

Package ‘BIGpopA’

August 21, 2026

Title Pedigree Validation Genetic Composition of Diploids & Polyploids

Version 2.0.0

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. 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

Depends R (>= 4.4.0)

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

Suggests covr, knitr (>= 1.10), rmarkdown, scales, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Repository CRAN

Date/Publication 2026-08-21 21:50:57 UTC

Contents

allele_freq_poly	2
check_ped	3
find_parentage	4
solve_composition_poly	7
validate_pedigree	8
Index	11

allele_freq_poly	<i>Compute Allele Frequencies for Populations</i>
------------------	---

Description

Computes allele frequencies for specified populations given SNP array data.

Usage

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

Arguments

- geno matrix of genotypes coded as the dosage of allele B (0, 1, 2, ..., ploidy) with individuals in rows (named) and SNPs in columns (named).
- 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).

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)

```

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 Cycles detected in the pedigree. 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
)
```

Arguments

genotypes_file	Path to a TSV/CSV/TXT file, OR a data.frame / data.table with an 'id' column followed by marker columns coded as 0, 1, 2.
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.
method	Character. One of "best_male_parent", "best_female_parent", "best_match", or "best_pair" (default).
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.

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.

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)

```

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,
  ped = NULL,
  groups = NULL,
  mia = FALSE,
  sire = FALSE,
  dam = FALSE,
  ploidy = 2
)

```

Arguments

Y	numeric matrix of genotypes (columns) from all animals (rows) in the population, coded as dosage of allele B (0, 1, 2, ..., ploidy).
X	numeric matrix of allele frequencies (rows) from each reference panel (columns). Frequencies are relative to allele B.
ped	data.frame giving pedigree information. Must be formatted with columns: ID, Sire, Dam.
groups	list of IDs categorized by breed/population. If specified, output will be a list of results categorized by breed/population.
mia	logical. Only applies if ped argument is supplied. If TRUE, returns a data.frame containing the inferred maternally inherited allele for each locus for each animal instead of breed composition results.
sire	logical. Only applies if ped argument is supplied. If TRUE, returns a data.frame containing sire genotypes for each locus for each animal instead of breed composition results.

dam	logical. Only applies if ped argument is supplied. If TRUE, returns a data.frame containing dam genotypes for each locus for each animal instead of breed composition results.
ploidy	integer. The ploidy level of the species (e.g., 2 for diploid, 3 for triploid).

Value

A data.frame, or a list of data.frames when groups is not NULL, containing breed/ancestry composition results.

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, suggests best-matching replacements, 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
)
```

Arguments

<code>pedigree_file</code>	Path to the pedigree file (TSV/CSV/TXT), OR a data.frame / data.table with columns: id, male_parent, female_parent.
<code>genotypes_file</code>	Path to the genotypes file (TSV/CSV/TXT), OR a data.frame / data.table with an id column followed by marker columns coded as 0, 1, 2.
<code>founders_file</code>	Character, optional. Path to a one-column file listing founder IDs. Founders with both parents coded as 0 are left unchanged. Defaults to NULL.
<code>trio_error_threshold</code>	Numeric. Maximum Mendelian error percentage to classify a trio as pass (default: 5.0). Must be between 0 and 100.
<code>min_markers</code>	Integer. Minimum non-missing markers required to evaluate a trio (default: 10).
<code>single_parent_error_threshold</code>	Numeric. Maximum homozygous-marker mismatch percentage for a parent to be considered acceptable (default: 2.0). Must be between 0 and 100.
<code>verbose</code>	Logical. If TRUE, prints progress, summary, and results to the console (default: TRUE).
<code>plot_results</code>	Logical. If TRUE, prints a histogram of trio Mendelian error percentages with a threshold line (default: TRUE).
<code>ploidy</code>	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.

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.
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)
```

Index

allele_freq_poly, [2](#)

check_ped, [3](#)

find_parentage, [4](#)

solve_composition_poly, [7](#)

validate_pedigree, [8](#)