Package 'hsphase'

February 17, 2026

Type Package

Title Phasing, Pedigree Reconstruction, Sire Imputation and Recombination Events Identification of Half-sib Families Using SNP Data

Version 3.0.0

Date 2026-2-15

Depends snowfall, R (>= 3.5.0)

LinkingTo RcppArmadillo (>= 0.4.300.8.0), Rcpp (>= 0.11.2)

Imports Rcpp, gdata

Description Identification of recombination events, haplotype reconstruction, sire imputation and pedigree reconstruction using half-sib family SNP data.

License GPL-3

NeedsCompilation yes

Repository CRAN

RoxygenNote 7.3.3

Author Mohammad Ferdosi [aut, cre], Cedric Gondro [aut]

Maintainer Mohammad Ferdosi <mhferdosi@yahoo.com>

Date/Publication 2026-02-17 10:50:02 UTC

Contents

`hsphase-package` *Phasing, Pedigree Reconstruction, Sire Imputation and Recombination Events for Half-sib Families*

Description

Identification of recombination events, haplotype reconstruction and sire imputation using half-sib family SNP data.

Details

Package:	hsphase
Type:	Package
Version:	3.0.0
Date:	2026-02-15
License:	GPL-3

Main functions:
bmh: Block partitioning
ssp: Sire inference
aio: Phasing
imageplot: Image plot of the block structure
rpoh: Reconstruct pedigree based on opposing homozygotes

Auxiliary functions:
hss: Half-sib family splitter
cs: Chromosome splitter
para: Parallel data analysis

Author(s)

Mohammad H. Ferdosi <mferdosi@une.udu.au>, Cedric Gondro <cgondro2@une.edu.au>
Maintainer: Mohammad H. Ferdosi <mhferdosi@yahoo.com>

References

Ferdosi, M. H., Kinghorn, B. P., van der Werf, J. H., & Gondro, C. (2013). Effect of genotype and pedigree error on detection of recombination events, sire imputation and haplotype inference using the hsphase algorithm. In *Proc. Assoc. Advmt. Anim. Breed. Genet* (Vol. 20, pp. 546–549). AAABG; Napier, New Zealand.

Ferdosi, M. H., Kinghorn, B. P., van der Werf, J. H. J., & Gondro, C. (2014). Detection of recombination events, haplotype reconstruction and imputation of sires using half-sib SNP genotypes. *Genetics Selection Evolution*, 46(1), 11.

Ferdosi, M. H., Kinghorn, B. P., van der Werf, J. H. J., Lee, S. H., & Gondro, C. (2014). hsphase: an R package for pedigree reconstruction, detection of recombination events, phasing and imputation of half-sib family groups. *BMC Bioinformatics*, 15(1), 172.

Ferdosi, M. H., & Boerner, V. (2014). A fast method for evaluating opposing homozygosity in large SNP data sets. *Livestock Science*.

Sahoo S., Ferdosi M.H., van der Werf J.H.J., and de las Heras-Saldana S., et al. (2025) *Proc. Assoc. Advmt. Anim. Breed. Genet*. 26: 323

Examples

```
genotype <- matrix(c(
  0,0,0,0,1,2,2,2,0,0,2,0,0,0,
  2,2,2,2,1,0,0,0,2,2,2,2,2,2,
  2,2,2,2,1,2,2,2,0,0,2,2,2,2,
  2,2,2,2,0,0,0,0,2,2,2,2,2,2,
  0,0,0,0,0,2,2,2,2,2,2,0,0,0
```

```
), ncol = 14, byrow = TRUE)

ssp(bmh(genotype), genotype)
aio(genotype)
imageplot(bmh(genotype), title = "ImagePlot example")
rplot(genotype, 1:14)
```

`.fastdist` *Calculate Genotypic Distances*

Description

Calculates a symmetric matrix of distances (canberra) between genotypes, based on a given genotype matrix. Each row in the 'GenotypeMatrix' represents a genotype, and each column represents a marker.

Usage

```
.fastdist(GenotypeMatrix)
```

Arguments

`GenotypeMatrix`	A matrix where each row represents a genotype and each column represents a marker. Genotypes should be coded as 0 for AA, 1 for AB, and 2 for BB, with 9 representing missing data.

Value

Returns a symmetric matrix of distances (canberra) between the genotypes specified in the 'GenotypeMatrix'. Row and column names of the returned matrix correspond to the row names of the 'GenotypeMatrix'.

Examples

```
# Simulate genotype data for 40 individuals across 1000 SNPs
genotypes <- .simulateHalfsib(numInd = 5, numSNP = 1000, recbound = 0:6, type = "genotype")
# Calculate the distance matrix
dist_matrix <- hsphase:::.fastdist(genotypes)
print(dist_matrix)
```

`.fixRotation` *Fix strand label rotation across consecutive block-structure columns*

Description

Internal helper to enforce a consistent strand-label orientation across adjacent columns of a block-structure matrix.

Usage

```
.fixRotation(blockStructure)
```

Arguments

`blockStructure`	A numeric matrix (typically individuals in rows, SNPs in columns) representing a block/strand structure. Values are expected to be small integers (commonly including ‘0‘, ‘1‘, ‘2‘ and possibly other internal codes).

Details

The input typically encodes sire strand-of-origin labels per individual (rows) and marker/SNP (columns), where ‘0‘ indicates unknown and non-zero values indicate an assigned strand/state. The native algorithm compares each column to the previous one and, when a "contrast" (swap of strand labels) increases agreement, it relabels the next column to reduce apparent strand-rotation between columns.

This function is a thin R wrapper around the native routine `fixRotation` implemented in C++ and called via `.Call()`.

At each step, the C++ code computes an agreement score between column `i` and column `i+1` using only positions where both columns are non-zero. It also computes the score after applying a contrast mapping to column `i+1` (conceptually swapping strand labels ‘1‘ and ‘2‘, leaving ‘0‘ unchanged). If the contrasted version agrees more with column `i`, the function relabels column `i+1`.

The relabeling performed by the native code is:

- ‘1 -> 3‘
- ‘2 -> 1‘
- ‘3 -> 2‘

leaving other values unchanged. (These codes are part of hsphase's internal block/strand encoding.)

Value

A numeric matrix with the same dimensions as `blockStructure`, where some entries in column `i+1` may be relabeled to improve consistency with column `i`. The transformation is applied iteratively from left to right across columns.

See Also

`bmh`, `ssp`, `aio` for creation and downstream usage of block structures.

.hblock *Build haplotype blocks from a BMH result matrix (native routine wrapper)*

Description

Internal wrapper around the native C routine `hblock`. It transforms a BMH result matrix into a block representation, with an optional maximum block size constraint.

Usage

```
.hblock(bmhResult, MaxBlock = 400)
```

Arguments

`bmhResult`	A numeric/integer matrix containing BMH results (block matching/haplotype-block intermediate output). Must be a matrix.
`MaxBlock`	Integer scalar. Maximum block size (default: 400).

Details

This function transposes and flattens `bmhResult` before passing it to compiled code via `.C`: `.C("hblock", ...)`.

Value

A matrix (same general shape as `bmhResult`) containing inferred block structure. Row and column names are propagated from `bmhResult` where available.

.maf *Calculate minor allele frequency (MAF)*

Description

Calculates the minor allele frequency (MAF) for a single SNP coded as: 0 = AA, 1 = AB, 2 = BB, and 9 = missing.

Usage

```
.maf(snp)
```

Arguments

`snp`	A numeric vector of genotypes for one SNP. Values must be 0, 1, 2, or 9 (missing).

Value

A single numeric value: the minor allele frequency (MAF).

Examples

```
snp_data <- c(0, 0, 1, 2, 2, 9)
.maf(snp_data)
```

`.o2tH` *Convert a one-row haplotype to a two-row haplotype*

Description

Converts a haplotype matrix where each individual is represented by one row and alleles are stored in alternating columns (1st allele, 2nd allele, ...) into a two-row-per-individual representation.

Usage

```
.o2tH(haplotype)
```

Arguments

`haplotype`	A haplotype object:

- a matrix with individuals in rows and allele columns in pairs (i.e. `ncol(haplotype)` must be even).

Details

Internally, any allele code of '2' is converted to '0' before conversion.

Value

An integer matrix in a two-row-per-individual format with `2 * nrow(haplotype)` rows and `ncol(haplotype) / 2` columns. Row names are interleaved using the original individual names.

.ptr2por *Convert a two-row haplotype per individual to a one-row representation*

Description

Converts a haplotype matrix where each individual is represented by two rows (allele 1 and allele 2) into a single-row-per-individual representation.

Usage

```
.ptr2por(haplotype)
```

Arguments

haplotype	A matrix containing haplotypes with two rows per individual.

Value

A matrix with one row per individual (allele 1 and allele 2 combined).

.simulateHalfsib *Simulate Half-Sibling Genotypes*

Description

This function simulates genotypes for a set of half-siblings based on specified parameters, including the number of individuals, the number of SNPs, recombination boundaries, and the type of data to return. It generates a sire genotype, maternal half-sib genotypes, and combines these to simulate offspring genotypes, optionally returning phased genotypes based on recombination events.

Usage

```
.simulateHalfsib(
  numInd = 40,
  numSNP = 10000,
  recbound = 0:6,
  type = "genotype"
)
```

Arguments

numInd	Integer, the number of half-siblings to simulate.
numSNP	Integer, the number of SNPs to simulate for each individual.
recbound	Numeric vector, specifying the range of possible recombination events to simulate.
type	Character string, specifying the type of data to return: "genotype" for genotypic data or any other string for phased genotypic data.

Value

Depending on the type parameter, this function returns a matrix of simulated genotypic data for half-siblings. If type is "genotype", it returns unphased genotypic data; otherwise, it returns phased genotypic data.

Examples

```
sim_genotypes <- .simulateHalfsib(numInd = 40, numSNP = 10000, recbound = 0:6, type = "genotype")
dim(sim_genotypes) # Should return 40 rows (individuals) and 100 columns (SNPs)
```

---

addSwitch *Add Switches*

Description

Add switch points to haplotypes by swapping the two haplotype rows for an individual from each switch point to the end of the chromosome.

Usage

```
addSwitch(haplotypeMatrix, switchPoints, minLength)
```

Arguments

haplotypeMatrix	matrix. Haplotypes for a half-sib family (**two rows per individual**).
switchPoints	list of integer/numeric vectors. Length must equal the number of individuals. Each element contains the switch positions (0-based/1-based depends on how they were produced; see Details). If there are no switches for an individual, use 0.
minLength	integer. Minimum distance between consecutive switch points. **Note:** in the current implementation this filter may not be enforced (depends on package version).

Details

Important: Each switch point causes a swap of the two haplotype rows for that individual from the switch position to the end.

The switchPoints list must have one element per individual (not per haplotype row). If an element is 0, no switch is applied for that individual.

If you rely on minLength to ignore nearby switches, verify your installed version enforces this rule.

Value

A matrix of the same dimension as haplotypeMatrix with switches applied.

See Also

groupMatSingle and fixSW

Examples

```
haplotype <- matrix(c(0, 0, 0, 0,
                      1, 1, 1, 1,
                      0, 0, 1, 1,
                      1, 1, 0, 0,
                      1, 1, 1, 1,
                      0, 0, 0, 0), byrow = TRUE, nrow = 6)

switchPoints <- list(firstInd = c(2), secondInd = c(1, 3), lastInd = 0)
addSwitch(haplotype, switchPoints, 0)
```

---

aio *All-in-one Phasing*

Description

Phasing of a single half-sib family group (single ordered chromosome).

Usage

```
aio(genotypeMatrix, bmh_forwardVectorSize = 30, bmh_excludeFP = TRUE,
    bmh_nsap = 3, bmh_fillMissing = FALSE, output = "phase")
```

Arguments

genotypeMatrix	matrix. Half-sib genotypes (one half-sib per row, SNPs ordered by map position in columns). Data must be numeric: 0, 1, 2 for AA, AB, BB. Use 9 for missing.
bmh_forwardVectorSize	integer. Number of heterozygous sites used to validate recombination events or detect genotyping/map errors.
bmh_excludeFP	logical. Exclude SNPs that may induce false heterozygous sites in the sire due to genotyping or map errors.
bmh_nsap	integer. Number of SNP per block to validate recombinations (e.g. 50K $\rightarrow$ 3, 700K $\rightarrow$ 10).
bmh_fillMissing	logical. If TRUE, recombination points are placed at the mid-point of the ambiguous interval rather than marking surrounding SNPs as missing.
output	character. If "phase", the function returns only the phased haplotypes matrix.

Details

This function calls bmh, ssp, and phf.

Value

If `output = "phase"`, returns a haplotype matrix with **two rows per individual** (first paternal, second maternal), coded as `0` (allele A), `1` (allele B), and `9` (missing/unphased).

Otherwise returns a list with elements:

- `phasedHalfsibs`
- `sireHaplotype`
- `blockStructure`

Note

Only this function needs to be called to phase a half-sib family. The genotype matrix must contain individuals from a single family and a single ordered chromosome.

See Also

`bmh`, `ssp`, `phf`

Examples

```
genotype <- matrix(c(       # Define a Half-sib Genotype Matrix
  2,1,0,            # Individual 1
  2,0,0,                    # Individual 2
  0,0,2                     # Individual 3
), byrow = TRUE, ncol = 3)  # There are 3 individulas with three SNPs

aio(genotype)          # The genotypes must include only one half-sib family and one chromosome
```

---

bmh *Block Partitioning*

Description

Identifies the block structure (chromosome segments) in a half-sib family that each individual inherited from its sire.

Usage

```
bmh(
  GenotypeMatrix,
  forwardVectorSize = 30,
  excludeFP = TRUE,
  nsap = 3,
  fillMissing = FALSE
)
```

Arguments

GenotypeMatrix	matrix. Half-sib genotypes (one half-sib per row; SNPs ordered by mapping position in the columns). Data should be numeric: 0, 1, 2 for AA, AB, BB. Use 9 for missing data.
forwardVectorSize	integer. Number of heterozygous sites used to validate recombination events or check for genotyping/map errors (50K → 30, 700K → 120).
excludeFP	logical. Exclude SNPs that may cause heterozygous sites in the sire due to genotyping errors or map errors.
nsap	integer. Number of SNP per block to validate recombinations (50K → 3, 700K → 10).
fillMissing	logical. Because the exact point of recombination is unknown, markers around recombination points may be set to missing. If TRUE, the recombination point is assumed to be in the middle of the ambiguous region, reducing missing markers.

Value

A matrix of block structure containing 1, 2, and 0. Values 1 and 2 represent the two sire strands (arbitrary labeling within a chromosome), and 0 indicates unknown origin.

Note

The genotype matrix must contain individuals from only one half-sib family and one ordered chromosome.

See Also

ssp, phf, aio, imageplot

Examples

```
genotype <- matrix(c(
  0,2,1,1,1,
  2,0,1,2,2,
  2,2,1,0,2,
  2,2,1,1,1,
  0,0,2,1,0
), ncol = 5, byrow = TRUE)

bmh(genotype)
```

co *Crossover Detection*

Description

Detects possible crossover segments by comparing pairs of individuals in a half-sib family.

Usage

```
co(genotypeMatrix)
```

Arguments

genotypeMatrix	matrix. Half-sib genotypes (one individual per row; SNPs in columns ordered by map position). Genotypes must be numeric: 0, 1, 2 for AA, AB, BB and 9 for missing.

Value

A matrix Returns a matrix with the number of crossover events for each site.

Examples

```
genotype <- matrix(c(
  2,1,0,
  2,0,2,
  0,0,2
), byrow = TRUE, ncol = 3)

co(genotype)
```

cs *Chromosome Splitter*

Description

Splits the genotype list generated by hss into chromosomes based on a map file/data.frame and orders SNPs by chromosomal position.

Usage

```
cs(halfsib, mapPath, separator = " ")
```

Arguments

halfsib	list. List of genotype matrices (one family per list item).
mapPath	character path to the map file (column 1 -> SNP names, column 2 -> chromosome name and column 3 -> SNP position in base pairs) or, alternatively, the name of a dataframe with the mapping information (in the same format)
separator	character. Field separator for the map file.

Details

The map file should include only the chromosomes that will be analyzed. For example, the Y and X chromosomes should be excluded (and others optionally). Names of each element in the list can be used for further categorization. The header must be "Name Chr Position".

Value

Returns a list of matrices, the number of elements in this list is the number of half-sib families multiplied by the number of chromosomes.

Examples

```
# Please run demo(hsphase)
```

---

fixSW *Fixing Switch Errors*

Description

Fix switch errors in haplotypes for a half-sib family.

Usage

```
fixSW(haplotype, ohMax = 0, windowsSize = 100, minLength = 100, cpus = 2)
```

Arguments

haplotype	matrix. Haplotypes for a half-sib family (two rows per individual).
ohMax	integer. Maximum tolerated opposing homozygotes when grouping each partition (increase if genotyping errors exist).
windowsSize	integer. Partition size (number of SNPs).
minLength	integer. Minimum length between switches.
cpus	integer. Number of CPU threads.

Value

A haplotype matrix with switch errors corrected.

See Also

groupMatSingle and addSwitch

Examples

```
haplotype <- .simulateHalfsib(7, 2500, type = "haplotype")$phased
switches <- list(500,0,0,1200,c(1000,2000),500,1200)

haplotype2 <- addSwitch(haplotype, switches, 0)

gMat <- groupMatSingle(haplotype2, 100, 2, "haplotype")
imageplot(gMat, title = "Before fixing switches")

haplotype3 <- fixSW(haplotype2, 0, 100, 100)

gMat2 <- groupMatSingle(haplotype3, 100, 2, "haplotype")
imageplot(gMat2, title = "After fixing switches")
```

---

genotypes *Example Genotype Data Set*

Description

An example genotype matrix for the **hsphase** package.

Usage

```
data(genotypes)
```

Format

A genotype `matrix` with:

- **Columns:** SNPs
- **Rows:** animals/individuals

Genotype coding follows the package conventions (typically `0`, `1`, `2` and `9` for missing).

References

Sahoo S., Ferdosi M.H., van der Werf J.H.J., and de las Heras-Saldana S., et al. (2025) *Proc. Assoc. Advmt. Anim. Breed. Genet*. 26: 323

groupMatSingle *Grouping a Half-sib Family*

Description

Group the genotype or haplotype of a half-sib family into partitions using opposing homozygotes.

Usage

```
groupMatSingle(haplotype, windowsSize, cpus = 2, input = "haplotype", oh = 0)
```

Arguments

haplotype	matrix. Haplotypes (two rows per individual) or genotypes (one row per individual) depending on input.
windowsSize	integer. Partition size.
cpus	integer. Number of CPU threads.
input	character. Either "haplotype" or "genotype".
oh	integer. Threshold for opposing homozygotes used for grouping (increase if genotyping errors exist).

Value

A grouping matrix.

See Also

addSwitch and fixSW

Examples

```
haplotype <- .simulateHalfsib(10, 5000, type = "haplotype")$phased
gMat <- groupMatSingle(haplotype, 100, 2, "haplotype")
imageplot(gMat)
```

hbp *Haplotype Blocks of Phased Data*

Description

Creates a block-structure matrix for a half-sib family based on phased data of the sire and the half-sib family.

Usage

```
hbp(PhasedGenotypeMatrix, PhasedSireGenotype, strand = "auto")
```

Arguments

`PhasedGenotypeMatrix`	`matrix`. Haplotypes for a half-sib family (**two rows per individual**). Alleles should be coded as `0` and `1`; use `9` for missing/unphased if present.
`PhasedSireGenotype`	`matrix`. Haplotypes of the sire (two rows; same SNP order as `PhasedGenotypeMatrix`).
`strand`	`character`. Method for identification of paternal strand. Use `"auto"` (recommended; default) or specify `"1"` or `"2"` to force a strand definition.

Value

A `matrix` in which `3` or `4` indicates the SNP originates from, respectively, sire strand 1 or strand 2. `0` indicates the origin is unknown.

Note

The input matrices must contain individuals from a single half-sib family and a single ordered chromosome. The SNP order must match between inputs.

See Also

aio, ssp

Examples

```
sire <- matrix(c(
  0,0,0,0,0,1,                  # Haplotype one of the sire
  0,1,1,1,1,0                   # Haplotype two of the sire
  ), byrow = TRUE, ncol = 6)

haplotypeHalfsib <- matrix(c(
  1,0,1,1,1,1,                  # Individual one, haplotype one
  0,1,0,0,0,0,                  # Individual one, haplotype two
  0,1,1,0,1,1,                  # Individual two, haplotype one
  1,0,0,1,0,0                   # Individual two, haplotype two
```

```
), byrow = TRUE, ncol = 6)    # 0s and 1s are alelle a and b

hbp(haplotypeHalfsib, sire)
```

---

hh *Heatmap of Half-sibs*

Description

Creates a heatmap of a half-sib dataset using an opposing-homozygotes (OH) matrix, with optional sidebars showing inferred and/or real pedigree groupings.

Usage

```
hh(oh, inferredPedigree, realPedigree, pedOnly = TRUE)
```

Arguments

oh	matrix. Opposing-homozygotes matrix (e.g. output of ohg).
inferredPedigree	matrix. Inferred pedigree (e.g. output of rpoh).
realPedigree	matrix. Original pedigree.
pedOnly	logical. If TRUE, consider only individuals that exist in the real pedigree.

Value

Returns a heatmap of the OH matrix with sidebars color-coded by sire groups from the inferred and original pedigrees (where provided).

Author(s)

The function uses colors generated by the *getcol* function in the *made4* package (Aedin Culhane).

See Also

ohg and rpoh

Examples

```
c1h1 <- .simulateHalfsib(numInd = 62, numSNP = 5000)
c1h2 <- .simulateHalfsib(numInd = 38, numSNP = 5000)
Genotype <- rbind(c1h1, c1h2)

oh <- ohg(Genotype)
hh(oh)
```

hss *Half-sib Family Splitter*

Description

Splits the dataset into half-sib family groups based on a pedigree.

Usage

```
hss(pedigree, genotype, minHS = 4, check = TRUE)
```

Arguments

pedigree	matrix the pedigree matrix should contain at least two columns, the first column with the half-sib IDs and the second column with the sires IDs
genotype	matrix genotype matrix with SNP ordered by mapping position in the columns. Data should be numeric. Use 0, 1 and 2 respectively for AA, AB and BB. Use 9 for missing data
minHS	integerMinimum number of offspring in a half-sib family
check	logical check the genotype file for the possible errors

Details

Only half-sib groups that have more than 3 individuals will be returned.

Value

Returns a list of numeric matrices, each matrix is a half-sib family.

Note

Pedigree must have at least two columns with sample ids (Column 1) and sire ids (Column 2).

Examples

```
# Please run demo(hsphase)
```

`imageplot` *Image Plot of Blocking Structure*

Description

Create an image plot of the blocking structure.

Usage

```
imageplot(x, title = c(), rv = FALSE, ...)
```

Arguments

`x`	`matrix`. Blocking structure (output of `bmh` or `hbp`).
`title`	`character` (or `NULL`). Title of the image plot.
`rv`	`logical`. If `TRUE`, reverse the colour scheme.
`...`	Optional graphical parameters.

Details

White indicates regions of unknown origin; red and blue correspond to the two sire strands.

Author(s)

This is a modified version of a function written by Chris Seidel, available at `http://www.phaget4.org/R/image_matrix.html`.

See Also

`bmh`, `aio`

Examples

```
genotype <- matrix(c(
  0,2,1,1,1,
  2,0,1,2,2,
  2,2,1,0,2,
  2,2,1,1,1,
  0,0,2,1,0
), ncol = 5, byrow = TRUE)

imageplot(bmh(genotype))
```

impute *Impute of Low Density SNP Marker to High Density (Paternal Strand)*

Description

Impute the paternal strand from low density to high density utilising high density sire haplotype.

Usage

```
impute(halfsib_genotype_ld, sire_hd, bmh_forwardVectorSize = 30,
bmh_excludeFP = TRUE, bmh_nsap = 3)
```

Arguments

halfsib_genotype_ld	matrix half-sib genotypes with low density marker (one half-sib per row, with SNP ordered by mapping position in the columns. Data should be numeric. Use 0, 1 and 2 respectively for AA, AB and BB. Use 9 for missing data)
sire_hd	matrix haplotype of sire (this parameter can be sequence data or any phased sire - the matrix should have rownames which are the sample IDs and colnames which are the SNP names)
bmh_forwardVectorSize	integer number of heterozygous sites used to validate recombination events or check for genotyping errors
bmh_excludeFP	logical exclude SNPs that may cause heterozygous sites in the sire due to genotyping errors or map errors
bmh_nsap	integer number of SNPs per block

Value

Return an imputed half-sib matrix.

See Also

bmh, ssp and phf

map *Example Map File*

Description

An example map dataset for the **hsphase** package.

Usage

```
data(map)
```

Format

A `data.frame` with the following columns:

Name SNP identifier

Chr Chromosome

Position SNP position in base pairs

References

Sahoo S., Ferdosi M.H., van der Werf J.H.J., and de las Heras-Saldana S., et al. (2025) *Proc. Assoc. Advmt. Anim. Breed. Genet*. 26: 323

ohd *Opposing Homozygote Detection*

Description

Counts, for each animal, the number of loci where it contributes opposing homozygotes in sites that imply heterozygosity in the sire.

Usage

```
ohd(genotypeMatrix, unique_check = FALSE, SNPs = 6000)
```

Arguments

`genotypeMatrix`	`matrix`. Half-sib genotypes (one half-sib per row, SNPs ordered by mapping position in the columns). Data should be numeric: `0`, `1`, `2` for `AA`, `AB`, `BB` and `9` for missing.
`unique_check`	`logical`. If `TRUE`, counts opposing homozygotes using a uniqueness rule (see Details).
`SNPs`	`integer`. Number of SNPs to use. (Only applicable if supported by the installed version.)

Value

A numeric vector with the number of heterozygous sites that each sample caused.

Note

This function can be used to identify pedigree errors; i.e., outliers with unusually high values.

Author(s)

This method was suggested by Bruce Tier <btier@une.edu.au> to identify pedigree errors.

Examples

```
genotype <- matrix(c(
  2,1,0,
  2,0,0,
  0,0,2
), byrow = TRUE, ncol = 3)

ohd(genotype)
```

---

ohg *Matrix of Opposing Homozygotes*

Description

Creates a matrix of pairwise opposing-homozygote (OH) counts from a genotype matrix.

Usage

```
ohg(genotypeMatrix)
```

Arguments

`genotypeMatrix` `matrix`. Genotypes (numeric): `0`, `1`, `2` for `AA`, `AB`, `BB` and `9` for missing.

Value

Returns a square matrix (sample $\times$ sample) of pairwise counts of opposing homozygotes. (Some versions may return this matrix inside a named list element.)

Note

This function can be slow for large datasets.

Author(s)

Ferdosi, M. H., & Boerner, V. (2014). A fast method for evaluating opposing homozygosity in large SNP data sets. Livestock Science.

See Also

rpoh

Examples

```
genotype <- matrix(c(
  2,1,0,
  2,0,0,
  0,0,2
), byrow = TRUE, ncol = 3)

ohg(genotype)
```

---

ohplot *Opposing Homozygotes Plot*

Description

Plot the sorted vectorized matrix of Opposing Homozygotes.

Usage

```
ohplot(oh, genotype, pedigree, check = FALSE)
```

Arguments

oh	integer Opposing homozygotes matrix (Output of ohg)
genotype	matrix genotype of one chromosome (data should be numeric. Use 0, 1 and 2 for respectively AA, AB and BB. Use 9 for missing data)
pedigree	matrix the pedigree matrix should contain at least two columns, the first column with the half-sib IDs and the second column with the sires IDs. This argument is optional.
check	logical check the genotype file for the possible errors

Details

The cut off line shows the edge of most different groups.

See Also

ohg and rpoh

Examples

```
set.seed(100)
chr <- list()
sire <- list()
set.seed(1)
chr <- list()
for(i in 1:5)
{
chr[[i]] <- .simulateHalfsib(numInd = 20, numSNP = 5000, recbound = 1:10)
sire[[i]] <- ssp(bmh(chr[[i]]), chr[[i]])
sire[[i]] <- sire[[i]][1,] + sire[[i]][2,]
sire[[i]][sire[[i]] == 18] <- 9
}

Genotype <- do.call(rbind, chr)
rownames(Genotype) <- 6:(nrow(Genotype) + 5)
sire <- do.call(rbind, sire)
rownames(sire) <- 1:5
Genotype <- rbind(sire, Genotype)
oh <- ohg(Genotype)  # creating the Opposing Homozygote matrix
pedigree <- as.matrix(data.frame(c(1:5, 6:(nrow(Genotype))),
rep = c(rep(0,5), rep(1:5, rep(20,5)))))
ohplot(oh, Genotype, pedigree, check = TRUE)
```

`para` *Parallel Analysis of Data*

Description

This function uses the list of matrices (the output of `cs`) and runs one of the options, on each element of the list, in parallel.

Usage

```
para(halfsibs, cpus = 1, option = "bmh", type = "SOCK", bmh_forwardVectorSize = 30,
bmh_excludeFP = TRUE, bmh_nsap = 3,  bmh_fillMissing = FALSE,pmMethod = "constant")
```

Arguments

`halfsibs`	`list` list of matrices of half-sibs (can be generated with `hss` and `cs` functions)
`cpus`	`numeric` number of CPUs (thread)
`option`	`character` type of analysis
`type`	`character` type of cluster for parallel analysis
`bmh_forwardVectorSize`	`integer` number of heterozygous sites used to validate recombination events or check for genotyping errors

bmh_excludeFP	logical exclude SNPs that may cause heterozygous sites in the sire due to genotyping errors or map errors
bmh_nsap	integer number of SNPs per block
bmh_fillMissing	logical Because the exact point of the recombinations is unknown, the markers around the recombination points are considered missing. By setting this argument to true, the recombination point is assumed to be in the middle unknown point, so no missing SNPs at the recombination point would be considered.
pmMethod	character method for creating the recombination matrix

Details

Type of analysis can be bmh, ssp, aio, pm, or rec (refer to pm, rplot and vignette for more information about rec).

Value

Returns a list of matrices with the results (formats specific to the option selected).

Examples

```
# Please run demo(hsphase)
```

---

pedigree *Example Pedigree*

Description

An example pedigree dataset for the **hsphase** package.

Usage

```
data(pedigree)
```

Format

A data.frame with the following columns:

First column Half-sib (offspring) IDs

Second column Sire IDs

References

Sahoo S., Ferdosi M.H., van der Werf J.H.J., and de las Heras-Saldana S., et al. (2025) *Proc. Assoc. Advmt. Anim. Breed. Genet*. 26: 323

pedigreeNaming *Fix Pedigree Errors*

Description

Tries to link the inferred pedigree from rpoh with sire IDs in the original pedigree and fix pedigree errors.

Usage

```
pedigreeNaming(inferredPedigree, realPedigree)
```

Arguments

inferredPedigree	matrix. Inferred pedigree (output of rpoh).
realPedigree	matrix. Original pedigree.

Details

This function calls bmh and recombinations to count the number of recombinations in each half-sib group.

Value

Returns the inferred pedigree with the best match to sire names used in the original pedigree file.

See Also

rpoh and ohg

Examples

```
# Please run demo(hsphase)
```

phf *Half-sib Family Phasing*

Description

Phases a half-sib family using the block structure and an imputed sire haplotype matrix.

Usage

```
phf(GenotypeMatrix, blockMatrix, sirePhasedMatrix)
```

Arguments

GenotypeMatrix	matrix. Half-sib genotypes (one half-sib per row; SNPs ordered by mapping position in the columns). Data should be numeric: 0, 1, 2 for AA, AB, BB. Use 9 for missing data.
blockMatrix	matrix. Blocking structure (output of bmh).
sirePhasedMatrix	matrix. Imputed sire haplotypes (output of ssp).

Value

Returns a matrix containing the phased parental haplotypes of the half-sibs (**two rows per individual**). Alleles are coded as 0 (A), 1 (B), and 9 (missing/unphased).

Note

The genotype matrix must contain individuals from one half-sib family and one ordered chromosome. This function is used by aio for complete phasing of a half-sib group.

See Also

aio

Examples

```
genotype <- matrix(c(
  2,1,0,
  2,0,0,
  0,0,2
), byrow = TRUE, ncol = 3)

block <- bmh(genotype)
phf(genotype, block, ssp(block, genotype))
```

---

pm *Probability Matrix*

Description

Creates a recombination (probability) matrix based on the blocking structure.

Usage

```
pm(blockMatrix, method = "constant")
```

Arguments

blockMatrix	matrix. Blocking structure (output of bmh).
method	character. Method for creating the recombination matrix. Typically "constant" or "relative".

Details

This function identifies recombination between two consecutive sites and marks recombination sites with 1. If there are unknown sites between two blocks, it marks these sites with:

- 1 for the "constant" method, or
- $1/m$ for the "relative" method, where m is the number of unknown sites.

Examples

```
genotype <- matrix(c(
  0,2,0,1,0,
  2,0,1,2,2,
  2,2,1,0,2,
  2,2,1,1,1,
  0,0,2,1,0
), ncol = 5, byrow = TRUE)

block <- bmh(genotype)
pm(block)
```

---

pogc *Parent-Offspring Group Constructor*

Description

Assign offspring to parents based on an opposing-homozygotes (OH) matrix.

Usage

```
pogc(oh, genotypeError)
```

Arguments

oh	matrix. Opposing homozygotes matrix (output of ohg).
genotypeError	integer. Number of genotyping errors allowed when interpreting the oh matrix.

Value

A data.frame with two columns:

- animal ID
- assigned parent ID

See Also

ohg, hss, rpoh

Examples

```
set.seed(1)
chr <- list()
sire <- list()

for(i in 1:5)
{
  chr[[i]] <- .simulateHalfsib(numInd = 20, numSNP = 5000, recbound = 1:10)
  sire[[i]] <- ssp(bmh(chr[[i]]), chr[[i]])
  sire[[i]] <- sire[[i]][1,] + sire[[i]][2,]
  sire[[i]][sire[[i]] == 18] <- 9
}

Genotype <- do.call(rbind, chr)
rownames(Genotype) <- 6:(nrow(Genotype) + 5)
sire <- do.call(rbind, sire)
rownames(sire) <- 1:5
Genotype <- rbind(sire, Genotype)

oh <- ohg(Genotype)
pogc(oh, 5)
```

---

readGenotype *Read and Check the Genotype File*

Description

Reads a genotype file and optionally checks it for common formatting/data issues.

Usage

```
readGenotype(genotypePath, separatorGenotype = " ", check = TRUE)
```

Arguments

genotypePath	character. Path to the genotype file (animals in rows and SNPs in columns). SNPs should be coded as 0, 1, 2 for AA, AB, BB. Use 9 for missing data. Please refer to the vignette for more information.
separatorGenotype	character. Field separator used in the genotype file.
check	logical. If TRUE, check the genotype file for possible errors.

Value

A genotype matrix.

Note

Please refer to the vignette for more information.

recombinations *Recombination Number*

Description

Counts the number of recombinations for each individual based on the block structure.

Usage

```
recombinations(blockMatrix)
```

Arguments

blockMatrix	matrix. Block structure (output of bmh).

Value

A numeric vector of recombination counts with length equal to the number of individuals (rows) in blockMatrix.

See Also

bmh

Examples

```
genotype <- matrix(c(
  2,1,0,0,
  2,0,2,2,
  0,0,2,2,
  0,2,0,0
), byrow = TRUE, ncol = 4)

recombinations(bmh(genotype))
```

rplot *Recombination Plot*

Description

Creates a plot showing the sum of recombination events across a half-sib family.

Usage

```
rplot(x, distance, start = 1, end = ncol(x), maximum = 100,
      overwrite = FALSE, method = "constant")
```

Arguments

x	matrix. Half-sib genotypes (one half-sib per row; SNPs ordered by mapping position in columns). Numeric coding: 0, 1, 2 for AA, AB, BB. Use 9 for missing data.
distance	numeric (or integer). Physical distances between markers (length must match ncol(x) or the plotted range).
start	integer. First marker index for the plot.
end	integer. Last marker index for the plot.
maximum	integer. Maximum number of recombinations to show (higher recombination rates will be omitted).
overwrite	logical. Draw over the current plot (default FALSE).
method	character. Method passed to pm (e.g., "constant" or "relative").

Examples

```
genotype <- matrix(c(
  0,2,0,1,0,
  2,0,1,2,2,
  2,2,1,0,2,
  2,2,1,1,1,
  0,0,2,1,0
), ncol = 5, byrow = TRUE)

rplot(genotype, c(1,2,3,4,8))
```

---

rpoh *Reconstruct Pedigree Based on Matrix of Opposing Homozygotes*

Description

Reconstructs a half-sib pedigree based on a matrix of opposing homozygotes.

Usage

```
rpoh(genotypeMatrix, oh, forwardVectorSize = 30, excludeFP = TRUE, nsap = 3,
maxRec = 15, intercept = 26.3415, coefficient = 77.3171, snpnooh, method, maxsnpnooh)
```

Arguments

genotypeMatrix	matrix genotype of one chromosome (data should be numeric. Use 0, 1 and 2 for respectively AA, AB and BB. Use 9 for missing data)
oh	integer Opposing homozygotes matrix (Output of ohg)
forwardVectorSize	integer number of heterozygous sites used to validate recombination events or check for genotyping errors

excludeFP	logical excludes SNPs that may cause heterozygous sites in the sire due to genotyping errors or map errors
nsap	integer number of SNP per block to validate recombinations
maxRec	integer maximum number of expected recombinations per individual
intercept	integer intercept of fitted model
coefficient	integer coefficient of fitted model
snpnooh	integer number of SNPs used to create *oh* matrix (this number must be divided by 1000)
method	character pedigree reconstruction method
maxsnpnooh	numeric the maximum number of allowing opposing homozygote in a half-sib family

Details

Four methods *simple*, *recombinations*, *calus* and *manual* can be utilized to reconstruct the pedigree.

The following examples show the arguments require for each method.

```
pedigree1 <- rpoh(oh = oh, snpnooh = 732, method = "simple")
pedigree2 <- rpoh(genotypeMatrix = genotypeChr1, oh = ohg(genotype), maxRec = 10 , method =
"recombinations")
pedigree3 <- rpoh(genotypeMatrix = genotype, oh = oh, method = "calus")
pedigree4 <- rpoh(oh = oh, maxsnpnooh = 31662, method = "manual")
```

Value

Returns a data frame with two columns, the first column is animals' ID and the second column is sire identifiers (randomly generated).

Note

Method can be *recombinations*, *simple*, *calus* or *manual*. Please refer to vignette for more information.

The sire genotype should be removed before using this function utilizing pogc function.

See Also

bmh and recombinations

Examples

```
# Please run demo(hsphase)
```

ssp *Sire Imputation and Phasing*

Description

Infers (imputes) and phases the sire haplotypes based on the block structure matrix and homozygous sites of the half-sib genotype matrix.

Usage

```
ssp(blockMatrix, genotypeMatrix)
```

Arguments

`blockMatrix`	`matrix`. Block structure (output of `bmh`).
`genotypeMatrix`	`matrix`. Half-sib genotypes (one individual per row). Genotypes coded as `0`, `1`, `2` for `AA`, `AB`, `BB`. Use `9` for missing data.

Value

A `matrix` with two rows (one per sire haplotype) and columns corresponding to SNPs in genotype order. Alleles are coded as `0` (A) and `1` (B). Alleles that could not be imputed are coded as `9`.

See Also

`phf`, `aio`, `imageplot`

Examples

```
genotype <- matrix(c(
  0,2,1,1,1,
  2,0,1,2,2,
  2,2,1,0,2,
  2,2,1,1,1,
  0,0,2,1,0
), ncol = 5, byrow = TRUE)

ssp(bmh(genotype), genotype)
```

switchDetector *Switch Detector*

Description

Detect switch errors in the haplotypes of a half-sib family.

Usage

```
switchDetector(groupMatrix)
```

Arguments

groupMatrix	matrix. Group matrix generated by groupMatSingle.

Value

A list of integer vectors. The list length equals the number of individuals. Each vector contains the locations of detected switch errors for that individual.

See Also

groupMatSingle

Examples

```
haplotype <- .simulateHalfsib(8, 3000, type = "haplotype")$phased
switches <- list(2500,0,0,1200,c(1000,2000),500,2000,0)

haplotype2 <- addSwitch(haplotype, switches, 0)
gMat <- groupMatSingle(haplotype2, 100, 2, "haplotype")

switchDetector(gMat)
```

Index