Package: inferRecom (via r-universe)

July 2, 2026

Title Homologous Recombination Detection in Family Pedigrees

Version 0.99.1

Description This package implements a pedigree-based algorithm to detect homologous recombination. Additional functions are supplied to detect runs of homozygosity and to phase haplotypes at informative SNPs.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

biocViews Genetics, SNP, GenomeAnnotation, Sequencing, Coverage

Depends R (>= 4.5.0), SummarizedExperiment, GenomicRanges, S4Vectors

Imports methods, utils, IRanges, snpStats, BiocParallel, dplyr, stats

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

BugReports https://github.com/catherinefayemahoney/inferRecom/issues

URL https://github.com/catherinefayemahoney/inferRecom

Config/testthat/edition 3

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-01-05 19:05:27 UTC

RemoteUrl https://github.com/BiocStaging/inferRecom

RemoteRef HEAD

RemoteSha 820b7fd7f8146d2e82eeac2cf308e455a78444df

Contents

`crossoverData` *Example Crossover Data*

Description

Pre-computed maternal and paternal crossover events from the simCEU dataset. Used for testing and demonstrating haplotype phasing functionality.

Format

GRanges objects stored as RDS files with metadata columns:

childId Child identifier

familyId Family identifier

startSnp SNP name at crossover interval start

finishSnp SNP name at crossover interval end

startCm Genetic position at interval start (cM)

finishCm Genetic position at interval end (cM)

Details

Two crossover datasets are provided:

- `xoMat.rds` - Maternal crossover events
- `xoPat.rds` - Paternal crossover events

These objects contain detected crossover events from 3-child families in the simCEU dataset. Crossovers were identified using:

```
# Maternal crossovers
xoMat <- xoDetect(
  plinkFile = "simCEU",
  mapFile = "female_chr4.txt",
  familySize = 3,
  parent = "mother",
  snpFilter = 5,
```

```
  cmFilter = 1
)

# Paternal crossovers
xoPat <- xoDetect(
  plinkFile = "simCEU",
  mapFile = "male_chr4.txt",
  familySize = 3,
  parent = "father",
  snpFilter = 5,
  cmFilter = 1
)
```

Access the crossover data using:

```
dataPath <- system.file("extdata", package = "inferRecom")
xoMat <- readRDS(file.path(dataPath, "xoMat.rds"))
xoPat <- readRDS(file.path(dataPath, "xoPat.rds"))
```

`geneticMaps-data` *Genetic Maps for Chromosome 4*

Description

Sex-specific and sex-averaged genetic maps for chromosome 4.

Format

Tab-delimited text file with columns:

chr Chromosome identifier (chr4)

pos Physical position in base pairs (hg19/GRCh37)

rate Recombination rate (cM/Mb) at this position

cM Cumulative genetic distance in centiMorgans from chromosome start

Access the files using:

```
femaleMap <- read.delim(file.path(dataPath, "female_chr4.txt"))
```

Source

Derived from European sex-specific maps from Bherer, et al. (2014).

References

Bhérer, C., Campbell, C. L., & Auton, A. (2017). Refined genetic maps reveal sexual dimorphism in human meiotic recombination at multiple scales. Nature communications, 8(1), 14994.

hzRun *Detect Runs of Homozygosity in PLINK Genotype Data*

Description

This function identifies runs of homozygosity (ROH) in PLINK genotype data based on minimum physical length and SNP count thresholds. It can optionally filter for specific samples, apply a minimum genetic distance criterion, and separate results by case/control status.

Usage

```
hzRun(
  plinkFile,
  mapFile = NULL,
  minMb = 1,
  minSnps = 5,
  minCm = NULL,
  rsOnly = FALSE,
  sampleIds = NULL,
  caseControl = FALSE,
  BPPARAM = SerialParam()
)
```

Arguments

`plinkFile`	Character string; path to the PLINK prefix (without extensions).
`mapFile`	Character string or NULL; path to a tab-delimited genetic map file containing columns: `chr`, `pos`, `cM`. If NULL, cM filtering is not available. Default is NULL.
`minMb`	Numeric; minimum physical length of ROH in megabases. Default is 1.
`minSnps`	Integer; minimum number of consecutive homozygous SNPs required for an ROH. Default is 5.
`minCm`	Numeric or NULL; minimum genetic length of ROH in centiMorgans. If NULL, no genetic length filtering is applied. Requires mapFile to be specified. Default is NULL.
`rsOnly`	Logical; whether to restrict analysis to SNPs with names beginning with "rs". Default is FALSE.
`sampleIds`	Character vector or NULL; specific sample IDs to analyze. If NULL, all samples are analyzed. Default is NULL.
`caseControl`	Logical; if TRUE, function returns a GRanges of case and control ROH. Requires affected status in FAM file. Default is FALSE.
`BPPARAM`	A BiocParallelParam object specifying parallel execution. Default is `SerialParam()` for serial execution. Use `MulticoreParam(workers = n)` for parallel processing across n cores.

Details

Runs of homozygosity (ROH) are continuous genomic segments where an individual is homozygous at all marker positions. ROH can indicate autozygosity (inheritance of identical haplotypes from a common ancestor) and are used to estimate inbreeding, identify disease-associated loci, and study population history.

This function identifies ROH by:

1. Converting genotypes to binary (1 = homozygous, 0 = heterozygous/missing)
2. Identifying runs using run-length encoding
3. Filtering runs by minimum SNP count and physical length
4. Optionally filtering by minimum genetic length (cM)

Genotypes from **snpStats** are coded as:

- 0 = homozygous reference (AA) - counted as homozygous
- 1 = heterozygous (AB) - breaks ROH
- 2 = homozygous alternate (BB) - counted as homozygous
- NA = missing - breaks ROH

Filtering strategy:

- Physical length (minMb): Always applied
- SNP count (minSnps): Always applied
- Genetic length (minCm): Applied only if mapFile is provided and minCm is not NULL

Parallel processing is performed by individual, improving efficiency for datasets with many samples.

Value

A GRanges object containing detected ROH with metadata columns: `sampleId`, `startSnp`, `finishSnp`, `numSnps`. If genetic map is provided, also includes `startCm`, `finishCm`. If `caseControl = TRUE`, returns a GRanges with two elements: `case` and `control`. Returns an empty GRanges if no ROH are detected.

Examples

```
# Serial execution (default)

# Get path to example data
dataPath <- system.file("extdata", package = "inferRecom")
plinkFile <- file.path(dataPath, "simCEU")

# Detect ROH
rohData <- hzRun(
    plinkFile = plinkFile,
    minMb = 1,
    minSnps = 50
)
```

```
# View results
head(rohData)

# Analyze specific individuals
sampleIds <- c("F3C1", "F4C3", "F5P2")
rohSubset <- hzRun(
    plinkFile = plinkFile,
    minMb = .5,
    minSnps = 50,
    sampleIds = sampleIds
)

rohSubset

# Separate by case/control status
rohCc <- hzRun(
    plinkFile = plinkFile,
    minMb = .5,
    minSnps = 50,
    caseControl = TRUE
)

rohCases <- rohCc$case
rohControls <- rohCc$control

rohCases
rohControls
```

`simCEU-data` *Example PLINK Dataset - simCEU*

Description

A simulated CEU (Utah residents with Northern and Western European ancestry) dataset for demonstrating crossover detection and ROH analysis.

Format

PLINK binary format files (.bed, .bim, .fam) containing:

Samples Simulated family trios and larger pedigrees with 2-3 children

SNPs Chromosome 4 markers

Genotypes Simulated genotype data with realistic LD structure

Families Mix of 2-child and 3-child families for testing

Details

The dataset includes three PLINK binary files:

- `simCEU.bed` - Binary genotype data
- `simCEU.bim` - Variant information (SNP IDs, positions, alleles)
- `simCEU.fam` - Sample information (family IDs, relationships)

Access the files using:

```
dataPath <- system.file("extdata", package = "inferRecom")
plinkFile <- file.path(dataPath, "simCEU")
```

Source

Simulated data based on CEU population structure from 1000 Genomes Project and R package `sim1000G`

References

Siva, N. (2008). 1000 Genomes project.

Dimitromanolakis, A., Xu, J., Krol, A., & Briollais, L. (2019). sim1000G: a user-friendly genetic variant simulator in R for unrelated individuals and family-based designs. BMC bioinformatics, 20(1), 26.

`xoDetect` *Detect Meiotic Recombinations from PLINK Genotype Data*

Description

This function identifies recombination events in family-based genotype data, using PLINK-formatted input files and a genetic map. It supports 2- or 3-child family structures, and can optionally write results to disk if an output path is provided.

Usage

```
xoDetect(
  plinkFile,
  mapFile,
  familySize = c(2, 3),
  parent = c("mother", "father"),
  rsOnly = FALSE,
  snpFilter = 5,
  cmFilter = 1,
  caseControl = FALSE,
  out = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

plinkFile	Character string; path to the PLINK prefix (without extensions).
mapFile	Character string; path to a tab-delimited map file containing columns: chr, pos, rate, cM.
familySize	Integer; either 2 or 3, specifying the family pedigree size.
parent	Character; either "mother" or "father", specifying the parent to trace crossovers from.
rsOnly	Logical; whether to restrict to SNPs with names beginning with "rs". Default is FALSE.
snpFilter	Integer; minimum number of SNPs separating putative crossovers. Default is 5.
cmFilter	Numeric; minimum genetic distance (cM) separating putative crossovers. Default is 1.
caseControl	Logical; if TRUE, function returns a GRangesList of case and control crossovers for 3+ child families. Default is FALSE.
out	Character or NULL; if provided, output is written as a CSV at this path.
BPPARAM	A BiocParallelParam object specifying parallel execution. Default is SerialParam() for serial execution. Use MulticoreParam(workers = n) for parallel processing across n cores.

Details

This function reads PLINK genotype data (via **snpStats**) and infers loci where meiotic recombination has occurred, resolved to the nearest informative SNPs. The algorithm identifies informative marker configurations where parents are heterozygous/homozygous and filters for Mendelian consistency before detecting inheritance state changes that indicate crossovers.

For 3-child families, children are analyzed in triples to allow identification of the specific recombinant child. For 2-child families, state changes between the two siblings are used to infer crossover locations, but the individual cannot be determined.

If out is provided, the resulting table is written to disk as UTF-8 encoded CSV without row names.

Parallel processing is performed by family, improving efficiency for datasets with many families.

Value

A GRanges listing detected crossover intervals with metadata columns: childId (in 3-child families), familyId, startSnp, finishSnp, startPos, finishPos, startCm, finishCm.

Examples

```
# Serial execution (default)

# Load example data
dataPath <- system.file("extdata", package = "inferRecom")
plinkFile <- file.path(dataPath, "simCEU")

# Basic maternal crossover detection
```

```
mapFemale <- file.path(dataPath, "female_chr4.txt")
xoMat3 <- xoDetect(
    plinkFile = plinkFile,
    mapFile = mapFemale,
    familySize = 3,
    parent = "mother"
)

# View results
xoMat3

# Basic 2-child family
xoMat2 <- xoDetect(
    plinkFile = plinkFile,
    mapFile = mapFemale,
    familySize = 2,
    parent = "mother"
)

# View results
xoMat2

# Maternal crossover detection with case/control separation
mapFemale <- file.path(dataPath, "female_chr4.txt")
xoMat3CC <- xoDetect(
    plinkFile = plinkFile,
    mapFile = mapFemale,
    familySize = 3,
    parent = "mother",
    caseControl = TRUE
)

# View results
xoMat3CC$case
xoMat3CC$control
```

---

xoPhase *Phase Haplotypes Using Crossover Information*

Description

This function phases parental and child haplotypes based on detected crossover events from maternal and paternal meioses.

Usage

```
xoPhase(
  plinkFile,
  xoDetectPaternal,
```

```
  xoDetectMaternal,
  famIds = NULL,
  rsOnly = TRUE,
  outputFormat = c("list", "summarizedExperiment", "vcf"),
  vcfOutput = "phased",
  BPPARAM = SerialParam()
)
```

Arguments

`plinkFile`	Character string; path to the PLINK prefix (without extensions).
`xoDetectPaternal`	GRanges; output from `xoDetect()` for paternal crossovers (parent = "father").
`xoDetectMaternal`	GRanges; output from `xoDetect()` for maternal crossovers (parent = "mother").
`famIds`	Character vector or NULL; specific family IDs to phase. If NULL, all families with 5+ members are analyzed. Default is NULL.
`rsOnly`	Logical; whether to restrict to SNPs with names beginning with "rs". Default is TRUE.
`outputFormat`	Character string; format for output. Options are "list" (default, returns named list of DataFrames), "summarizedExperiment" (returns SummarizedExperiment object), or "vcf" (writes VCF files).
`vcfOutput`	Character string; path prefix for VCF output files. Only used when outputFormat = "vcf". Each family will be written to a separate VCF file with suffix "_.vcf". Default is "phased".
`BPPARAM`	A BiocParallelParam object specifying parallel execution. Default is `SerialParam()` for serial execution. Use `MulticoreParam(workers = n)` for parallel processing across n cores.

Details

This function performs haplotype phasing using family-based genetic data and detected crossover events. The phasing process:

1. Identifies informative SNPs where parents are heterozygous/homozygous
2. Assigns alleles (1 or 2) to children based on inheritance
3. Converts allele codes to nucleotides (A, C, G, T)
4. Imputes homozygous sites from the other parent
5. Phases parental haplotypes using crossover breakpoints
6. Adds back non-informative SNPs with inferred genotypes where possible
7. Uses neighboring informative SNPs to infer phase for ambiguous cases
8. Outputs diploid genotypes for all family members across all SNPs

Non-informative SNPs are handled as follows:

- Both parents homozygous (same allele): Genotypes confidently inferred

- Both parents heterozygous: Phase uncertain, set to NA
- One parent heterozygous, one homozygous: Phase inferred from neighboring informative SNPs using haplotype continuity

The function requires crossover detection results from both parents as GRanges objects. Run `xoDetect()` separately for maternal and paternal crossovers before using this function.

Parallel processing is performed by family, improving efficiency for datasets with many families.

Output format: Each phased family contains columns:

- `rsID` - SNP identifier
- `location` - Physical position in base pairs
- For each child: `<childID>Pat`, `<childID>Mat` - paternal and maternal haplotypes
- For parents: `<parentID>_1`, `<parentID>_2` - two haplotypes

Value

Depends on outputFormat:

- "list": A named list of DataFrame objects, one per family. Each DataFrame contains phased haplotypes with columns for SNP ID, physical position, and phased alleles for each family member.
- "summarizedExperiment": A SummarizedExperiment object containing phased genotypes across all families with rowData (SNP information) and colData (sample information).
- "vcf": Writes VCF files and returns paths to created files.

Returns a message if no phase information is available for a family.

Examples

```
# Load example data
dataPath <- system.file("extdata", package = "inferRecom")
plinkFile <- file.path(dataPath, "simCEU")
mapFemale <- file.path(dataPath, "female_chr4.txt")
mapMale <- file.path(dataPath, "male_chr4.txt")

# Detect crossovers for both parents (returns GRanges objects)
xoPat <- xoDetect(
    plinkFile = plinkFile,
    mapFile = mapMale,
    familySize = 3,
    parent = "father"
)
xoMat <- xoDetect(
    plinkFile = plinkFile,
    mapFile = mapFemale,
    familySize = 3,
    parent = "mother"
)

# Phase haplotypes (serial execution, default)
```

```
phased <- xoPhase(
    plinkFile = plinkFile,
    xoDetectPaternal = xoPat,
    xoDetectMaternal = xoMat
)

# Access phased haplotypes for a specific family
family1Phase <- phased[[1]]
family1Phase

# Phase haplotypes for subset of families
phasedSubset <- xoPhase(
    plinkFile = plinkFile,
    xoDetectPaternal = xoPat,
    xoDetectMaternal = xoMat,
    famIds = c("F4", "F6")
)

phasedSubset
```

Index