestry_split(vcf, msp, mode = "dosage", chrom = "chr19",
                       verbose = FALSE)
split$AFR[1:3, 1:5] # African dosages (variants x samples)
split$EUR[1:3, 1:5] # European dosages
head(split$variant_info)
```

ancestry_split_dosage *Run ancestry splitting pipeline*

Description

Complete pipeline: split genotype matrix by ancestry, return dosage matrices. Handles sample and variant mismatches by using only overlapping data. Memory efficient: single-pass subsetting using index vectors.

Usage

```
ancestry_split_dosage(
  gt_matrix = NULL,
  pt_matrix = NULL,
  vcf_path = NULL,
  msp_path = NULL,
  chrom = NULL,
  verbose = TRUE
)
```

Arguments

<code>gt_matrix</code>	Integer matrix of genotypes (rows=variants, cols=samples). Values: 0=homozygous ref, 1=heterozygous, 2=homozygous alt. rownames: variant IDs (e.g., "chr:pos"). colnames: sample IDs. May be NULL when <code>vcf_path</code> and <code>msp_path</code> are supplied.
<code>pt_matrix</code>	Integer matrix of parent-of-origin ancestry codes (rows=samples, cols=regions/variants). Values: 1=EUR/EUR, 2=AFR/EUR (mixed), 3=AFR/AFR. rownames: sample IDs (must match colnames of <code>gt_matrix</code>). colnames: region IDs (e.g., "chr:start-end"). May be NULL when <code>vcf_path</code> and <code>msp_path</code> are supplied.
<code>vcf_path</code>	Path to a phased VCF or BCF file. When supplied together with <code>msp_path</code> , the function delegates to ancestry_split (Step 1, mode = "dosage"), bypassing the <code>gt_matrix/pt_matrix</code> arguments.
<code>msp_path</code>	Path to an RFMix MSP file (plain text or gzipped TSV). Used together with <code>vcf_path</code> as a shortcut input.
<code>chrom</code>	Chromosome name to restrict to (e.g. "chr19"). Only used when <code>vcf_path/msp_path</code> are supplied.
<code>verbose</code>	Print progress messages (default TRUE)

Value

List with elements:

<code>african</code>	African ancestry-specific dosage matrix
<code>european</code>	European ancestry-specific dosage matrix
<code>counts</code>	List of ancestry counts per region/sample
<code>overlap</code>	Overlap information

GLA shrinkage

The `vcf_path/msp_path` shortcut delegates to [ancestry_split](#), which computes and applies per-arm global local ancestry (GLA) shrinkage automatically from tract positions. The direct `gt_matrix/pt_matrix` path has no genomic position information and cannot derive GLA automatically, so it always uses the original (unshrunk) p1/p2 estimator; this path also reorders and filters variants internally (sample/region overlap, monomorphic removal), so a caller-supplied `gla/arm_id` could not be reliably kept aligned. Call [split_diploid](#) directly (no internal reordering) if you need GLA shrinkage with a pre-computed `gla/arm_id` of your own.

Examples

```
# Example with mismatched dimensions
gt <- matrix(c(2, 1, 0, 1, 2, 1), nrow = 2, ncol = 3,
             dimnames = list(c("chr1:100", "chr1:200"),
                             c("sample_A", "sample_B", "sample_C")))

pt <- matrix(c(3, 2, 1, 3, 2, 1), nrow = 3, ncol = 2,
            dimnames = list(c("sample_A", "sample_B", "sample_D"),
                            c("chr1:50-150", "chr1:150-250")))

result <- ancestry_split_dosage(gt, pt)
# Only sample_A and sample_B are kept (common to both)
# Only chr1:100 and chr1:200 that overlap regions are kept

# Shortcut: build matrices directly from VCF + MSP
vcf <- system.file("extdata", "toy.vcf.gz", package = "lantern")
msp <- system.file("extdata", "toy.msp.tsv", package = "lantern")
result <- ancestry_split_dosage(vcf_path = vcf, msp_path = msp,
                               chrom = "chr19", verbose = FALSE)
```

ancestry_split_phased *Run phased ancestry splitting pipeline*

Description

Backward-compatible 2-population (AFR/EUR) wrapper around [ancestry_split](#) (Step 1, mode = "haplotype") and, when write_vcf = TRUE, [write_ancestry_gds](#) (Step 2). For K > 2 populations or to keep the split and GDS-writing steps separate, call [ancestry_split\(\)](#) and [write_ancestry_gds\(\)](#) directly.

Usage

```
ancestry_split_phased(
  vcf_path,
  msp_path,
  out_path,
  chrom = NULL,
  write_vcf = TRUE,
  verbose = TRUE
)
```

Arguments

vcf_path	Path to phased VCF/BCF file (plain or gzipped). Read with bcftools if it is on PATH (fastest; required for BCF), otherwise with SeqArray . Set options(lantern.vcf_reader = "bcftools") or "seqarray" to force one.
msp_path	Path to RFMix MSP file (plain text or gzipped TSV).

out_path	Output directory for GDS and cache files.
chrom	Chromosome to process (e.g., "chr19" or "19"). If NULL, all chromosomes present in the VCF are used.
write_vcf	Logical; if TRUE, materialize ancestry-specific GDS files via write_ancestry_gds (Step 2).
verbose	Print step-by-step progress messages.

Value

Invisibly, a list with elements:

african	Numeric matrix (variants x samples) of African dosages.
european	Numeric matrix (variants x samples) of European dosages.
variant_info	data.frame with columns chrom, pos, ref, alt.
sample_ids	Character vector of common sample IDs.
tract_info	data.frame of ancestry tracts from the MSP file.
overlap	List of intersection statistics.
vcf_paths	(when write_vcf = TRUE) Named list with african_gds/european_gds paths.

Examples

```
vcf <- system.file("extdata", "toy.vcf.gz", package = "lantern")
msp <- system.file("extdata", "toy.msp.tsv", package = "lantern")
out_dir <- tempfile()
result <- ancestry_split_phased(vcf, msp, out_path = out_dir,
                               chrom = "chr19", write_vcf = FALSE,
                               verbose = FALSE)

head(result$variant_info)
unlink(out_dir, recursive = TRUE)
```

cauchy_combine

Cauchy combination test for multiple p-values

Description

Combines K p-values from correlated or independent tests into a single meta-analysis p-value using the Cauchy combination test (Liu & Xie 2020). Unlike Fisher's method, the test remains valid under arbitrary dependency structures among p-values.

Usage

```
cauchy_combine(p_values, weights = NULL)
```

Arguments

- p_values** Numeric vector of p-values ($K \geq 2$). Values must be in $(0, 1]$. Exact zeros are clipped to `.Machine$double.eps`.
- weights** Optional non-negative numeric weight vector of the same length as **p_values**. Defaults to equal weights.

Value

A single numeric p-value in $(0, 1]$.

References

Liu, Y. and Xie, J. (2020). Cauchy Combination Test: A Powerful Test With Analytic p-Value Calculation Under Arbitrary Dependency Structures. *J. Am. Stat. Assoc.* **115**(529), 393–402. [doi:10.1080/01621459.2018.1554485](https://doi.org/10.1080/01621459.2018.1554485)

See Also

[write_dosage_gds](#), [ancestry_split](#), [ancestry_smmat](#)

Examples

```
# Combine African-ancestry and European-ancestry p-values for one gene
cauchy_combine(c(0.01, 0.04))

# Three-population meta-analysis
cauchy_combine(c(0.01, 0.04, 0.20))

# Weighted combination (double weight on AFR)
cauchy_combine(c(0.01, 0.04), weights = c(2, 1))

# Apply across a results table (one gene per row)
# mapply(cauchy_combine, split(cbind(p_aa, p_ee), seq_len(nrow(res))))
```

read_bed_file

Read a PLINK .bed file of BED-encoded local ancestry calls

Description

Reads a PLINK binary genotype file (`.bed/.bim/.fam` trio) into an integer matrix, for the case where local ancestry has been encoded as PLINK genotype calls at ancestry-tract "SNPs" (as consumed by `src/step1_vcf_split_by_ancestry.R`): homozygous first allele = pure EUR/EUR (code 1), heterozygous = mixed AFR/EUR (code 2), homozygous second allele = pure AFR/AFR (code 3), missing = no call (code 0). This matches `snpStats::read.plink()`'s raw numeric convention (values feed directly into [split_diploid/ancestry_split_dosage](#) as `pt_matrix/ancestry` without `snpStats` as a dependency).

Usage

```
read_bed_file(bed, bim, fam)
```

Arguments

bed	Path to the .bed file.
bim	Path to the .bim file (variant IDs are taken from column 2).
fam	Path to the .fam file (sample IDs are taken from column 2, i.e. IID).

Value

Integer matrix (variants x samples), dimnamed from the .bim/.fam files. Values: 0 = no call, 1 = homozygous first allele, 2 = heterozygous, 3 = homozygous second allele.

See Also

[split_diploid](#), [ancestry_split_dosage](#)

Examples

```
ext <- system.file("extdata", package = "lantern")
pt <- read_bed_file(file.path(ext, "toy_ancestry.bed"),
                    file.path(ext, "toy_ancestry.bim"),
                    file.path(ext, "toy_ancestry.fam"))
pt[, 1:6] # ancestry tracts x samples, codes 1/2/3
```

split_diploid	<i>Split genotype matrix by ancestry</i>
---------------	--

Description

Split a genotype matrix into African and European ancestry-specific dosage matrices based on a parent-of-origin ancestry matrix.

Usage

```
split_diploid(gt_genotype, ancestry, gla = NULL, arm_id = NULL)
```

Arguments

gt_genotype	Integer matrix of genotypes (0, 1, 2) - Rows represent variants - Columns represent samples
ancestry	Integer matrix of ancestry codes (1, 2, 3) - Same dimensions as gt_genotype - Values: 1=EUR/EUR, 2=AFR/EUR (mixed), 3=AFR/AFR

gla	Optional 2 x K numeric matrix (rows "p"/"q", columns population names) of per-arm global local ancestry proportions, as produced by <code>ancestry_split</code> internally. When NULL (default), GLA shrinkage is disabled and behavior matches the original p1/p2 estimator exactly, including its 0.5/0.5 singleton fallback.
arm_id	Optional integer vector (length = nrow(gt_genotype)) giving each variant's row index into gla (0-based). Required together with gla.

Value

List with two elements:

african	Matrix of African ancestry-specific dosages
european	Matrix of European ancestry-specific dosages

Examples

```
# Genotype matrix: 5 variants x 4 samples
gt <- matrix(c(2, 1, 0, 1, 2, 1, 0, 2, 1, 0,
               1, 1, 1, 0, 2, 1, 0, 1, 0, 1), nrow = 5, ncol = 4)

# Ancestry matrix: same dimensions
ancestry <- matrix(c(3, 2, 1, 3, 2, 1, 2, 2, 1, 1,
                    3, 1, 2, 1, 3, 2, 1, 2, 1, 3), nrow = 5, ncol = 4)

result <- split_diploid(gt, ancestry)
result$african
result$european
```

split_diploid_multi	<i>Split unphased genotype matrix by ancestry for K populations</i>
---------------------	---

Description

Generalises `split_diploid` to an arbitrary number of ancestries. For each variant, population allele proportions $p_1, \dots, p_K$ (summing to 1) are estimated from samples with unambiguous ancestry (pure-ancestry homozygotes and heterozygotes, plus mixed hom-alt which each contribute one allele to each parent pool). Ambiguous heterozygotes in each mixed pair are then split by the conditional pairwise ratio $p_i/(p_i + p_j)$. When no unambiguous alt alleles exist for a pair (singleton edge case), the split defaults to 0.5 / 0.5.

Usage

```
split_diploid_multi(
  gt_genotype,
  ancestry,
  pure_codes,
  mixed_codes,
```

```

    gla = NULL,
    arm_id = NULL
  )

```

Arguments

gt_genotype	Integer matrix (variants x samples) of genotypes (0/1/2).
ancestry	Integer matrix (variants x samples) of ancestry codes, same dimensions as gt_genotype.
pure_codes	Named integer vector mapping each population label to its pure-ancestry diploid code. Example: c(AFR = 3L, EUR = 1L, NAT = 4L).
mixed_codes	Data frame with three columns: code Integer ancestry code for this mixed pair. pop1 Character name of the first parent population (must be a name in pure_codes). pop2 Character name of the second parent population (must be a name in pure_codes). One row per mixed-ancestry diploid type. Example for 3 populations: data.frame(code=c(2L, 5L, 6L), pop1=c("AFR", "AFR", "EUR"), pop2=c("EUR", "NAT", "NAT"))
gla	Optional 2 x K numeric matrix (rows "p"/"q", columns matching names(pure_codes)) of per-arm global local ancestry proportions, as produced by ancestry_split internally. When NULL (default), GLA shrinkage is disabled and behavior matches the original p[k] estimator exactly, including its 0.5/0.5 singleton fallback.
arm_id	Optional integer vector (length = nrow(gt_genotype)) giving each variant's row index into gla (0-based). Required together with gla.

Value

Named list of K numeric matrices (variants x samples), one per population in pure_codes, in the same order. Each element is named after the corresponding entry in pure_codes.

See Also

[split_diploid](#) for the hardcoded 2-population (AFR + EUR) wrapper; [split_haplotype_multi](#) for the phased K-population analogue.

Examples

```

# 3-population example: AFR=3, EUR=1, NAT=4; mixed codes 2/5/6
set.seed(1)
gt <- matrix(sample(0:2, 20, replace = TRUE, prob = c(.8,.15,.05)),
             nrow = 4, ncol = 5)
# ancestry codes: 3=AFR/AFR, 1=EUR/EUR, 4=NAT/NAT,
#                2=AFR/EUR, 5=AFR/NAT, 6=EUR/NAT
anc <- matrix(sample(c(1L,2L,3L,4L,5L,6L), 20, replace = TRUE),
             nrow = 4, ncol = 5)
pure <- c(AFR = 3L, EUR = 1L, NAT = 4L)
mixed <- data.frame(code = c(2L, 5L, 6L),
                   pop1 = c("AFR", "AFR", "EUR"),
                   pop2 = c("EUR", "NAT", "NAT"))
out <- split_diploid_multi(gt, anc, pure, mixed)

```

```
names(out) # "AFR" "EUR" "NAT"
```

split_haplotype

Split phased haplotypes by ancestry

Description

Split phased haplotypes deterministically into African and European ancestry-specific dosage matrices. Each haplotype's allele is added to whichever ancestry pool matches its local ancestry call; haplotypes with unrecognised ancestry codes contribute nothing to either pool.

Usage

```
split_haplotype(
  gt_hap0,
  gt_hap1,
  anc_hap0,
  anc_hap1,
  pop_codes = c(AFR = 0L, EUR = 1L)
)
```

Arguments

gt_hap0	Integer matrix (variants x samples) of haplotype-0 alleles (0/1).
gt_hap1	Integer matrix (variants x samples) of haplotype-1 alleles (0/1).
anc_hap0	Integer matrix (variants x samples) of haplotype-0 ancestry codes.
anc_hap1	Integer matrix (variants x samples) of haplotype-1 ancestry codes.
pop_codes	Named integer vector of length 2 giving the AFR and EUR ancestry codes as used in the MSP file. Defaults to c(AFR = 0L, EUR = 1L) (RFMix convention).

Value

List with two numeric matrices:

african	African ancestry-specific dosage (variants x samples)
european	European ancestry-specific dosage (variants x samples)

Examples

```
gt0 <- matrix(c(1L, 0L, 0L, 1L), 2, 2)
gt1 <- matrix(c(0L, 1L, 1L, 0L), 2, 2)
a0 <- matrix(c(0L, 1L, 0L, 1L), 2, 2)
a1 <- matrix(c(1L, 0L, 1L, 0L), 2, 2)
split_haplotype(gt0, gt1, a0, a1)
```

split_haplotype_multi *Split phased haplotypes into K population-specific dosage matrices*

Description

Generalises [split_haplotype](#) to an arbitrary number of ancestries. Each haplotype's allele is routed to the dosage pool whose code matches its local ancestry call. Haplotypes with unrecognised codes contribute 0 to all pools. NA genotypes are treated as 0 (reference allele).

Usage

```
split_haplotype_multi(  
  gt_hap0,  
  gt_hap1,  
  anc_hap0,  
  anc_hap1,  
  pop_codes = c(AFR = 0L, EUR = 1L)  
)
```

Arguments

gt_hap0	Integer matrix (variants x samples) of haplotype-0 alleles (0/1).
gt_hap1	Integer matrix (variants x samples) of haplotype-1 alleles (0/1).
anc_hap0	Integer matrix (variants x samples) of haplotype-0 ancestry codes.
anc_hap1	Integer matrix (variants x samples) of haplotype-1 ancestry codes.
pop_codes	Named integer vector mapping population label to ancestry code, e.g. c(AFR = 0L, EUR = 1L, NAT = 2L). Codes must match those used in the MSP file (RFMix default: 0-based integers).

Details

For the two-population case `split_haplotype_multi` and `split_haplotype` are equivalent; prefer `split_haplotype_multi` for new code that may later be extended to three or more ancestries.

Value

Named list of K numeric matrices (variants x samples), one per entry in `pop_codes`, in the same order. List names equal the names of `pop_codes`.

See Also

[split_haplotype](#) for the two-population convenience wrapper, [split_diploid](#) for unphased (diploid-code) splitting.

Examples

```
# Three-population panel: AFR=0, EUR=1, NAT=2
gt0 <- matrix(c(1L, 0L, 0L, 1L, 1L, 0L), nrow = 3, ncol = 2)
gt1 <- matrix(c(0L, 1L, 1L, 0L, 0L, 1L), nrow = 3, ncol = 2)
a0 <- matrix(c(0L, 1L, 2L, 0L, 1L, 2L), nrow = 3, ncol = 2)
a1 <- matrix(c(1L, 2L, 0L, 2L, 0L, 1L), nrow = 3, ncol = 2)
out <- split_haplotype_multi(gt0, gt1, a0, a1, c(AFR = 0L, EUR = 1L, NAT = 2L))
names(out) # "AFR" "EUR" "NAT"
out$NAT    # native-ancestry dosage matrix
```

write_ancestry_gds	<i>Write all populations of an ancestry split to GDS files (Step 2)</i>
--------------------	---

Description

Takes the list returned by [ancestry_split](#) (or the backward-compatible [ancestry_split_dosage](#) / [ancestry_split_phased](#) output, which has the same per-population matrix + variant_info + sample_ids shape) and writes one GDS file per population via [write_dosage_gds](#), printing progress as it goes. This is Step 2 of the package's Step1/Step2/Step3 workflow: most users only need to call it so they have GDS paths to pass to [ancestry_smmat](#) (Step 3); it's exported mainly for standalone use (e.g. caching GDS files between SLURM jobs).

Usage

```
write_ancestry_gds(split_result, out_path, verbose = TRUE)
```

Arguments

split_result	List as returned by ancestry_split : named per-population dosage matrices plus variant_info and sample_ids.
out_path	Output directory (created if it doesn't exist). One file <population>.gds is written per population.
verbose	Print per-population progress messages.

Value

Named list of GDS file paths, one per population, e.g. `list(AFR = "out/AFR.gds", EUR = "out/EUR.gds")`. Pass this directly as `gds_paths` to [ancestry_smmat](#).

See Also

[ancestry_split](#), [ancestry_smmat](#)

Examples

```
vcf <- system.file("extdata", "toy.vcf.gz", package = "lantern")
msp <- system.file("extdata", "toy.msp.tsv", package = "lantern")
split <- ancestry_split(vcf, msp, mode = "dosage", chrom = "chr19",
                        verbose = FALSE)

out_dir <- tempfile()
gds <- write_ancestry_gds(split, out_dir, verbose = FALSE)
gds  # one GDS file per population
unlink(out_dir, recursive = TRUE)
```

write_dosage_gds	<i>Write an ancestry-specific dosage matrix to a SeqArray GDS file</i>
------------------	--

Description

Converts a dosage matrix (rows = variants, columns = samples) to a SeqArray GDS file via a temporary VCF with a DS FORMAT field. The resulting GDS can be passed directly to `GMMAT::SMMAT()` via the `gds.fn` argument with `is.dosage = TRUE`.

Usage

```
write_dosage_gds(dosage_mat, variant_info, sample_ids, gds_path)
```

Arguments

dosage_mat	Numeric matrix; rows = variants, columns = samples. Values should be in [0, 2].
variant_info	Data frame with columns <code>chrom</code> , <code>pos</code> , <code>ref</code> , <code>alt</code> (one row per variant in the same order as rows of <code>dosage_mat</code>). Returned directly by ancestry_split .
sample_ids	Character vector of sample identifiers, length equal to <code>ncol(dosage_mat)</code> .
gds_path	Output file path for the GDS file (e.g. "afr.gds").

Value

Invisibly returns `gds_path`.

See Also

[ancestry_split](#), [write_ancestry_gds](#), [cauchy_combine](#)

Examples

```
vcf <- system.file("extdata", "toy.vcf.gz", package = "lantern")
msp <- system.file("extdata", "toy.msp.tsv", package = "lantern")
split <- ancestry_split(vcf, msp, mode = "haplotype", chrom = "chr19",
                        verbose = FALSE)
afr_gds <- tempfile(fileext = ".gds")
write_dosage_gds(split$AFR, split$variant_info, split$sample_ids, afr_gds)
```

```
unlink(afr_gds)
```

Index

ancestry_smmat, [2](#), [5](#), [6](#), [10](#), [16](#)
ancestry_split, [4](#), [5](#), [7](#), [8](#), [10](#), [12](#), [13](#), [16](#), [17](#)
ancestry_split_dosage, [6](#), [10](#), [11](#), [16](#)
ancestry_split_phased, [8](#), [16](#)

cauchy_combine, [2](#), [4](#), [9](#), [17](#)

read_bed_file, [10](#)

split_diploid, [7](#), [10](#), [11](#), [11–13](#), [15](#)
split_diploid_multi, [5](#), [6](#), [12](#)
split_haplotype, [14](#), [15](#)
split_haplotype_multi, [5](#), [13](#), [15](#)

write_ancestry_gds, [2–6](#), [8](#), [9](#), [16](#), [17](#)
write_dosage_gds, [10](#), [16](#), [17](#)