Package 'lantern'

November 7, 2026

Type Package

Title Ancestry-Specific Rare Variant Association Analysis

Version 0.99.1

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, where ambiguous mixed-ancestry heterozygotes are split with shrinkage toward 1/2, 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> 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>.

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, GMMAT, SeqArray, SeqVarTools, stats, utils

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

VignetteBuilder knitr

biocViews Software, Genetics, GenomeWideAssociation, SNP, VariantAnnotation

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

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

git_branch devel

git_last_commit aec0b24

git_last_commit_date 2026-10-29

Repository Bioconductor 3.24

Date/Publication 2026-11-06

Author Yu Wang [aut, cre],
Bjoernar Tuftin [aut],
Paul Auer [aut]

Maintainer Yu Wang <ywang@mcw.edu>

Contents

lantern-package 2
.compute_gene_weights 3
.parse_msp 4
.parse_phased_gt_matrix 4
.parse_vcf_msp_common 5
.resolve_vcf_contig 5
.vcf_reader 5
.vcf_read_gt 6
.vcf_sample_ids 6
ancestry_smmat 6
ancestry_split 9
ancestry_split_dosage 10
ancestry_split_phased 12
cauchy_combine 13
count_ancestry_codes 14
read_bed_file 15
split_diploid 16
split_diploid_multi 17
split_haplotype 18
split_haplotype_multi 19
write_ancestry_gds 20
write_dosage_gds 21

Index **22**

`lantern-package` *lantern: Ancestry-Specific Rare Variant Analysis*

Description

R package for ancestry-specific rare variant association analysis with pure C backend for performance-critical operations.

Step 1 – split

`ancestry_split`

Step 2 – write GDS

`write_ancestry_gds`

Step 3 – associate

ancestry_smmat

Author(s)

Maintainer: Yu Wang <ywang@mcw.edu>

Authors:

- Yu Wang <ywang@mcw.edu>
- Bjoernar Tuftin <btuftin@email.unc.edu>
- Paul Auer <pauer@mcw.edu>

See Also

Useful links:

- https://yuw444.github.io/lantern/
- https://github.com/yuw444/lantern
- Report bugs at https://github.com/yuw444/lantern/issues

.compute_gene_weights *Per-gene ancestry weights for the Cauchy combination (Step 3 helper)*

Description

Generalises the retired src/step3_weight_finding.R script's median per-gene ancestry count logic to K populations (that functionality now lives here, fed by ancestry_counts/ variant_info from Step 1): for each gene in gene_group_file, takes the median (across the gene's variants) of the per-variant pure- ancestry sample counts in ancestry_counts.

Usage

```
.compute_gene_weights(
  ancestry_counts,
  variant_info,
  gene_group_file,
  pop_names
)
```

Arguments

ancestry_counts	Numeric matrix (variants x populations), as returned by ancestry_split.
variant_info	data.frame with columns chrom/pos, row-aligned to ancestry_counts, as returned by ancestry_split.
gene_group_file	Path to the gene_group file (no header; columns gene, chr, pos, ref, alt, weight) – the same file passed to GMMAT::SMMAT().
pop_names	Character vector of population names to produce weight columns for (typically names(gds_paths) from ancestry_smmat).

Value

Numeric matrix (genes x pop_names), or NULL if none of pop_names match colnames(ancestry_counts).

`.parse_msp` *Parse RFMix MSP file*

Description

Parse the RFMix MSP (local ancestry tract) file into ancestry matrices. MSP format: two header lines followed by data rows. Line 1: population codes (e.g., "#Subpopulation order/codes: AFR=0 EUR=1"). Line 2: column headers. Data rows: ancestry calls per haplotype per tract.

Usage

```
.parse_msp(msp_path, verbose = TRUE)
```

Arguments

msp_path	Path to MSP file (plain text or gzipped).
verbose	Print progress messages.

Value

List with: pop_codes (named integer vector), sample_ids (character), tract_df (data.frame sorted by spos), anc_hap0 (integer matrix: samples x tracts), anc_hap1 (integer matrix: samples x tracts).

`.parse_phased_gt_matrix`
Parse phased GT matrix into haplotype integer matrices (vectorised)

Description

Parse phased GT matrix into haplotype integer matrices (vectorised)

Usage

```
.parse_phased_gt_matrix(
  gt_mat,
  n_variants,
  n_samples,
  sample_order,
  verbose = FALSE
)
```

Arguments

gt_mat	Character matrix (variants x samples).
n_variants	Number of variants.
n_samples	Number of samples.
sample_order	Column names for output matrices.
verbose	Print progress.

Value

List: `hap0`, `hap1` (integer matrices), `missing` (logical matrix).

`.parse_vcf_msp_common` *Parse an MSP file and a phased VCF into haplotype/ancestry matrices*

Description

Internal helper shared by `ancestry_split`: parses the RFMix MSP file, queries VCF sample IDs, intersects samples, parses phased GT into haplotype matrices, maps each variant to its ancestry tract, and broadcasts tract-level ancestry to variant-level ancestry.

Usage

```
.parse_vcf_msp_common(vcf_path, msp_path, chrom = NULL, verbose = TRUE)
```

`.resolve_vcf_contig` *Resolve a chromosome name against a VCF/BCF's declared contigs*

Description

Tries `chrom` as given, with a leading "chr" stripped, and with a leading "chr" added, against the `##contig=<ID=...>` lines in the file's header. Used to restrict `bcftools query` to one chromosome via `-t` without guessing the file's naming convention. Returns `NULL` (triggering an unrestricted query, filtered in R instead) if the header has no contig lines or none of the candidates match.

Usage

```
.resolve_vcf_contig(vcf_path, chrom)
```

`.vcf_reader` *Choose the VCF-reading backend*

Description

`"bcftools"` when the `bcftools` binary is on `PATH` (fastest; also reads BCF), otherwise `"seqarray"` (no external binary). `options(lantern.vcf_reader = "bcftools"/"seqarray")` forces one; the default `"auto"` picks as above.

Usage

```
.vcf_reader()
```

.vcf_read_gt *Read variant sites and GT strings from a VCF*

Description

Both backends return the same thing: `chrom`, `pos`, `ref`, `alt` vectors and `gt`, a character matrix (variants x all VCF samples) of GT strings exactly as `bcftools query -f '[%GT]'` prints them (e.g. `"0|1"`, `"./."`), so everything downstream is backend-agnostic. `chrom` restricts the read where the backend can do so cheaply; callers still filter by chromosome afterward.

Usage

```
.vcf_read_gt(vcf_path, chrom, reader, verbose = TRUE)

.vcf_read_gt_seqarray(vcf_path, chrom, verbose = TRUE)
```

.vcf_sample_ids *Read the sample IDs of a VCF*

Description

Read the sample IDs of a VCF

Usage

```
.vcf_sample_ids(vcf_path, reader)
```

ancestry_smmat *Run ancestry-stratified SMMAT + Cauchy combination (Step 3)*

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

gds_paths	Named list/vector of GDS file paths, one per ancestry population, e.g. `c(AFR = "afr.gds", EUR = "eur.gds")` – as returned by `write_ancestry_gds`. Any K >= 1 populations are supported; add an extra named entry (e.g. `OBSERVED = ...`) to include an additional (e.g. non-ancestry-split) GDS in the same SMMAT + Cauchy combination run.
pheno	data.frame with an `id_col` column plus the response and covariates referenced in `formula`.
formula	Model formula for the null model, e.g. `y ~ age + sex`.
kinship	Numeric matrix (or object coercible via `as.matrix`) with row/column names equal to sample IDs.
gene_group_file	Either a path to the gene_group file passed to `GMMAT::SMMAT()` (no header; columns gene, chr, pos, ref, alt, weight), or a data.frame/matrix with those same six columns. Either way, it is read, has any leading `"chr"` stripped from the chr column (matching `SeqArray::seqVCF2GDS()`'s convention – see the "chr column" note below), and written to a fresh temporary file that is deleted with `unlink()` when `ancestry_smmat()` returns; the caller's own path/data.frame 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 `GMMAT::SMMAT()` matches variants by exact `chr:pos:ref:alt` string equality with no fuzzy/positional alignment.
ancestry_counts, variant_info	Optional: pass `ancestry_split()$ancestry_counts` / `$variant_info` straight through to compute per-gene ancestry weights for the Cauchy combination. If omitted, equal weights are used.
id_col	Name of the sample ID column in `pheno` and `formula`'s `id` argument to `glmmkin` (default `"id"`).
family	A family object for `GMMAT::glmmkin()` (default `gaussian()`).
gene_col	Name of the gene/group ID column in `SMMAT()`'s output (default `"group"`, matching `GMMAT::SMMAT()`).
p_col	Name of the p-value column in `SMMAT()`'s output to combine (default `"E.pval"`, matching `GMMAT::SMMAT()`'s default `tests = "E"`; use `"O.pval"`/`"S.pval"`/`"B.pval"` if you pass a different `tests` via `...`).

`MAF.range`, `miss.cutoff`, `method`, `ncores`
: Passed straight through to `GMMAT::SMMAT()` – same names, same meaning as in a direct `SMMAT()` call.

`verbose`
: Print this function's own step-by-step progress messages (Steps 1-6 below). Distinct from `GMMAT::SMMAT()`'s own internal `verbose` (its per-call progress bar), which is not exposed separately and stays at SMMAT's default (`FALSE`); every other `SMMAT()` argument is available via `...`.

`...`
: Any other `GMMAT::SMMAT()` argument, forwarded as-is – for example `tests` (`"B"`, `"S"`, `"O"`, or the default `"E"`), `rho`, `MAF.weights.beta`, `missing.method`, `use.minor.allele`, `auto.flip`, `Garbage.Collection`, `group.file.sep`, or `meta.file.prefix`. `null.obj`, `geno.file`, `group.file`, and `is.dosage` are always supplied internally by `ancestry_smmat()` and cannot be overridden this way. See `?GMMAT::SMMAT` for the full list and defaults.

Value

A list with two elements:

`results`
: data.frame with one row per gene: `gene`, one `p_<POP>` column per population, one `w_<POP>` weight column per population (only if `ancestry_counts`/`variant_info` were supplied), and `p_cauchy`.

`smmat_results`
: Named list (one entry per population, same names as `gds_paths`) of the raw `GMMAT::SMMAT()` result data.frames.

See Also

`ancestry_split`, `write_ancestry_gds`, `cauchy_combine`

Examples

```
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,
  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).
`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 `ancestry_smmat`'s per-gene weight calculation.

`tract_info`
: data.frame of ancestry tracts from the MSP file.

`overlap`
: List of intersection/filtering statistics.

Shrinkage toward 1/2 (dosage mode)

A mixed-ancestry heterozygote carries one haplotype from each of its two ancestries, so which one holds the alt allele cannot be read off the genotype. Its allele is split between the two ancestries in proportion to the variant's unambiguous carriers (the raw ratio), shrunk toward 1/2 by w, the share of the variant's alt-allele evidence that comes from ambiguous heterozygotes:

$$p = (1 - w)\, p_{raw} + w/2.$$

A variant carried only by ambiguous heterozygotes ($w = 1$) is split 1/2 : 1/2. With K populations every pair is shrunk toward 1/2 in the same way, with w computed from that pair's own evidence (the unambiguous alleles on its two populations and its ambiguous heterozygotes); see `split_diploid_multi`. `mode = "haplotype"` is deterministic and applies no shrinkage.

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

gt_matrix	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 vcf_path and msp_path are supplied.
pt_matrix	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 gt_matrix). colnames: region IDs (e.g., "chr:start-end"). May be NULL when vcf_path and msp_path are supplied.
vcf_path	Path to a phased VCF or BCF file. When supplied together with msp_path, the function delegates to ancestry_split (Step 1, mode = "dosage"), bypassing the gt_matrix/pt_matrix arguments.
msp_path	Path to an RFMix MSP file (plain text or gzipped TSV). Used together with vcf_path as a shortcut input.
chrom	Chromosome name to restrict to (e.g. "chr19"). Only used when vcf_path/msp_path are supplied.
verbose	Print progress messages (default TRUE)

Value

List with elements:

african	African ancestry-specific dosage matrix (variants x samples)
european	European ancestry-specific dosage matrix
counts	List (african, european, mixed) of per-variant counts of samples with ancestry code 3, 1 and 2
overlap	Overlap information

Shrinkage

Both input paths split ambiguous mixed-ancestry heterozygotes with the same shrinkage toward 1/2 as ancestry_split (see its "Shrinkage toward 1/2" section). The target is the same for every variant, so the direct gt_matrix/pt_matrix path needs no position information.

Matching variants to regions

Each variant (row of gt_matrix) is paired with an ancestry column of pt_matrix by, in order of preference: identical names; position, when rownames look like "chr:pos" and colnames like "chr:start-end" (the first region containing the variant; several variants may share a region); or, when either matrix has no names, position in the matrix.

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 = NULL,
  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 the GDS files. Only needed (and then required) when `write_vcf = TRUE`.
`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

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))))
```

---

count_ancestry_codes *Count ancestry codes in matrix*

Description

Count occurrences of a specific ancestry code in each row of a matrix. Internal building block used by ancestry_split (to compute ancestry_counts) and ancestry_split_dosage.

Usage

```
count_ancestry_codes(mat, code)
```

Arguments

mat	Integer matrix with ancestry codes (1, 2, 3) - Rows represent genomic regions/windows - Columns represent samples
code	Ancestry code to count: - 1 = EUR/EUR (Pure European) - 2 = AFR/EUR (Mixed) - 3 = AFR/AFR (Pure African)

Value

Integer vector with count of code per row

`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 = sub("\\.bed$", ".bim", bed),
  fam = sub("\\.bed$", ".fam", bed)
)
```

Arguments

`bed`	Path to the `.bed` file.
`bim`	Path to the `.bim` file (variant IDs are taken from column 2). Defaults to `bed` with its extension replaced by `.bim`, the usual PLINK file set layout.
`fam`	Path to the `.fam` file (sample IDs are taken from column 2, i.e. IID). Defaults to `bed` with its extension replaced by `.fam`.

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")
# .bim and .fam are found next to the .bed file
pt <- read_bed_file(file.path(ext, "toy_ancestry.bed"))
pt[, 1:6]   # ancestry tracts x samples, codes 1/2/3
```

`split_diploid` *Split genotype matrix by ancestry*

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, shrink = FALSE)
```

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
`shrink`	If `TRUE`, shrink each mixed heterozygote's split toward 1/2 by $w = N_5/(D + N_5)$, the ambiguous share of the variant's alternate alleles (D unambiguous, N_5 ambiguous; hom-alt carriers count 2), as `ancestry_split` does. If `FALSE` (default), the raw p1/p2 ratio is used, with a 0.5/0.5 fallback only when every carrier is ambiguous.

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` *Split unphased genotype matrix by ancestry for K populations*

Description

Generalises `split_diploid` to an arbitrary number of ancestries. For each variant, population allele proportions $p_1, \ldots, p_K$ (summing to 1) are estimated from samples with unambiguous ancestry (pure-ancestry homozygotes and heterozygotes, plus mixed hom-alts 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 = c(AFR = 3L, EUR = 1L),
  mixed_codes = data.frame(code = 2L, pop1 = "AFR", pop2 = "EUR"),
  shrink = FALSE
)
```

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)`. The default is the two-population coding used by `split_diploid` (3 = AFR/AFR, 1 = EUR/EUR).
`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. The default is the two-population mixed code 2 = AFR/EUR, so with both defaults `split_diploid_multi(gt, ancestry)` matches `split_diploid(gt, ancestry)`. Example for 3 populations: `data.frame(code=c(2L,5L,6L), pop1=c("AFR","AFR","EUR"), pop2=c("EUR","NAT","N`
`shrink`	If `TRUE`, shrink each ambiguous pair (a, b) toward 1/2 by $w = N_{ab}/(n_a + n_b + N_{ab})$, the ambiguous share of that pair's own evidence: n_a, n_b are the unambiguous alleles on a and b (hom-alt carriers count 2; a mixed hom-alt gives 1 to each), and N_{ab} the pair's ambiguous heterozygotes. Alleles on other populations do not enter w. `ancestry_split` always shrinks. If `FALSE` (default), the raw p[k] ratio is used, with a 0.5/0.5 fallback only when a pair has no unambiguous evidence.

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 *Write all populations of an ancestry split to GDS files (Step 2)*

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 *Write an ancestry-specific dosage matrix to a SeqArray GDS file*

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 chrom, pos, ref, alt (one row per variant in the same order as rows of dosage_mat). Returned directly by ancestry_split.
sample_ids	Character vector of sample identifiers, length equal to ncol(dosage_mat).
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

∗ **internal**
.compute_gene_weights, 3
.parse_msp, 4
.parse_phased_gt_matrix, 4
.parse_vcf_msp_common, 5
.resolve_vcf_contig, 5
.vcf_read_gt, 6
.vcf_reader, 5
.vcf_sample_ids, 6
count_ancestry_codes, 14
lantern-package, 2
.compute_gene_weights, 3
.parse_msp, 4
.parse_phased_gt_matrix, 4
.parse_vcf_msp_common, 5
.resolve_vcf_contig, 5
.vcf_read_gt, 6
.vcf_read_gt_seqarray *(*.vcf_read_gt*)*, 6
.vcf_reader, 5
.vcf_sample_ids, 6

ancestry_smmat, *3*, 6, *9, 10*, *14*, *20, 21*
ancestry_split, *2, 3*, *5*, *8*, 9, *11, 12*, *14*, *16, 17*, *20, 21*
ancestry_split_dosage, 10, *14, 15*, *20*
ancestry_split_phased, 12, *20*

cauchy_combine, *6*, *8*, 13, *21*
count_ancestry_codes, 14

lantern *(*lantern-package*)*, 2
lantern-package, 2

read_bed_file, 15

split_diploid, *15*, 16, *17, 18*, *20*
split_diploid_multi, *9, 10*, 17
split_haplotype, 18, *19, 20*
split_haplotype_multi, *9*, *18*, 19

write_ancestry_gds, *2*, *6–10*, *12*, 20, *21*
write_dosage_gds, *14*, *20*, 21