Package: inferRecom (via r-universe)

November 25, 2026

Title Homologous Recombination Detection in Nuclear Families

Version 1.0.0

Description This package implements a nuclear family-based algorithm to detect homologous recombination. Using genotypes in PLINK binary format, recombination events are inferred from changes in the inheritance state of informative SNPs between siblings. Closely spaced crossovers that are unlikely to represent true recombination can be filtered out using genetic maps or a SNP-count filter. Additional functions are supplied to detect runs of homozygosity and to phase haplotypes at informative SNPs and intervals that are inferred identical-by-descent. Recombination intervals and runs of homozygosity are returned as `GRanges` objects, and phased genotypes are returned as a `VariantAnnotation` `VCF` object, for use with other Bioconductor packages and genome browsers.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

VignetteBuilder knitr

biocViews Genetics, SNP, GenomeAnnotation, Sequencing, Coverage, DataImport

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

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

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
make libbz2-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 23:28:11 UTC

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

RemoteRef RELEASE_3_24

RemoteSha dd81292fad59e2067b6dbe74d047eb5537cbbf61

Contents

crossoverData 2
geneticMaps-data 4
hzRun 4
simCEU-data 7
xoDetect 8
xoPhase 10

Index **14**

`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 the families with three or more children 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"))
```

Value

No R object is returned. Access files via `system.file("extdata", package = "inferRecom")`.

Source

Created by `inst/scripts/make-crossovers.R`.

geneticMaps-data *Genetic Maps for Chromosome 4*

Description

Female, male and sex-averaged genetic maps for chromosome 4 (`female_chr4.txt`, `male_chr4.txt`, `sexavg_chr4.txt`).

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"))
```

Value

No R object is returned. Access files via `system.file("extdata", package = "inferRecom")`.

Source

The chromosome 4 files of the refined European maps of Bherer, et al. (2017), unchanged; see `inst/scripts/make-genetic-maps.R`.

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 GRangesList of case and control ROH, using the affected status in the FAM file with PLINK coding: 2 = affected (case); 1 = unaffected or missing (control). 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 GRangesList with two elements: `case` and `control`. Returns an empty GRanges if no ROH are detected. Chromosome names use the UCSC style (e.g. `chr4`).

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

Simulated pedigrees based on HapMap CEU (Utah residents with Northern and Western European ancestry) genotypes, for demonstrating crossover detection, phasing and ROH analysis. These data are for package development and examples only and do not reflect any biological principle: recombination was simulated with a uniform map of 100 cM across the region, far above real recombination rates, so that the short region contains crossovers.

Format

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

Families 8 nuclear families, 40 individuals: three families with 2 children, three with 3, one with 4 and one with 5

SNPs 1162 SNPs on chromosome 4 between 78.0 and 79.3 Mb (GRCh37)

Genotypes Founder haplotypes sampled from HapMap CEU genotypes, with recombination simulated in each meiosis

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, sex and affected status). One child per family is affected (PLINK phenotype 2); everyone else is unaffected (1).

Access the files using:

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

Value

No R object is returned. Access files via `system.file("extdata", package = "inferRecom")`.

Source

Simulated with the R package `sim1000G` from HapMap CEU genotypes (2010 release); see `inst/scripts/make-simCEU.R` and `inst/scripts/README.md`.

References

The International HapMap 3 Consortium (2010). Integrating common and rare genetic variation in diverse human populations. Nature, 467, 52-58.

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 informative SNPs separating two putative crossovers in the same child. A crossover that is reversed within fewer SNPs, typically because of genotyping error, is removed together with its reversal. 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, using the affected status of each child in the FAM file with PLINK coding: 2 = affected (case); 1 = unaffected or missing (control). 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. Chromosome names use the UCSC style (e.g. chr4).

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,
  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.
`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. Outputs diploid genotypes for all family members across all SNPs

Non-informative SNPs are handled as follows. In every case, a child whose genotype is missing or inconsistent with its parents' genotypes is left unphased.

- Both parents homozygous: each parent passes on its only allele.

- Both parents heterozygous: homozygous children carry their only allele. Each parent is phased from which of its haplotypes the homozygous children inherited at the nearest phased SNPs on both sides, and left unknown if that evidence is missing or conflicting. A heterozygous child is phased from the haplotypes it inherited from the phased parents.
- One parent heterozygous, one homozygous (informative SNPs left out because of a missing or inconsistent genotype): each child's alleles follow from the genotypes. The heterozygous parent is phased from which of its haplotypes the children inherited at the nearest phased SNPs on both sides, and left unknown if that evidence is missing or conflicting.

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.

Value

A VCF object with one row per SNP and one column per family member, for all phased families. The `GT` genotype field holds phased genotypes (e.g. `"0|1"`), coded against `REF` (PLINK allele 2, BIM column 6) and `ALT` (PLINK allele 1, BIM column 5), the convention of plink2 and of PLINK 1.9 with `--keep-allele-order`. A BIM file made by PLINK 1.9 without that flag has the minor allele first, so `REF` is then the major allele, not necessarily the reference allele; `"."` marks an allele that could not be phased. For children the first allele is paternal and the second maternal; for parents the alleles are the two phased haplotypes. `colData` holds `familyId`, `fatherId`, `motherId`, `sex` and `affected` from the FAM file. Families without phase information are omitted with a message. Chromosome names use the UCSC style (e.g. `chr4`). Use `VariantAnnotation::writeVcf()` to write the result to a file.

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
)

phased

# Phased genotypes for the members of family F6
f6 <- colData(phased)$familyId == "F6"
head(geno(phased)$GT[, f6])

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

phasedSubset

# Write to a VCF file
vcfFile <- tempfile(fileext = ".vcf")
writeVcf(phasedSubset, vcfFile)
```

Index

crossoverData, 2

geneticMaps-data, 4
GRanges, *2*, *6*, *9*, *11*
GRangesList, *5*, *6*, *9*

hzRun, 4

simCEU-data, 7

VCF, *12*

xoDetect, 8
xoPhase, 10