# Package: BIGpopA (via r-universe)

October 27, 2026

**Title** Pedigree Validation Genetic Composition of Diploids & Polyploids

**Version** 2.2.1

**Maintainer** Josue Chinchilla-Vargas <josue.chinchilla@ufl.edu>

**Description** Tools for pedigree quality control and genomic breed/line composition estimation in diploid and polyploid breeding populations. 'BIGpopA' provides functions to check and correct common pedigree errors, assign parentage from SNP genotype data using Mendelian error rates, validate parent-offspring trios, and estimate genome-wide breed or line composition using quadratic programming. Pedigree validation and parentage assignment support any ploidy, using a polysomic Mendelian test for even ploidy and a homozygosity-based check for odd ploidy. Genotypes can be supplied as dosage tables, VCF files, or 'PLINK' .ped files. For more details about the included 'breedTools' functions, see Funkhouser et al. (2017) <doi:10.2527/tas2016.0003>.

**License** Apache License (>= 2)

**URL** https://CRAN.R-project.org/package=BIGpopA, https://github.com/Breeding-Insight/BIGpopA

**BugReports** https://github.com/Breeding-Insight/BIGpopA/issues

**Encoding** UTF-8

**Language** en-US

**Depends** R (>= 4.4.0)

**Imports** dplyr, janitor, quadprog, data.table, ggplot2

**Suggests** covr, knitr (>= 1.10), rmarkdown, scales, testthat (>= 3.0.0), vcfR (>= 1.15.0), withr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.1.1

**NeedsCompilation** no

**Author** Josue Chinchilla-Vargas [cre, aut], Alexander Sandercock [aut], University of Florida [cph] (Breeding Insight)

**Repository** https://cran.r-universe.dev

**Date/Publication** 2026-10-27 18:47:11 UTC

**RemoteUrl** https://github.com/cran/BIGpopA

**RemoteRef** HEAD

**RemoteSha** decd647bd8095bc96fc5317fdae0a1b14a6c5ff2

# Contents

allele_freq_poly . . . . . 2
assign_composition_line . . . . . 3
check_ped . . . . . 4
find_parentage . . . . . 6
ped_to_dosage . . . . . 9
solve_composition_poly . . . . . 11
validate_pedigree . . . . . 12
vcf_to_dosage . . . . . 15

**Index** **18**

---

`allele_freq_poly` *Compute Allele Frequencies for Populations*

---

## Description

Computes allele frequencies for specified populations given SNP array data.

## Usage

```
allele_freq_poly(geno, populations, ploidy = 2)
```

## Arguments

| | |
|---|---|
| `geno` | Genotypes coded as the dosage of allele B (0, 1, 2, ..., ploidy) as any of: a matrix or data.frame with individuals in rows (named) and SNPs in columns (named); a data.frame with an `id` / `ID` column followed by SNP columns; a path to a TSV/CSV/TXT file whose first column holds the IDs; a path to a VCF file (`.vcf` or `.vcf.gz`) or a `vcfR` object, converted with `vcf_to_dosage()` using `ploidy`; or a path to a PLINK `.ped` file (diploid only), converted with `ped_to_dosage()`. |
| `populations` | list of named populations. Each population has a vector of IDs that belong to the population. Allele frequencies will be derived from all animals in each population. |
| `ploidy` | integer indicating the ploidy level (default is 2 for diploid). |

## Details

When geno is a .ped file, the returned matrix carries a `counted_allele` attribute. `solve_composition_poly()` uses it to code a validation .ped file with the same counted allele at each marker.

## Value

A matrix of allele frequencies with SNPs in rows and populations in columns.

## References

Funkhouser SA, Bates RO, Ernst CW, Newcom D, Steibel JP. Estimation of genome-wide and locus-specific breed composition in pigs. Transl Anim Sci. 2017 Feb 1;1(1):36-44.

## Examples

```
geno_matrix <- matrix(
  c(4, 1, 4, 0,
    2, 2, 1, 3,
    0, 4, 0, 4,
    3, 3, 2, 2,
    1, 4, 2, 3),
  nrow = 4, ncol = 5, byrow = FALSE,
  dimnames = list(paste0("Ind", 1:4), paste0("S", 1:5))
)

pop_list <- list(
  PopA = c("Ind1", "Ind2"),
  PopB = c("Ind3", "Ind4")
)

allele_freqs <- allele_freq_poly(geno = geno_matrix,
                                 populations = pop_list,
                                 ploidy = 4)
print(allele_freqs)
```

---

`assign_composition_line`

*Assign Each Sample to a Reference Line*

---

## Description

Builds the composition table shown in the Familia BreedTools tab from the output of `solve_composition_poly()`: one row per sample with the estimated percentage of each reference population and the predicted line. A sample is assigned to the population with its highest proportion only if that proportion is at least `assign_threshold`; otherwise it is labelled `"Undetermined"`.

### Usage

```
assign_composition_line(composition, assign_threshold = 50)
```

### Arguments

| | |
|---|---|
| composition | Matrix or data.frame returned by `solve_composition_poly()`: samples in rows, reference populations in columns (proportions summing to 1). An `R2` column, if present, is dropped. |
| assign_threshold | Numeric between 0 and 100. Minimum percentage the highest population must reach for the sample to be assigned to it (default: 50). Use 0 to always assign the highest population. |

### Value

A data.frame with an `ID` column, one column per reference population holding the estimated percentage (0 to 100, rounded to one decimal) and a `Predicted line` column. Ties between populations go to the first column.

### Examples

```
comp <- rbind(
  Sample01 = c(Duroc = 0.998, Hampshire = 0.000, Landrace = 0.002, R2 = 0.991),
  Sample02 = c(Duroc = 0.529, Hampshire = 0.471, Landrace = 0.000, R2 = 0.973),
  Sample03 = c(Duroc = 0.340, Hampshire = 0.330, Landrace = 0.330, R2 = 0.952)
)
assign_composition_line(comp)
assign_composition_line(comp, assign_threshold = 0)
```

---

## check_ped *Check and Correct Common Pedigree Errors*

---

### Description

Reads a 3-column pedigree file (id, male_parent, female_parent) and performs quality checks, optionally correcting detected errors. Exact duplicates and missing parents are always corrected. Conflicting trios and inconsistent sex roles are corrected when their respective arguments are TRUE. Cycles are reported only and must be resolved manually.

### Usage

```
check_ped(
  ped.file,
  seed = NULL,
  verbose = TRUE,
  correct_conflicting_trios = TRUE,
  correct_inconsistent_sex_roles = TRUE
)
```

## Arguments

| | |
|---|---|
| `ped.file` | Path to the pedigree text file (TSV/CSV/TXT), OR a data.frame / data.table with columns: id, male_parent, female_parent. |
| `seed` | Optional integer seed for reproducibility. Pass NULL (default) to skip setting a seed. |
| `verbose` | Logical. If TRUE (default), prints the report to the console. |
| `correct_conflicting_trios` | Logical. If TRUE (default), sets conflicting male_parent and female_parent to 0 and collapses to one row per ID. |
| `correct_inconsistent_sex_roles` | Logical. If TRUE (default), sets male_parent and female_parent to 0 for rows involving IDs found as both, then removes any resulting exact duplicates. |

## Value

An invisible named list of data frames:

**exact_duplicates** Exact duplicate rows found in the input.

**conflicting_trios** IDs with conflicting male_parent or female_parent assignments.

**inconsistent_sex_roles** Rows where a conflicting ID appears as male_parent or female_parent.

**missing_parents** Parent IDs absent from id, added as founders.

**dependencies** Individuals recorded as their own parent, and cycles (individuals that are their own ancestors). Each cycle is listed once, as a path in which every id is a recorded parent of the one before it, back to the first id (e.g. "A -> B -> A"). Must be resolved manually.

**corrected_pedigree** Corrected pedigree table.

## Author(s)

Josue Chinchilla-Vargas

## Examples

```
# Self-contained example using a data.frame
ped_df <- data.frame(
  id            = c("A", "B", "C", "C", "D"),
  male_parent   = c("0", "0", "A", "A", "B"),
  female_parent = c("0", "0", "B", "B", "C"),
  stringsAsFactors = FALSE
)
ped_errors <- check_ped(ped.file = ped_df, seed = 101919, verbose = FALSE)
names(ped_errors)
head(ped_errors$corrected_pedigree)


library(data.table)
ped_dt <- data.table(id = c("A", "B", "C"),
                     male_parent   = c("0", "0", "A"),
                     female_parent = c("0", "0", "B"))
```

```
ped_errors <- check_ped(ped.file = ped_dt, verbose = FALSE)
```

---

`find_parentage` *Find Parentage Assignments for Progeny*

---

## Description

Assigns the most likely parent(s) to each progeny from SNP genotype data using Mendelian error rates or homozygous mismatch rates. Parents or progeny absent from the genotype file are removed with a warning.

## Usage

```
find_parentage(
  genotypes_file,
  parents_file,
  progeny_file,
  method = "best_pair",
  min_markers = 10,
  error_threshold = 5,
  show_ties = TRUE,
  allow_parent_selfing = FALSE,
  exclude_self_match = TRUE,
  verbose = TRUE,
  plot_results = TRUE,
  ploidy = 2,
  top_n = 1,
  founders_file = NULL
)
```

## Arguments

| | |
|---|---|
| `genotypes_file` | Genotypes as any of: a path to a TSV/CSV/TXT file; a path to a VCF file (`.vcf` or `.vcf.gz`) or a `vcfR` object, converted with `vcf_to_dosage()` using `ploidy`; a path to a PLINK `.ped` file (diploid only), converted with `ped_to_dosage()`; or a data.frame / data.table with an 'id' column followed by marker columns coded as allele-B dosage (0, 1, ..., ploidy). |
| `parents_file` | Path to a TSV/CSV/TXT file, OR a data.frame / data.table with an 'id' column and an optional 'sex' column ('M', 'F', or 'A'). If absent, all parents are treated as ambiguous. |
| `progeny_file` | Path to a TSV/CSV/TXT file, OR a data.frame / data.table with an 'id' column. For `method = "fill_pedigree"` it must be a pedigree with `id`, `male_parent` and `female_parent` columns (e.g. the `corrected_pedigree` from `validate_pedigree()`); see Details. |

`method` Character. One of "best_male_parent", "best_female_parent", "best_match", "best_pair" (default), or "fill_pedigree" (requires a pedigree as `progeny_file`; see Details).

`min_markers` Integer. Minimum markers required; fewer flags low_markers (default: 10).

`error_threshold`
Numeric. Maximum mismatch percentage; exceeded values flag high_error (default: 5.0). Must be between 0 and 100.

`show_ties` Logical. If TRUE, tied best pairs are appended as suffix columns. Default is TRUE.

`allow_parent_selfing`
Logical. If FALSE, candidate pairs with identical male and female parent IDs are excluded. Applies only when method is "best_pair". Default is FALSE.

`exclude_self_match`
Logical. If TRUE, each progeny ID is excluded from its own candidate parent set, preventing self-matches when progeny are also present in the parents file. Default is TRUE.

`verbose` Logical. If TRUE, prints progress and summary. Default is TRUE.

`plot_results` Logical. If TRUE, plots the Mendelian error distribution. Requires ggplot2. Default is TRUE.

`ploidy` Integer >= 2. Ploidy level of the species (2 = diploid, 4 = tetraploid, ...). Genotypes must be coded as allele-B dosage (0, 1, ..., ploidy). Even ploidy uses the polysomic gamete-range Mendelian test (assumes autopolyploid inheritance; conservative for allopolyploids). Odd ploidy (e.g. triploid), where balanced gametes are undefined, falls back to a model-free opposite-homozygote exclusion evaluated on homozygous-informative markers only (reduced power). Default is 2.

`top_n` Integer >= 1. Number of best candidates to report per progeny. With the default of 1, only the best assignment is reported (plus tied best pairs when `show_ties = TRUE` and `method = "best_pair"`). With `top_n > 1`, candidates are ranked by error (ties broken by the number of markers tested) and the best `top_n` are reported: rank 1 in the base columns and ranks 2 to `top_n` in suffix columns (`male_parent_2`, `female_parent_2`, `mendelian_error_pct_2`, `markers_tested_2` for "best_pair"; `best_match_2`, `mendelian_error_pct_2`, `markers_tested_2` for the other methods). Ties are then ranked like any other candidate, so `show_ties` has no effect. Progeny with fewer than `top_n` candidates have `NA` in the remaining ranks. Status refers to rank 1 only.

`founders_file` Optional. Founder IDs as a path to a one-column file, a character vector of ids, or a data.frame (first column). Only used when `progeny_file` is a pedigree: founders with both parents unknown are skipped and not searched. Default is NULL.

## Details

**Method "fill_pedigree" ("fill in the blanks").** The `progeny_file` must have `male_parent` and `female_parent` columns. A parent coded `0`, `NA` or empty is treated as unknown and each row is handled on its own: a known male searches only females, a known female searches only males, and

a row with both unknown gets the full pair search (unless listed in `founders_file`). Rows with both parents known are skipped. The search is the "best_pair" Mendelian trio test with the known parent held fixed, and candidates come from `parents_file` (sex M/F/A as before); the known parent only needs genotypes. Two columns are added to the results: `search_mode` ("pair", "known_male" or "known_female") and `known_parent_error_pct` (homozygous mismatch between the known parent and the progeny; a high value suggests the known parent is wrong). A known parent absent from the genotype file gives `status = "known_parent_no_genotype"`. Without `method = "fill_pedigree"`, any `male_parent` / `female_parent` columns in `progeny_file` are ignored.

## Value

A named list (returned invisibly) with elements:

**pass** Progeny with a confident parentage assignment.

**high_error** Progeny whose best assignment exceeds the error threshold.

**low_markers** Progeny with insufficient markers for a valid assignment.

**full_results** Complete data.table with all progeny and all output columns. `tied_candidates` gives the number of candidates (pairs for "best_pair") that share the lowest error for each progeny; values above 1 mean the best assignment is ambiguous.

**plot** ggplot object if plot_results = TRUE, otherwise NULL.

## Author(s)

Josue Chinchilla-Vargas

## Examples

```
geno_df <- data.frame(
  id  = c("P1", "P2", "P3", "Off1", "Off2"),
  S1  = c(0L, 2L, 0L, 1L, 0L),
  S2  = c(2L, 0L, 2L, 1L, 2L),
  S3  = c(0L, 2L, 0L, 1L, 0L),
  S4  = c(2L, 0L, 2L, 1L, 2L),
  S5  = c(0L, 2L, 0L, 1L, 0L),
  S6  = c(2L, 0L, 2L, 1L, 2L),
  S7  = c(0L, 2L, 0L, 1L, 0L),
  S8  = c(2L, 0L, 2L, 1L, 2L),
  S9  = c(0L, 2L, 0L, 1L, 0L),
  S10 = c(2L, 0L, 2L, 1L, 2L)
)

parents_df <- data.frame(
  id  = c("P1", "P2", "P3"),
  sex = c("M",  "F",  "F"),
  stringsAsFactors = FALSE
)

progeny_df <- data.frame(
  id = c("Off1", "Off2"),
  stringsAsFactors = FALSE
```

```
)

results <- find_parentage(
  genotypes_file = geno_df,
  parents_file   = parents_df,
  progeny_file   = progeny_df,
  method         = "best_pair",
  verbose        = FALSE,
  plot_results   = FALSE
)
print(results$full_results)
```

---

ped_to_dosage *Convert a PLINK .ped File to a BIGpopA Genotype Matrix*

---

## Description

Reads a PLINK 1 text pedigree file (.ped) and returns allele dosages in the layout used by the rest of BIGpopA: individuals in rows, markers in columns, coded as the number of copies of the counted allele (0, 1, 2) with NA for missing calls. PLINK .ped files are diploid only.

## Usage

```
ped_to_dosage(
  ped,
  map = "auto",
  format = c("data.frame", "matrix"),
  counted_allele = NULL,
  missing_codes = c("0", "N", "-", "."),
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| ped | Path to a PLINK .ped file (whitespace separated, two allele columns per marker). |
| map | "auto" (default) uses a .map file with the same name next to the .ped when one exists. A path uses that .map file; NULL ignores any .map and names markers by position. |
| format | Character. "data.frame" (default) returns a data.table with an id column followed by one column per marker, the input expected by find_parentage() and validate_pedigree(). "matrix" returns a numeric matrix with individuals in named rows and markers in named columns, the input expected by allele_freq_poly() and solve_composition_poly(). |
| counted_allele | Named character vector, or NULL (default). Allele to count at each marker, named by marker. Markers not listed use the default rule. Typically the counted_allele attribute of a previous result. |

| | |
|---|---|
| missing_codes | Character vector of allele codes treated as missing. Default is c("0", "N", "-", "."). A call is missing when either allele is missing. |
| verbose | Logical. If TRUE, prints a conversion report. Default is TRUE. |

## Details

The .ped format stores allele letters and has no reference/alternate pair, so the counted allele (allele B) is chosen per marker. By default it is the alphabetically / numerically last allele observed at the marker, so 1/2 coded files count allele 2 and an A/G marker counts G. To code a second file the same way as a first one (for example validation genotypes against a reference panel), pass the first file's counted_allele attribute through the counted_allele argument. Otherwise a marker that is monomorphic in one file can be coded in the opposite direction.

Only the individual ID (column 2) is used from the six leading pedigree columns. A .map file is optional and only supplies marker names; it is not used for allele coding. Without it, markers are named SNP1, SNP2, ... in file order.

## Value

A data.table with an id column followed by marker columns (format = "data.frame"), or an integer matrix of individuals by markers (format = "matrix"). The attribute "ploidy" is 2 and the attribute "counted_allele" holds the allele counted at each returned marker. Markers with more than two alleles are dropped with a warning.

## Author(s)

Josue Chinchilla-Vargas

## See Also

vcf_to_dosage(), find_parentage(), allele_freq_poly()

## Examples

```
ped_path <- tempfile(fileext = ".ped")
writeLines(c(
  "F1 P1 0 0 1 -9 A A G G A G",
  "F1 P2 0 0 2 -9 A G G G G G",
  "F1 O1 P1 P2 1 -9 A G G G A G"
), ped_path)

genotypes <- ped_to_dosage(ped_path, map = NULL, verbose = FALSE)
print(genotypes)
attr(genotypes, "counted_allele")

unlink(ped_path)
```

solve_composition_poly

*Compute Genome-Wide Breed Composition*

## Description

Computes genome-wide breed/ancestry composition using quadratic programming on a batch of animals.

## Usage

```
solve_composition_poly(Y, X, ploidy = 2)
```

## Arguments

| | |
|---|---|
| Y | Genotypes (columns) from all animals (rows) in the population, coded as dosage of allele B (0, 1, 2, ..., ploidy), as any of: a numeric matrix or data.frame with named rows; a data.frame with an `id` / `ID` column followed by SNP columns; a path to a TSV/CSV/TXT file whose first column holds the IDs; a path to a VCF file (`.vcf` or `.vcf.gz`) or a `vcfR` object, converted with `vcf_to_dosage()` using `ploidy`; or a path to a PLINK `.ped` file (diploid only), converted with `ped_to_dosage()`. A `.ped` file is coded with the `counted_allele` attribute of `X` when present (see `allele_freq_poly()`), so reference and validation `.ped` files count the same allele at each marker. |
| X | numeric matrix of allele frequencies (rows) from each reference panel (columns). Frequencies are relative to allele B. |
| ploidy | integer. The ploidy level of the species (e.g., 2 for diploid, 3 for triploid). |

## Value

A matrix with one row per animal, one column per reference population (estimated proportions, summing to 1), and an `R2` column.

## References

Funkhouser SA, Bates RO, Ernst CW, Newcom D, Steibel JP. Estimation of genome-wide and locus-specific breed composition in pigs. Transl Anim Sci. 2017 Feb 1;1(1):36-44.

## Examples

```
allele_freqs_matrix <- matrix(
  c(0.625, 0.500,
    0.500, 0.500,
    0.500, 0.500,
    0.750, 0.500,
    0.625, 0.625),
  nrow = 5, ncol = 2, byrow = TRUE,
```

```
  dimnames = list(paste0("SNP", 1:5), c("VarA", "VarB"))
)

val_geno_matrix <- matrix(
  c(2, 1, 2, 3, 4,
    3, 4, 2, 3, 0),
  nrow = 2, ncol = 5, byrow = TRUE,
  dimnames = list(paste0("Test", 1:2), paste0("SNP", 1:5))
)

composition <- solve_composition_poly(Y = val_geno_matrix,
                                      X = allele_freqs_matrix,
                                      ploidy = 4)
print(composition)
```

---

validate_pedigree *Validate Pedigree Trios Using Mendelian Error Analysis*

---

## Description

Validates parent-offspring trios against SNP genotype data using Mendelian error rates. Identifies incorrect parentage assignments and outputs a corrected pedigree. Founder trios (both parents coded as 0) are preserved unchanged if a founders file is supplied. Trios absent from the genotype file are retained as no_genotype_data.

## Usage

```
validate_pedigree(
  pedigree_file,
  genotypes_file,
  founders_file = NULL,
  trio_error_threshold = 5,
  min_markers = 10,
  single_parent_error_threshold = 2,
  verbose = TRUE,
  plot_results = TRUE,
  ploidy = 2,
  marker_trio_table = FALSE
)
```

## Arguments

| | |
|---|---|
| pedigree_file | Path to the pedigree file (TSV/CSV/TXT), OR a data.frame / data.table with columns: id, male_parent, female_parent. |
| genotypes_file | Genotypes as any of: a path to a TSV/CSV/TXT file; a path to a VCF file (.vcf or .vcf.gz) or a vcfR object, converted with vcf_to_dosage() using ploidy; a path to a PLINK .ped file (diploid only), converted with ped_to_dosage(); or |

a data.frame / data.table with an id column followed by marker columns coded as allele-B dosage (0, 1, ..., ploidy).

`founders_file` Optional. Founder IDs as a path to a one-column file, a character vector of ids, or a data.frame (first column). Founders with both parents coded as 0 are left unchanged. Defaults to NULL.

`trio_error_threshold`
Numeric. Maximum Mendelian error percentage to classify a trio as pass (default: 5.0). Must be between 0 and 100.

`min_markers` Integer. Minimum non-missing markers required to evaluate a trio (default: 10).

`single_parent_error_threshold`
Numeric. Maximum homozygous-marker mismatch percentage for a parent to be considered acceptable (default: 2.0). Must be between 0 and 100.

`verbose` Logical. If TRUE, prints progress, summary, and results to the console (default: TRUE).

`plot_results` Logical. If TRUE, prints a histogram of trio Mendelian error percentages with a threshold line (default: TRUE).

`ploidy` Integer >= 2. Ploidy level of the species (2 = diploid, 4 = tetraploid, ...). Genotypes must be coded as allele-B dosage (0, 1, ..., ploidy). Even ploidy uses the polysomic gamete-range Mendelian test (assumes autopolyploid inheritance; conservative for allopolyploids). Odd ploidy (e.g. triploid), where balanced gametes are undefined, falls back to a model-free opposite-homozygote exclusion evaluated on homozygous-informative markers only (reduced power). Default is 2.

`marker_trio_table`
Logical. If TRUE, the returned list includes `marker_trio_table`, a marker-by-trio table of per-marker statuses (see Details). Default is FALSE.

## Details

The cells of `marker_trio_table` take one of these values: `match`; `mismatch_male` or `mismatch_female` (the offspring is impossible with that parent alone, whatever the other parent contributes); `mismatch_both` (the offspring is impossible with either parent alone); `mismatch_combination` (each parent is compatible alone but the pair is not, for example a 0 x 0 cross with a heterozygous offspring); `untestable` (odd ploidy, marker not informative); `missing_*` (no call, named after every individual without a call: `missing_progeny`, `missing_male`, `missing_female`, `missing_male_female`, `missing_progeny_male`, `missing_progeny_female` or `missing_progeny_male_female`).

The `recommended_correction` column of `full_results` takes these values:

**none** Trio passes, or a parent is missing / no genotype data.

**remove_male_parent, remove_female_parent** The trio fails and only one parent fails the single-parent homozygous check; that parent is set to 0 in `corrected_pedigree`.

**remove_both** The trio fails and both parents fail the homozygous check; both are set to 0.

**unresolved** The trio fails but neither parent fails the homozygous check (for example a recorded self whose offspring carries alleles the parent lacks). The pedigree is likely wrong but the faulty parent cannot be identified, so `corrected_pedigree` is left unchanged for manual review.

**keep_both** Only with low markers: the trio is within the threshold and both parents pass.

Low-marker trios get the same values prefixed with `low_markers_`. In the plot, every trio above the error threshold is coloured as a failure; unresolved trios are shown in grey.

## Value

An invisible named list with the following elements:

**pass** Trios that passed the Mendelian error threshold.

**fail** Trios that failed the Mendelian error threshold.

**low_markers** Trios with insufficient markers for evaluation.

**no_genotype_data** Trios absent from the genotype file.

**founders** Trios identified as founders.

**missing_parents** Trios with one or both parents coded as 0 (non-founders).

**full_results** Complete data.table with all trios and all output columns.

**corrected_pedigree** Pedigree table after applying recommended corrections.

**marker_summary** Per-marker Mendelian mismatch counts over every trio with both parents known and genotyped: `marker`, `trios_tested`, `trios_mismatch` and `mismatch_pct`, sorted worst first. Markers with a much higher rate than the rest (e.g. paralogous or miscalled markers) are candidates for removal before rerunning.

**marker_trio_table** If `marker_trio_table = TRUE`, a data.table with one row per marker (`marker`) and one column per trio with both parents known and genotyped, named `id_male_parent_female_parent`; otherwise NULL. See Details for the status values.

**plot** ggplot object if plot_results = TRUE, otherwise NULL.

## Author(s)

Josue Chinchilla-Vargas

## Examples

```
geno_df <- data.frame(
  id  = c("P1", "P2", "P3", "Off1", "Off2"),
  S1  = c(0L, 2L, 0L, 1L, 0L),
  S2  = c(2L, 0L, 2L, 1L, 2L),
  S3  = c(0L, 2L, 0L, 1L, 0L),
  S4  = c(2L, 0L, 2L, 1L, 2L),
  S5  = c(0L, 2L, 0L, 1L, 0L),
  S6  = c(2L, 0L, 2L, 1L, 2L),
  S7  = c(0L, 2L, 0L, 1L, 0L),
  S8  = c(2L, 0L, 2L, 1L, 2L),
  S9  = c(0L, 2L, 0L, 1L, 0L),
  S10 = c(2L, 0L, 2L, 1L, 2L)
)

ped_df <- data.frame(
  id            = c("Off1", "Off2"),
```

```
  male_parent   = c("P1",   "P1"),
  female_parent = c("P2",   "P3"),
  stringsAsFactors = FALSE
)

results <- validate_pedigree(
  pedigree_file  = ped_df,
  genotypes_file = geno_df,
  verbose        = FALSE,
  plot_results   = FALSE
)
print(results$full_results)
```

---

vcf_to_dosage *Convert a VCF File to a BIGpopA Genotype Matrix*

---

## Description

Reads a variant call format (VCF) file, or an existing vcfR object, and returns allele dosages in the layout used by the rest of BIGpopA: individuals in rows, markers in columns, coded as the dosage of the alternate (B) allele (0, 1, ..., ploidy) with NA for missing calls.

## Usage

```
vcf_to_dosage(
  vcf,
  ploidy = NULL,
  format = c("data.frame", "matrix"),
  dosage_field = NULL,
  ref_dosage = FALSE,
  biallelic_only = TRUE,
  marker_names = c("auto", "id", "position"),
  min_marker_call_rate = 0,
  min_sample_call_rate = 0,
  verbose = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| vcf | Path to a VCF file (.vcf or .vcf.gz), OR a vcfR object. |
| ploidy | Integer >= 2, or NULL (default) to infer the ploidy from the GT field. When supplied, it is checked against the ploidy observed in the file and a warning is issued if they disagree. |

| | |
|---|---|
| format | Character. "data.frame" (default) returns a data.table with an id column followed by one column per marker, the input expected by find_parentage() and validate_pedigree(). "matrix" returns a numeric matrix with individuals in named rows and markers in named columns, the input expected by allele_freq_poly() and solve_composition_poly(). |
| dosage_field | Character or NULL (default). Name of a FORMAT field holding dosages directly (for example "DS" or "UD"). When supplied, that field is read instead of counting alleles in GT, and ploidy must be given (it cannot be inferred from a dosage field). |
| ref_dosage | Logical. If TRUE, count reference alleles instead of alternate alleles (ploidy minus the alternate dosage). Default is FALSE, which matches the "dosage of allele B" coding used throughout BIGpopA. |
| biallelic_only | Logical. If TRUE (default), markers with more than one ALT allele are dropped. If FALSE, every non-reference allele is counted towards the dosage, so a multiallelic marker is collapsed to reference vs non-reference. |
| marker_names | Character. "auto" (default) uses the VCF ID column when it is complete and unique, and falls back to CHROM_POS otherwise. "id" forces the ID column and "position" forces CHROM_POS. |
| min_marker_call_rate | Numeric between 0 and 1. Markers called in fewer than this proportion of individuals are dropped. Default is 0 (no filtering). |
| min_sample_call_rate | Numeric between 0 and 1. Individuals called at fewer than this proportion of the retained markers are dropped. Applied after the marker filter. Default is 0 (no filtering). |
| verbose | Logical. If TRUE, prints a conversion report. Default is TRUE. |
| ... | Further arguments passed to vcfR::read.vcfR(), for example nrows to read only the first records of a large file. |

## Details

The function is ploidy agnostic. Dosage is obtained by counting the non-reference alleles in each GT call, so a diploid 0/1 returns 1 and a tetraploid 0/1/1/1 returns 3, with no assumption about the ploidy level. When ploidy is not supplied it is inferred from the number of allele slots in the calls themselves (the most common slot count across the file). Calls whose slot count disagrees with the file ploidy, for instance a diploid call in an otherwise tetraploid file, are set to missing and reported, since the downstream functions require a single ploidy level.

Phased (|) and unphased (/) separators are both accepted. A call is treated as missing when any of its allele slots is ..

vcfR is used to read the file and is listed under Suggests, so it must be installed before calling this function (install.packages("vcfR")).

## Value

A data.table with an id column followed by marker columns (format = "data.frame"), or a numeric matrix of individuals by markers (format = "matrix"). The attribute "ploidy" records the ploidy used.

## Author(s)

Josue Chinchilla-Vargas

## See Also

find_parentage(), validate_pedigree(), allele_freq_poly()

## Examples

```
if (requireNamespace("vcfR", quietly = TRUE)) {
  # Write a small tetraploid VCF (3 markers x 4 samples) to a temporary file
  vcf_path <- tempfile(fileext = ".vcf")
  writeLines(c(
    "##fileformat=VCFv4.3",
    "##FORMAT=<ID=GT,Number=1,Type=String,Description=\"Genotype\">",
    paste("#CHROM", "POS", "ID", "REF", "ALT", "QUAL", "FILTER", "INFO",
          "FORMAT", "S1", "S2", "S3", "S4", sep = "\t"),
    paste("chr1", "100", "snp1", "A", "T", ".", ".", ".", "GT",
          "0/0/0/0", "0/0/0/1", "0/1/1/1", "1/1/1/1", sep = "\t"),
    paste("chr1", "200", "snp2", "C", "G", ".", ".", ".", "GT",
          "0/0/1/1", "./././.", "0/0/0/0", "0/1/1/1", sep = "\t"),
    paste("chr1", "300", "snp3", "G", "A", ".", ".", ".", "GT",
          "1/1/1/1", "1/1/1/1", "0/0/1/1", "0/0/0/1", sep = "\t")
  ), vcf_path)

  # tetraploid VCF -> id + marker columns for find_parentage()
  genotypes <- vcf_to_dosage(vcf_path, ploidy = 4)
  print(genotypes)

  # same data as an individuals x markers matrix for allele_freq_poly()
  geno_mat <- vcf_to_dosage(vcf_path, ploidy = 4, format = "matrix")

  unlink(vcf_path)
}
```

# Index

allele_freq_poly, 2
allele_freq_poly(), *9–11*, *16, 17*
assign_composition_line, 3

check_ped, 4

find_parentage, 6
find_parentage(), *9, 10*, *16, 17*

ped_to_dosage, 9
ped_to_dosage(), *2*, *6*, *11, 12*

solve_composition_poly, 11
solve_composition_poly(), *3, 4*, *9*, *16*

validate_pedigree, 12
validate_pedigree(), *6*, *9*, *16, 17*
vcf_to_dosage, 15
vcf_to_dosage(), *2*, *6*, *10–12*
vcfR::read.vcfR(), *16*