Package: xoiGBS (via r-universe)

June 1, 2026

Version 0.1.8

Date 2026-03-12

Title Analyze Recombination and Crossover Interference with Sequence-Based Genotypes

Description Analyze recombination and crossover interference from genotyping-by-sequencing (GBS) data on a backcross.

Author Karl W Broman [aut, cre] (<https://orcid.org/0000-0002-4914-6671>)

Maintainer Karl W Broman <broman@wisc.edu>

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.7), R.utils, data.table, stats, parallel

Suggests testthat, devtools, roxygen2

License MIT + file LICENSE

URL https://github.com/kbroman/xoiGBS

BugReports https://github.com/kbroman/xoiGBS/issues

Encoding UTF-8

LinkingTo Rcpp

ByteCompile true

Roxygen list(markdown=TRUE)

Config/roxygen2/version 8.0.0

Repository https://kbroman.r-universe.dev

Date/Publication 2026-06-01 16:48:41 UTC

RemoteUrl https://github.com/kbroman/xoiGBS

RemoteRef HEAD

RemoteSha 2fa9ef8caf9e5b6c082fda356a44b83bfb18630f

Contents

`calc_genoprob_gbs` *Calculate genotype probabilities from GBS allele counts*

Description

Calculate genotype probabilities using an HMM, from high-throughput sequencing data with counts of alleles at SNPs

Usage

```
calc_genoprob_gbs(
  counts,
  map,
  error_prob1 = 0.002,
  error_prob2 = 0.002,
  map_function = c("haldane", "kosambi", "c-f", "morgan"),
  cores = 1
)
```

Arguments

`counts`	Three-dimensional array of counts, positions x individuals x two alleles, with alleles A and B where we are expecting genotypes AA and AB
`map`	Vector of marker positions, same length as `nrow(counts)`, in cM
`error_prob1`	Error probability for sequencing errors
`error_prob2`	Error probability for locus errors
`map_function`	Map function for converting from cM distances to recombination fractions
`cores`	Number of CPU cores to use, for multi-core calculations

Value

Matrix of genotype probabilities, positions x individuals, for Pr(het)

grab2XO *Grab all double-crossover locations*

Description

Grab all double-crossover locations, as a matrix with two columns, for cases of exactly 2 crossovers

Usage

```
grab2XO(xoloc)
```

Arguments

xoloc	List of matrices, of crossover locations, as output from inferXOloc().

Value

Matrix with locations of pair of crossovers, when there are exactly two

grabXO *Grab all crossover locations*

Description

Grab all crossover locations, as a vector

Usage

```
grabXO(xoloc)
```

Arguments

xoloc	List of matrices, of crossover locations, as output from inferXOloc().

Value

Vector of crossover locations

inferXOloc *Infer crossover locations*

Description

Infer crossover locations from a matrix of genotype probabilities as produced by `calc_genoprob_gbs()`

Usage

```
inferXOloc(genoprob, map, low = 0.1, high = 0.9)
```

Arguments

genoprob	Matrix of genotype probabilities, positions x individuals, as produced by `calc_genoprob_gbs()`
map	Vector of marker positions, same length as `nrow(genoprob)`
low	Lower threshold; if probability below this threshold, infer homozygous
high	Higher threshold; if probaability above this threshold, infer heterozygous

Value

List of matrices (of length `ncol(genoprob)`, each having columns with estimated location and left and right interval endpoints. If no crossovers, it will be a matrix with no rows.

order_ids *Get numeric order of individual IDs*

Description

Get numeric order of individual IDs, when they're of the form blah25 or blah25-2 Can also have dup info at the end, like blah25_dup1 or blah25_dup2

Usage

```
order_ids(ids, decreasing = FALSE)
```

Arguments

ids	Vector of individual IDs (as character string)
decreasing	If TRUE, get decreasing order (largest to smallest)

Details

We assume the ids have an initial non-numerial label, followed by a number, followed possibly by _dup and then another number. The IDs are sorted first by the initial non-numeric bit, then by the number, then by the duplicate number (with absense of _dup taken to be duplicate "0" and _dup without a number taken to be duplicate "1").

Value

Input IDs, sorted numerically

See Also

`sort_ids()`

Examples

```
ids <- c("BCA70", "BCA1", "PWD1", "PWD2", "BCA2", "BCA75",
         "BCA70_dup", "PWD1_dup2", "PWD1_dup1")
order_ids(ids)
sort_ids(ids)
sort_ids(ids, decreasing=TRUE)
```

`randXOloc` *Randomizing estimated crossover locations.*

Description

Randomizing estimated crossover locations, uniform within their intervals.

Usage

```
randXOloc(xoloc)
```

Arguments

`xoloc`	Either a matrix, or a list of matrices, of crossover locations, as output from `inferXOloc()`.

Value

Object like that input, but with estimated crossover locations randomized.

read_counts *read counts data from a set of files*

Description

read counts data from a set of files

Usage

```
read_counts(files)
```

Arguments

files	A vector of character strings with the counts data. These text files can be gzipped. Each file should be one backcross individual, and the files should have six whitespace-delimited columns: chromosome, basepairs positions, allele1, allele2, readcount1, readcount2, with allele1 being the allele for the homozygote parent. Alternatively, it can be a single directory, in which case we read all the .txt or .gz files in that directory.

Details

Names are taken from the file names...everything before ".txt" or ".gz", but removing "_read_counts" if it's part of the name. So BCA81-2_read_counts.txt.gz becomes BCA81-2

Value

A list of data frames with the contents of the files

reorg_counts *reorganize count data*

Description

Reorganize count data, from by-individual data frames to a list of 3-dimensional arrays.

Usage

```
reorg_counts(counts, map = NULL, clean_chr = TRUE, quiet = TRUE)
```

Arguments

counts	A list of data frames, one per individual, with six columns: chromosome, basepairs position, allele1, allele2, readcount1, and readcount 2, as read with `read_counts()`.
map	Optional map of SNPs to consider in the output, as a list of vectors of marker positions. If provided, any other positions will be ignored.
clean_chr	If TRUE, remove "chr" from the chromosome names, so chr13 becomes just 13
quiet	If FALSE, print some tracing info

Value

A list containing `map` and `counts`, with the map being a list of vectors of Mbp positions, and the counts being a list of 3-dimensional arrays, position x individual x allele.

`sort_ids` *Sort individual IDs numerically*

Description

Sort individual IDs numerically, when they're of the form blah25 or blah25-2 Can also have dup info at the end, like blah25_dup1 or blah25_dup2

Usage

```
sort_ids(ids, decreasing = FALSE)
```

Arguments

`ids`	Vector of individual IDs (as character string)
`decreasing`	If TRUE, sort in decreasing order (largest to smallest)

Details

We assume the ids have an initial non-numerial label, followed by a number, followed possibly by _dup and then another number. The IDs are sorted first by the initial non-numeric bit, then by the number, then by the duplicate number (with absense of _dup taken to be duplicate "0" and _dup without a number taken to be duplicate "1").

Value

Input IDs, sorted numerically

See Also

`order_ids()`

Examples

```
ids <- c("BCA70", "BCA1", "PWD1", "PWD2", "BCA2", "BCA75",
         "BCA70_dup", "PWD1_dup2", "PWD1_dup1")
order_ids(ids)
sort_ids(ids)
sort_ids(ids, decreasing=TRUE)
```

`trimXO` *Trim tight double crossovers from crossover information*

Description

Trim tight double crossovers from crossover information

Usage

```
trimXO(xoloc, mind = 1)
```

Arguments

`xoloc`	Either a matrix, or a list of matrices, of crossover locations, as output from `inferXOloc()`.
`mind`	Minimum allowed distance between crossovers

Value

Object like that input, but with double-crossovers within `mind` of each other removed

Index