Package: RZooRoH (via r-universe)

November 1, 2026

Type Package

Title Partitioning of Individual Autozygosity into Multiple Homozygous-by-Descent Classes

Version 0.4.2

Maintainer Tom Druet <tom.druet@uliege.be>

Description Functions to identify Homozygous-by-Descent (HBD) segments associated with runs of homozygosity (ROH) and to estimate individual autozygosity (or inbreeding coefficient). HBD segments and autozygosity are assigned to multiple HBD classes with a model-based approach relying on a mixture of exponential distributions. The rate of the exponential distribution is distinct for each HBD class and defines the expected length of the HBD segments. This rate is called the ``rate of coancestry change". The HBD classes are therefore related to the age of the segments (longer segments and smaller rates for recent autozygosity / recent common ancestor). The functions allow to estimate the parameters of the model (rates of the exponential distributions called also rates of coancestry change; mixing proportions related to the inbreeding rate per layer or generation), to estimate global and local autozygosity probabilities and to identify HBD segments with the Viterbi decoding. Functions also allow to compute identity-by-descent (IBD) between pairs of haplotypes, to estimate kinship between pairs of individuals and to predict inbreeding in the future progeny of a genotyped couple. The current model is fully described in Druet and Gautier (2022) <doi:10.1016/j.tpb.2022.03.001>. The model and its properties were originally presented in Druet and Gautier (2017) <doi:10.1111/mec.14324>. Extension to IBD and kinship analyses is described in Forneris et al. (2025) <doi:10.1111/1755-0998.14068>.

Depends R (>= 3.5.0), methods

License GPL-3

Encoding UTF-8

LazyData true

Imports foreach, doParallel, parallel, data.table, RColorBrewer, iterators

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation yes

Author Tom Druet [aut, cre], Naveen Kumar Kadri [aut], Natalia Forneris [ctb], Pierre Faux [ctb], Amandine Bertrand [ctb], Mathieu Gautier [aut]

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

Date/Publication 2026-10-02 18:32:24 UTC

RemoteUrl https://github.com/cran/RZooRoH

RemoteRef HEAD

RemoteSha 8e5503dcbffa8b4e70e51f280ba654c08e9042fc

Contents

BBB_NMP_ad_subset . . . 3
BBB_NMP_GP_subset . . . 4
BBB_NMP_pl_subset . . . 4
BBB_PE_gt_subset . . . 5
BBB_samples . . . 6
ComputeFseg . . . 6
ComputeFsegAllT . . . 7
cumhbd . . . 8
cumkin . . . 8
genoex . . . 9
genosim . . . 9
hbd1 . . . 10
hbd2 . . . 11
kintaf_mix4l . . . 11
map1 . . . 12
merge_kinres . . . 12
merge_zres . . . 13
predhbd . . . 13
probhbd . . . 14
rara_mix10l . . . 15
realized . . . 16
rohbd . . . 16
soay_mix10l . . . 18
typs . . . 18
typsfrq . . . 19
update_kinres . . . 19

update_zres 20
wilt_mix10l 20
zoodata 21
zookin 25
zoomodel 29
zooplot_hbdseg 32
zooplot_individuals 33
zooplot_partitioning 34
zooplot_prophbd 36
zoorun 37

Index **43**

`BBB_NMP_ad_subset` *Example for "ad" format specification*

Description

A dataset containing real allele depth information for 1,000 SNPs. The data is available for ten individuals and the first 1,000 SNPs on chromosome 1. The data corresponds to a low-fold sequencing experiment. There are two columns per individuals (read counts for allele 1 and for allele 2).

Usage

```
BBB_NMP_ad_subset
```

Format

A data frame with 1,000 rows and 25 variables:

chr The chromosome number

marker_name The marker id

pos The position of the marker

allele1 The name of the first marker allele

allele2 The name of the second marker allele

id1_ad1 The read count for the first individual at the first markrer

id1_ad2 The read count for the first individual at the second markrer

id2_ad1 The read count for the second individual at the first markrer

id2_ad2 The read count for the second individual at the second markrer

id3_ad1, id3_ad2, id4_ad1, id4_ad2, id5_ad1, id5_ad2, id6_ad1, id6_ad2, id7_ad1, id7_ad2, id8_ad1, id8_ad2, id9_ad
The read counts for the other individuals

BBB_NMP_GP_subset *Example for "gp" format specification*

Description

A dataset containing real genotype probabilities for 1,000 SNPs. The data is available for ten individuals and 1,000 SNPs on chromosome 10. The data corresponds to a low-fold sequencing experiment. There are three columns per individuals (for genotypes 00, 01 and 11).

Usage

```
BBB_NMP_GP_subset
```

Format

A data frame with 1,000 rows and 35 variables:

chr The chromosome number

marker_name The marker id

pos The position of the marker

allele1 The name of the first marker allele

allele2 The name of the second marker allele

id1_gp1 The AA genotype probability for the first individual

id1_gp2 The AB genotype probability for the first individual

id1_gp3 The BB genotype probability for the first individual

id2_gp1 The AA genotype probability for the second individual

id2_gp2 The AB genotype probability for the second individual

id2_gp3 The BB genotype probability for the second individual

id3_gp1, id3_gp2, id3_gp3, id4_gp1, id4_gp2, id4_gp3, id5_gp1, id5_gp2, id5_gp3, id6_gp1, id6_gp2, id6_gp3, id7_gp
The genotype probabilities for the other individuals

BBB_NMP_pl_subset *Example for "pl" format specification*

Description

A dataset containing real genotype likelihoods in phred scores for 1,000 SNPs. The data is available for ten individuals and the first 1,000 SNPs on chromosome 1. The data corresponds to a low-fold sequencing experiment. There are three columns per individuals (for genotypes 00, 01 and 11).

Usage

```
BBB_NMP_pl_subset
```

Format

A data frame with 1,000 rows and 35 variables:

chr The chromosome number

marker_name The marker id

pos The position of the marker

allele1 The name of the first marker allele

allele2 The name of the second marker allele

id1_pl1 The AA phred likelihood for the first individual

id1_pl2 The AB phred likelihood for the first individual

id1_pl3 The BB phred likelihood for the first individual

id2_pl1 The AA phred likelihood for the second individual

id2_pl2 The AB phred likelihood for the second individual

id2_pl3 The BB phred likelihood for the second individual

id3_pl1, id3_pl2, id3_pl3, id4_pl1, id4_pl2, id4_pl3, id5_pl1, id5_pl2, id5_pl3, id6_pl1, id6_pl2, id6_pl3, id7_pl1, id7_
The phred likelihoods for the other individuals

BBB_PE_gt_subset *Example for "gt" format specification*

Description

A dataset containing real genotypes for 1,000 SNPs. The data is available for ten individuals and the first 1,000 SNPs on chromosome 1. The data corresponds to a WGS experiment. There is one columns per individual.

Usage

```
BBB_PE_gt_subset
```

Format

A data frame with 1,000 rows and 14 variables:

chr The chromosome number

pos The position of the marker

allele1 The name of the first marker allele

allele2 The name of the second marker allele

id1 The genotypes for the first individuals (id1)

id2 The genotypes for the second individuals (id2)

id3, id4, id5, id6, id7, id8, id9, id10 The genotypes of the remaining individuals

`BBB_samples` *A file with names or IDs for ten samples.*

Description

The names (or IDs) are provided for ten samples.

Usage

```
BBB_samples
```

Format

A data frame with 10 rows and one variable.

IDs The ids for ten individuals

`ComputeFseg` *Computes inbreeding coefficients based on HBD segments identified with the Viterbi algorithm (similar to a F_ROH estimator), restricted to HBD classes with a rate lower or equal to a chosen threshold T. Returns, per individual, the number of qualifying segments and the corresponding inbreeding coefficient FSEG.*

Description

Computes inbreeding coefficients based on HBD segments identified with the Viterbi algorithm (similar to a F_ROH estimator), restricted to HBD classes with a rate lower or equal to a chosen threshold T. Returns, per individual, the number of qualifying segments and the corresponding inbreeding coefficient FSEG.

Usage

```
ComputeFseg(zoores, zoodata, T = NULL, use_names = FALSE)
```

Arguments

`zoores`	a zoores object (output of zoorun)
`zoodata`	a zooin object (output of zoodata), it must be the zooin object associated to the zoores object
`T`	is the maximum HBD class rate to include (classes with zoores@krates <= T are summed)
`use_names`	if TRUE, replaces numeric identifiers with the individual names (zoodata@sample_ids). In that case, zoodata@sample_ids must exist. Set to FALSE (default) for an IBD analysis (in that case ids do not correspond to sample_ids but to positions in the list of pairs).

Value

a data.frame with one row per individual, the number of HBD segments (NSEG) and the inbreeding coefficient (FSEG) computed over all classes with rate <= T.

`ComputeFsegAllT` *Computes inbreeding coefficients based on HBD segments identified with the Viterbi algorithm (similar to a F_ROH estimator). Returns cumulative inbreeding coefficients estimated using different thresholds corresponding to the rates of the HBD classes.*

Description

Computes inbreeding coefficients based on HBD segments identified with the Viterbi algorithm (similar to a F_ROH estimator). Returns cumulative inbreeding coefficients estimated using different thresholds corresponding to the rates of the HBD classes.

Usage

```
ComputeFsegAllT(zoores, zoodata, use_names = FALSE)
```

Arguments

`zoores`	a zoores object (output of zoorun)
`zoodata`	a zooin object (output of zoodata), it must be the zooin object associated to the zoores object
`use_names`	if TRUE, replaces numeric identifiers with the individual names (zoodata@sample_ids). In that case, zoodata@sample_ids must exit. Set to FALSE (default) for an IBD analysis (in that case ids do not correspond to sample_ids but to positions in the list of pairs).

Value

a data.frame with one row per individual and one column per HBD class (cumulative), column names being prefixed "F_class". The values are the inbreeding coefficient estimated using HBD segments associated with HBD classes with a rate lower or equal to the threshold.

cumhbd *Computes the realized inbreeding coefficient*

Description

Computes the realized inbreeding coefficient with respect to a base population by summing the autozygosity from all HBD class with a rate lower or equal to the threshold T. This amounts to set the base population approximately 0.5*T generations ago. HBD classes with a higher rate are then no longer considered as autozygous.

Usage

```
cumhbd(zres, T = NULL)
```

Arguments

zres	The name of the zres object created by the zoorun function.
T	The value chosen to define the base population. When T is not provided, all HBD classes are considered to estimate the inbreeding coefficient.

Value

An array with the compute inbreeding coefficients for all the individuals in the analysis.

cumkin *Computes the realized kinship*

Description

Computes the realized kinship with respect to a base population by summing the relatedness from all classes with a rate lower or equal to the threshold T. This amounts to set the base population approximately 0.5*T generations ago. Classes with a higher rate are then no longer considered as IBD.

Usage

```
cumkin(kres, T = NULL)
```

Arguments

kres	The name of the kres object created by the zoorun function.
T	The value chosen to define the base population. When T is not provided, all classes are considered to estimate the kinship.

Value

An array with the computed kinship for all the pairs of individuals in the analysis.

genoex *Subset of a dataset with genotypes for 20 sheeps*

Description

A dataset containing real genotypes for 20 individuals. Genotypes are available for 14,831 SNPs from the first three chromosomes were selected. The twenty last columns correspond to the genotypes. Missing genotypes are set to 9.

Usage

```
genoex
```

Format

A data frame with 14,831 rows and 25 variables:

chr The chromosome number

marker_name The name of the marker

pos The position of the marker

allele1 The name of the first marker allele

allele2 The name of the second marker allele

id1 The genotype for the first individual

id2 The genotype for the second individual

id3, id4, id5, id6, id7, id8, id9, id10, id11, id12, id13, id14, id15, id16, id17, id18, id19, id20 The genotypes of the remaining individuals

genosim *Example from a small simulated data set*

Description

A dataset containing simulated genotypes for 20 individuals. Genotypes are available for 10,000 SNPs on 10 chromosomes. The ten last columns correspond to the genotypes.

Usage

```
genosim
```

Format

A data frame with 10,000 rows and 24 variables:

chr The chromosome number

pos The position of the marker

allele1 The name of the first marker allele

allele2 The name of the second marker allele

id1 The genotype for the first individual

id2 The genotype for the second individual

id3, id4, id5, id6, id7, id8, id9, id10, id11, id12, id13, id14, id15, id16, id17, id18, id19, id20 The genotypes of the remaining individuals

---

hbd1 *The result of local IBD probabilities for a pair of individuals among the 18 cows from the Amsterdam Island.*

Description

The results were obtained by running the a model with 4 layers and the predhbd function for all HBD classes. Results were extracted for chromosome 1.

Usage

```
hbd1
```

Format

the result is an array.

Details

Results were extracted for chromosome 1 and the map is also provided (map1).

hbd2 — *The result of local IBD probabilities for a pair of individuals among the 18 cows from the Amsterdam Island (recent only).*

Description

The results were obtained by running the a model with 4 layers and the predhbd function for HBD classes with a rate < 10. Results were extracted for chromosome 1.

Usage

```
hbd2
```

Format

the result is an array.

kintaf_mix4l — *The result of a kinship analysis on 18 cattle from the Amsterdam Island.*

Description

The results were obtained by running the a model with 4 layers and the kinship option among all pairs of individuals (genotyped at 23679 SNPs after filtering).

Usage

```
kintaf_mix4l
```

Format

the results are a zres object.

map1 — *The map for local IBD probabilities for a pair of individuals among the 18 cows from the Amsterdam Island (recent only).*

Description

The map corresponds to the results obtained by running the a model with 4 layers and the predhbd function (see hbd1 and hbd2). Results were extracted for chromosome 1.

Usage

```
map1
```

Format

the result is an array.

merge_kinres — *Merge several kinres objects generated by zookin*

Description

The function is used for example when the kinship analysis has been split in several tasks. All the kinres objects must come from the same original zoodata and from zookin() runs using the same model (zookin is applied to the same zoodata with the same model - as for merge_zres). In addition, the data should not overlap (the same pair of individuals should not be present in multiple kinres objects).

Usage

```
merge_kinres(list_kinres)
```

Arguments

list_kinres	A list with the name of the kinres objects to be merged.

Value

a single kinres object containing the results from the merged kinres objects. Note that the IBD segments are not sorted by ID number.

merge_zres *Merge several zres objects generated by zoorun*

Description

The function is used for example when the analysis has been split in several tasks. All the zres must come from the same original zoodata (zoorun is applied to the same zoodata with the same model). In addition, the data should not overlap (the same individual, or the same pair in an IBD analysis, should not be present in multiple zres).

Usage

```
merge_zres(list_zres)
```

Arguments

list_zres	A list with the name of the zres objects to be merged.

Details

The function works both for a standard (HBD) analysis and for an IBD analysis (zoorun run with ibd = TRUE), and detects automatically which one it is from the format of @sampleids. For a standard analysis, @ids are kept unchanged (they are absolute, stable positions in the original zoodata file) and only sorted, exactly as before. For an IBD analysis, @ids are only local to each zoorun() call and are therefore not comparable across the different zres objects being merged: the merged pairs are re-sorted numerically on (id1, hap1, id2, hap2) - or alphabetically if identifiers are not numeric - and @ids are reassigned as new, globally consistent numbers.

Value

a single zres object containing the results from the merged zres objects. Note that the HBD/IBD segments are not sorted by ID number.

predhbd *Extracts the IBD probabilities from the kres object*

Description

Extracts the locus specific IBD probabilities for a pair of individuals. This is the probability that one pair of chromosomes (one sampled in each individual) are IBD at that position. A specific chromosomal region can be specified. A threshold T can be used to determine which classes are used in the computation of the IBD probability. This function requires that the option "localhbd" was set to TRUE when creating the kres object.

Usage

```
predhbd(
  kres,
  zooin,
  num,
  chrom = NULL,
  startPos = NULL,
  endPos = NULL,
  T = FALSE
)
```

Arguments

kres	The name of the kres object created by the zookin function.
zooin	The name of the zdata object created by the zoodata function. See "zoodata" for more details.
num	The number of the pair of individuals to extract (correspond to the position in the list of pairs).
chrom	the number of the chromosome where we are looking for IBD segments. This chromosome number refers to the position of the chromosome in the list of all chromsomes present in the input genotype data.
startPos	The starting position (on the chromosome) of the interval from which we will extract IBD segments (1 by default).
endPos	The ending position (on the chromosome) of the interval from which we will extract IBD segments (last position by default).
T	The value chosen to define the base population (to determine which classes are used in estimated of IBD probability, which classes are considered autozygous). When T is not provided, all IBD classes are considered to estimate the local IBD probability.

Value

The function returns a vector of IBD probabilities averaged over the four combination of pairs of parental haplotypes for the specified pair of individuals and chromosomal region. The IBD probabilities are computed as the sum of the probabilities for each class with a rate smaller or equal than the threshold (the sum from all the IBD classes when T is not specified).

---

probhbd *Extracts the HBD probabilities from the zres object*

Description

Extracts the locus specific HBD probabilities for an individual. A specific chromosomal region can be specified. A threshold T can be used to determine which HBD classes are used in the computation of the HBD probability. This function requires that the option "localhbd" was set to TRUE when creating the zres object.

Usage

```
probhbd(
  zres,
  zooin,
  id,
  chrom = NULL,
  startPos = NULL,
  endPos = NULL,
  T = FALSE
)
```

Arguments

zres	The name of the zres object created by the zoorun function.
zooin	The name of the zdata object created by the zoodata function. See "zoodata" for more details.
id	The number of the individual to extract.
chrom	the number of the chromosome where we are looking for HBD segments. This chromosome number refers to the position of the chromosome in the list of all chromsomes present in the input genotype data.
startPos	The starting position (on the chromosome) of the interval from which we will extract HBD segments (1 by default).
endPos	The ending position (on the chromosome) of the interval from which we will extract HBD segments (last position by default).
T	The value chosen to define the base population (to determine which classes are used in estimated of HBD probability, which classes are considered autozygous). When T is not provided, all HBD classes are considered to estimate the local HBD probability.

Value

The function returns a vector of HBD probabilities for the specified individual and chromosomal region. The HBD probabilities are computed as the sum of the probabilities for each HBD class with a rate smaller or equal than the threshold (the sum from all the HBD classes when T is not specified).

---

rara_mix10l *The result of an analysis on 22 sheeps from Rasa Aragonesa population.*

Description

The results were obtained by running the default model (10 layers with pre-defined rates) on 22 individuals genotyped at 37465 SNPs.

Usage

```
rara_mix10l
```

Format

the results are a zres object.

`realized` *Extracts the realized autozygosity from the zres object*

Description

Extracts the realized autozygosity from the zres object. Extraction is performed for the indicated classes (all by default) and names are added to the columns. The function must be used with more than one individual is the zres object.

Usage

```
realized(zres, classNum = NULL)
```

Arguments

`zres`	The name of the zres object created by the zoorun function.
`classNum`	An array with the number of the classes to extract. All classes are extracted by default.

Value

The function returns a data frame with one row per individual and one column per extracted classes. In addition, it gives names to the columns. For a pre-defined model, the names of HBD classes are "R_X" where X is the rate of the corresponding class. For a model with rate estimation, the names of the HBD classes are "HBDclassX" where X is the number of the HBD class. For non-HBD classes, we use "NonHBD".

`rohbd` *Extracts the HBD segments from the zres object*

Description

Extracts the HBD segments (or RoH) from the zres object. Extraction is performed for the indicated individuals and the selected region (all by default).

Usage

```
rohbd(
  zres,
  ids = NULL,
  chrom = NULL,
  startPos = NULL,
  endPos = NULL,
  inside = TRUE
)
```

Arguments

zres	The name of the zres object created by the zoorun function.
ids	An array with the ids of the individuals to extract. All individuals are extracted by default.
chrom	the number of the chromosome where we are looking for HBD segments. This chromosome number refers to the position of the chromosome in the list of all chromsomes present in the input genotype data.
startPos	The starting position (on the chromosome) of the interval from which we will extract HBD segments (1 by default).
endPos	The ending position (on the chromosome) of the interval from which we will extract HBD segments (last position by default).
inside	A logical indicating whether we extract only segment within the interval (TRUE) or overlapping with the interval (FALSE). By 'within the interval', we mean that both starting and end position of the HBD segment should be in the interval. By 'overlapping', we mean that at least one part of the HBD segment should be located in the interval.

Value

The function returns a data frame with the HBD segments fitting the filtering rules (id and position). The data frame has one line per identified HBD segment and nine columns: id is the number of the individual in which the HBD segments is located, chrom is the chromosome of the HBD segments, start_snp is the number of the SNP at which the HBD segment starts (the SNP number within the chromosome), start_end is the number of the SNP at which the HBD segment ends (the SNP number within the chromosome), start_pos is the position at which the HBD segment starts (within the chromosome), start_end is the position at which the HBD segment ends (within the chromosome), number_snp is the number of consecutive SNPs in the HBD segment, length is the length of the HBD segment (for instance in bp or in cM/1000000) and HBDclass is the HBD class associated with the HBD segment.

soay_mix10l — *The result of an analysis on 110 sheeps from the Soay population.*

Description

The results were obtained by running the default model (10 layers with pre-defined rates) on 110 individuals genotyped at 37465 SNPs.

Usage

```
soay_mix10l
```

Format

the results are a zres object.

typs — *Subset of a dataset with genotypes for 6 individuals from a cattle population.*

Description

A dataset containing real genotypes for 6 individuals. Genotypes are available for a low density array with 6370 SNPs on 29 autosomes. The six last columns correspond to the genotypes. Missing genotypes are set to 9.

Usage

```
typs
```

Format

A data frame with 6370 rows and 10 variables:

chr The chromosome number

pos The position of the marker

allele1 The name of the first marker allele

allele2 The name of the second marker allele

id1 The genotypes for the first individuals (id1)

id2 The genotypes for the second individuals (id2)

id3 The genotypes for the third individuals (id3)

id4 The genotypes for the fourth individuals (id4)

id5 The genotypes for the fifth individuals (id5)

id6 The genotypes for the sixth individuals (id6)

typsfrq *A file with marker allele frequencies for the cattle population.*

Description

The allele frequencies of the first allele were computed in a larger sample.

Usage

```
typsfrq
```

Format

A data frame with 6370 rows and one variable.

frq The allele frequencies estimated for the first allele

update_kinres *Update one main kinres object with new results*

Description

The function is used for example when the main kinship analysis failed for one pair of individuals. The analysis is repeated for that pair with other settings, and the function inserts the new results in the main kinres object. The pair(s) in kinres2 must already be present in kinres1, and both objects must come from the same zoodata and the same model (as for update_zres). To avoid generating too large files, the updated kinres object can take the same name as kinres1, doing kinres1 <- update_kinres(kinres1,kinres2).

Usage

```
update_kinres(kinres1, kinres2)
```

Arguments

kinres1	The main kinres object that will be modified.
kinres2	The new kinres object, with the new results that will be inserted in the main kinres object.

Value

an updated kinres object containing the results from kinres1 updated by those from kinres2.

update_zres *Update one main zres object with new results*

Description

The function is used for example when the main analysis failed for one individual (or one pair, in an IBD analysis). The analysis is repeated with other settings and the function inserts the new results in the main zres object. However the model must still be the same! The individual(s)/pair(s) in zres2 must already be present in zres1. To avoid generating too large files, the updated zres object can take the same name as zres1, doing zres1 <- update_zres(zres1,zres2).

Usage

```
update_zres(zres1, zres2)
```

Arguments

zres1	The main zres object that will be modified.
zres2	The new zres object, with the new results that will be inserted in the main zres object.

Details

As for merge_zres, the function detects automatically whether this is a standard (HBD) or an IBD analysis from the format of @sampleids, and matches accordingly (on @ids for a standard analysis, on @sampleids for an IBD analysis, since @ids are then only local to each zoorun() call). In both cases, @ids in zres1 are left unchanged.

Value

an updated zres object containing the results from zres1 updated by those from zres2.

wilt_mix10l *The result of an analysis on 23 sheeps from Wiltshire population.*

Description

The results were obtained by running the default model (10 layers with pre-defined rates) on 23 individuals genotyped at 37465 SNPs.

Usage

```
wilt_mix10l
```

Format

the results are a zres object.

zoodata *Read the genotype data file*

Description

Read a data file and convert it to the RZooRoH format in a 'zooin' object required for further analysis.

Usage

```
zoodata(
  genofile,
  min_maf = 0,
  zformat = "gt",
  chrcol = 1,
  poscol = 0,
  supcol = 0,
  haploid = FALSE,
  allelefreq = NULL,
  freqem = TRUE,
  samplefile = NA,
  map_unit = "bp",
  RecRate = 1
)
```

Arguments

genofile	The name of the input data file. Note that the model is designed for autosomes. Other chromosomes and additional filtering (e.g. call rate, missing, HWE, etc.) should be performed prior to run RZooRoH with tools such as PLINK or bcftools for instance. The model works on an ordered map and ignores SNPs with a null position.
min_maf	The minimum allele frequency to keep variants in the analysis (optional / set to 0.00 by default to keep all markers). Values such as 0.01 allows to exclude monomorphic markers that are not informative and to reduce the size of the data and computational costs. There is no marker exclusion on call rate. However, we expect that data filtering is done prior to RZooRoH with tools such as PLINK or vcftools.
zformat	The code corresponding to the format of the data file ("gt" for genotypes, "gp" for genotype probabilities, "gl" for genotype likelihoods in Phred scores, "ad" for allelic depths). For all these formats, markers are ordered per rows and individuals per columns. Variants should be ordered by chromosome and position. By default, the format is based on the Oxford/GEN format, and the first five columns are chromosome identification (e.g, "1", "chr1"), the name of the marker, the position of the marker in base pairs or better in cM multiplied when genetic distances are known (see the map_unit and RecRate options for more

details), the first marker allele and the second marker allele. Information per individual varies according to the format. With the "gt" format we have one column per individual with 0, 1 and 2 indicating the number of copies of the first allele (and 9 for missing). With the "gp" format we have three column per individual with the probabilities of genotype 11 (homozygous for the first allele), genotype 12 and genotype 22 (this corresponds to the oxford GEN format). Similarly, with the "gl" format, we have three column per individual with the likelihoods for genotypes 11, 12 and 22 in Phred scores. Finally, with the "ad" format, we expect two columns per individual: the number of reads for allele 1 and the number of reads for allele 2. For these three last formats, missing values must be indicated by setting all elements to 0. If one of the columns is non-null for one individual, the genotype will be considered non-missing. Note that the marker alleles specified in columns 4 and 5 are not used.

For applications related to identity-by-descent (IBD) estimation, the ZooRoH model is applied on two phased chromosomes or two haploid chromosomes. Two additional formats are therefore available to provide haplotypes (or haploid data). The first is "vcf" which refers to a phased VCF (for example, the output from Beagle5). In that case, the VCF must contain only the phased haplotype information (e.g 0|1, 0|0, 1|1, etc). The phased VCF can only be used with diploid individuals (after phasing). By default, we assume that the first and second column indicate chromosome and SNP position and that the individual haplotypes information starts in column 10. The second format is "haps" that is similar to the "GEN" format with five columns (chromosome identification (e.g, "1", "chr1"), the name of the marker, the position of the marker in base pairs or better in cM multiplied by 1,000,000 when genetic distances are known, the first marker allele and the second marker allele) followed by the haplotypes (two columns for diploid individuals and one in case of haploid chromosome). The alleles are coded as 0 and 1. The "haps" format is the only format that can be used with haploid data.

It is possible to convert a PLINK PED/BED file or a VCF file to the RZooRoH genotype format. This can be done using tools such as PLINK or bcftools. Both of these tools can for example be used to convert to GEN format. With PLINK2, it is possible to convert the file to a dosage file (0/1/2), which is similar to the 'GT' format, but with six columns of metadata before the genotypes. Please note that the GEN format has recently been modified slightly, with some versions having 5 or 6 additional columns (with or without chromosome information), and the conversion options have changed with VCFtools.

For example, conversion to GT format is performed with PLINK2 using the command plink2 –bfile myinput –export Av –out myoutput. Then, the header line should be deleted. Ideally, NA genotypes should be replaced with 9, which can be achieved using a bash command:

sed -e 's/NA/9/g' myoutput > newfile

The 'newfile' can then be read with the zoodata function, but the poscol and supcol parameters (see below) must be adjusted to poscol=4 and supcol=6.

We also provide examples using previous versions of PLINK (version 1.9) and VCFtools (conversions are also possible with the new versions). As shown above, RZooRoH is flexible if there are additional columns, as users can adjust the reading using options such as 'chrcol', 'poscol' and 'supcol' (see below).

For ped files, PLINK (v1.9) can recode them to oxford gen format with plink –file myinput –recode oxford –autosome –out myoutput. The autosome option keeps only #' SNPs on autosomes as required by RZooRoH.

For vcf files, bcftools can be used to recode a vcf to the oxford gen format with the convert option: bcftools convert -t ^chrX,chrY,chrM -g outfile –chrom –tag GT myfile.vcf. The –chrom option is important to obtain chromosome number in the first column. The tag option allows to select which field from the vcf file (GT, PL, GL or GP) is used to generate the genotype probabilities exported in the oxford gen format. The -t option allows to exclude chromosomes (this is an example and chromosome names must be adapted if necessary). The needed output data is then outfile.gen.

If some genotype probabilities are missing, with a value of "-nan", you must replace them with "0" (triple 0 is considered as missing). This can be done with this command:

sed -e 's/-nan/0/g' file.gen > newfile.gen

It is important to properly encode missing records, as some missing symbols are not recognised and will generate errors. Note also that some software recodes missing values as 0.333, 0.333 and 0.333. This was not previously considered as missing data. This has no effect on emission probabilities, but slightly impacts the estimation of allele frequencies when the freqem option is FALSE. If the three genotype probabilities are now above 0.33, the genotype is considered missing. Ideally, missing values should be set to 0, 0, 0 (for the 'gp' format).

As for genotype formats, tools such as bcftools can be used to format files in the 'haps' or phased VCF format.

`chrcol` An optional argument that indicates the column number where the chromosome information is indicated (first column by default for all formats).

`poscol` An optional argument that indicates the column number where the marker position is indicated (third column by default for all formats except phased vcf where it is the second column). See also

`supcol` An optional argument that indicates the number of additional columns before the individuals genotypes or haplotypes (five columns are expected by default as described in the zformat argument description, except for phased VCF where this value is set to 9). Note that the function requires at least two information columns: the chromosome number and the marker position.

`haploid` An optional argument that indicates whether we work on an haploid organism or chromosome (default = FALSE). This is only compatible with the 'IBD' option from the zoorun function (we don't estimate HBD in haploid data!). In the case you use haploid data, the only possible format is "haps".

`allelefreq` A vector with allele frequencies for the first marker allele (optional). By default, the allele frequencies are estimated from the data. The option allows to skip this computation or to provide external allele frequencies estimated by another method or on another data set.

`freqem` A logical indicating whether allele frequencies should be estimated with an EM algorithm (TRUE by default). Simpler approaches are used when set to FALSE. The approach is ignored with the GT, VCF and HAPS formats. For high confidence genotypes (e.g., genotyping arrays, high-coverage sequencing data), it is

not necessary to use this EM approach as genotypes or haplotypes are known. The approach is more relevant with low-fold sequencing for example, and more so with the PL or GP format (the approximation with the AD format being closer to the EM).

`samplefile` A file with names of the samples (optional). It must match with the number of genotyped individuals. If none is provided, the position in the genofile is used as ID.

`map_unit` Can be "bp" or "cM". By default the variant positions are assumed to be the physical positions in base pairs (bp). If the user knows the map in centimorgans (cM), they can provide the positions in cM and set the map_unit to "cM". Note that the maximum resolution is 1e-6 cM: map positions will be rounded to this resolution, and two markers separated by less than 1e-6 cM may therefore end up at the same rounded position, in which case no recombination will be possible between them. If the positions are provided in bp but the cM-to-Mb ratio is not equal to 1, the user can specify the correct ratio using the 'RecRate' option (default: 1 cM/Mb).

`RecRate` This gives the number of centimorgans (cM) per megabase (Mb). By default this value is set to 1 cM per Mb. Therefore, the rates of the exponential distributions correspond to rates of coacenstry changes per Morgan, and the expected length of HBD segments is equal to 100 cM / R. This value will be used to convert the physical position to cM. Note that the maximum resolution in cM is 1e-6 cM: map positions will be rounded to this resolution, so two markers separated by less than 1e-6 cM after conversion (which becomes more likely as the cM-to-Mb ratio decreases below 1) may end up at the same rounded position, in which case no recombination will be possible between them. This option is not used when the positions are provided directly in cM (when "map_unit" is set to "cM").

Value

The function return a zooin object called containing the following elements: zooin@genos a matrix with the genotypes, genotype probabilities or haplotypes, zooin@bp an array with marker positions, zooin@chrbound a matrix with the first and last marker number for each chromosome, zooin@nind the number of individuals, zooin@nsnps the number of markers conserved after filtering for minor allele frequency, zooin@freqs an array with the marker allele frequencies, zooin@nchr the number of chromosomes, zooin@zformat the format of the data ("gt","gp","gl","ad","vcf","haps") and zooin@sample_ids (the names of the samples).

Examples

```
# Get the name and location of example files

myfile1 <- system.file("exdata","genoex.txt",package="RZooRoH")
myfile2 <- system.file("exdata","genosim.txt",package="RZooRoH")

# Load your data with default format into a zooin object named "data1":

data1 <- zoodata(myfile1)

# Load the first data file with default format and filtering out markers with MAF < 0.02
```

```
# into a zooin object called "data1frq002":

data1frq002 <- zoodata(myfile1, min_maf = 0.02)

# Load the first data file with default format, with external allele frequencies
# (here a random set we create) and filtering out markers with MAF < 0.01:

myrandomfreq <- runif(14831)
data1c <- zoodata(myfile1, allelefreq = myrandomfreq, min_maf = 0.01)

# Load the second data file and indicate your own format (chromosome number in column 1,
# map position in column 2, 4 columns before genotypes) and filtering out markers with
# MAF < 0.01. The created zooin object is called "Sim5":

Sim5 <- zoodata(myfile2, chrcol = 1, poscol =2, supcol = 4, min_maf = 0.01)
```

`zookin` *Use the ZooRoH model to estimate kinship between pairs of individuals*

Description

Apply the defined model to estimate kinship for pairs of individuals by running the ZooRoH model to estimate IBD probabilities between the four possible pairs of chromosome of the two individuals (one chromosome from each individual). As for zoorun this includes parameter estimation, computation of realized IBD, IBD probabilities, and identification of IBD segments. Options are similar to the zoorun function.

Usage

```
zookin(
  zoomodel,
  zooin,
  parameters = TRUE,
  fb = TRUE,
  vit = TRUE,
  localhbd = FALSE,
  nT = 1,
  optim_methods = "L-BFGS-B",
  maxiter = 1000,
  minmix = 1,
  maxr = 1e+08,
  kinpairs = NULL,
  RecTable = FALSE,
  additional_zmix = NULL,
  trim_ad = FALSE,
  hemiprob = 0
)
```

Arguments

zoomodel	A valid zmodel object as defined by the zoomodel function. The model indicates whether rates of exponential distributions are estimated or predefined, the number of classes, the starting values for mixing coefficients and rates, the error probabilities. See "zoomodel" for more details. zookin can not be run with a KL model because four different rate parameters would be estimated for each pair of haplotypes, making interpretation difficult.
zooin	A valid zdata object as obtained by the zoodata function. See "zoodata" for more details.
parameters	Specifies whether the parameters are estimated by optimization with the L-BFGS-B method (default) from the optim function (optional argument - true by default). If the user doesn't want to estimate the parameters he must set parameters=FALSE. In that case, the forward-backaward and Viterbi algorithms are run with the provided parameters.
fb	A logical indicating whether the forward-backward algorithm is run (optional argument - true by default). The Forward-Backward algorithm estimates the local probabilities to belong to each IBD or non-IBD class. By default, the function returns only the IBD probabilities for each class, averaged genome-wide, and corresponding to the realized IBD levels associated with each class. To obtain IBD probabilities at every marker position, the option localhbd must be set to true (this generates larger outputs).
vit	A logical indicating whether the Viterbi algorithm is run (optional argument - false by default). The Viterbi algorithm performs the decoding (determining the underlying class at every marker position). Whereas the Forward-Backward algorithms provide IBD probabilities (and how confident a region can be declared IBD), the Viterbi algorithm assigns every marker position to one of the defined classes (IBD or non-IBD). When informativity is high (many SNPs per IBD segments), results from the Forward-Backward and the Viterbi algorithm are very similar. The Viterbi algorithm is best suited to identify IBD segments. To estimate realized kinship and determine IBD status of a position, we recommend to use the Forward-Backward algorithm that better reflects uncertainty.
localhbd	A logical indicating whether the IBD probabilities for each individual at each marker are returned when using the Forward-Backward algorithm (fb option). This is an optional argument that is false by default.
nT	Indicates the number of parallel processes (workers) used when running RZooRoH in parallel (optional argument; one process by default, i.e. no parallelization). Parallelization relies on process forking and is limited to a single compute node: when submitting jobs on a cluster, make sure to request nT CPU cores on the same node (e.g. via –cpus-per-task on SLURM), not distributed across multiple nodes.
optim_methods	Indicates which method the optim R function will use to estimate the parameters of the model ("L-BFGS-B" by default). The possible methods are "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent". Type "? optim" to have more information. In our experience, the "L-BFGS-B" method works well but the method achieving the best likelihood is variable (according to the

data sets, the model, the priors, the constraints). The same goes for the efficiency (speed). When the zoorun does not converge, you can test with another method. Note that the only method allowing to put constraints on parameters is "L-BFGS-B" (other methods are unconstrained). It is also possible to provide multiple methods. These are then successively tested until one converges. This allows another method to be tested automatically in case of convergence issues. For example, you can test three methods setting optim_methods=c("L-BFGS-B","Nelder-Mead","SANN")). Note that convergence is considered problematic if optim returns and error or if it returns a flag of 10 or 51 (information is availble in the @optimerr slot).

`maxiter` Indicates the maximum number of iterations when estimating the parameters with the R optim function (optional argument - 100 by default). Iterations are not defined identically across methods. For instance, in one iteration of the "L-BFGS-B" method, the likelihood of the model, estimated with the forward algorithm, is evaluated multiple times. So, a value of 100 iterations is good for the "L-BFGS-B" method but larger values are required for some other algorithms. When mutliple optim_methods are provided, the same maxiter value will be used for each method. However, it is possible to provide multiple number of maxiter, one per method. For example, if you provide three methods (optim_methods=c("L-BFGS-B","Nelder-Mead","SANN")), you can provide three values maxiter=c(100,1000,10000).

`minmix` This indicates the minimal value for the mixing coefficients. By default it is set to 0 with the classical mixkl and kl models (unconstrained). However, when using the step option or the "Interval" HBDclass, the values is set to 1e-16 to avoid numerical problems. When running a KR model (with estimation of both rates and mixing coefficients), this constraint is only allowed with the "L-BFGS-B" method from optim.

`maxr` This indicates the maximum difference between rates of successive classes. It is an optional argument set to an arbitrarily large value (100000000). Adding such constraints might slow down the speed of convergence and we recommend to run first without this constraint (constraints are only allowed with the "L-BFGS-B" method from optim).

`kinpairs` A matrix with two columns, indicating the pairs of individuals being analyzed. This information is required for a zookin analysis.

`RecTable` This is an optimal parameter indicating whether a finite number of genetic distances are used (false by default). This function can be used only with the "Interval" HBDclass. The "Interval" option can be slow, in particular if large "intervals" of generations are defined. To speed up computations, some variables are precomputed for a finite set of genetic distances, select to cover a broad range of possible values. The real genetic distance between two genetic markers is then replaced by the closest value in the table (the difference between the true and used genetic distances being also lower than 10"%").

`additional_zmix`

This option enables additional starting values for the mixing coefficients to be provided. These are provided as a list of vectors of starting values. Each vector must have the same dimensions as the @mix_coef slot in the zoomodel. This option is only useful when some analyses do not converge (for one individual or a

pair of haplotypes). Convergence is considered problematic if optim returns and error or if it returns a flag of 10 or 51 (information is availble in the @optimerr slot). If additional starting values for the mixing coefficients are provided, the analyses will automatically be relaunched with these alternative starting values. Sometimes, this solves the issue. When multiple optim_methods and additional mixing coefficients are provided, RZooRoH will work per method: it will start the first optim_method with the default starting values of the mixing coefficients and then with the additional mixing coefficients, before moving to the second method. Estimation stops as soon as one combination converges.

`trim_ad` This is an option still under evaluation (for testing only)

`hemiprob` This is an option still under evaluation (for testing only)

Value

The function return a kinres object with several slots accesses by the "@" symbol. It is very similar to the zoores object. The three main results are kinres@realized (the matrix with partitioning of the genome in different IBD classes for each pair of individual), These values are scaled like coancestry coefficients and represent the probability that two alleles sampled in the two individuals are IBD (and in that length class). The sum of the realized values in all IBD classes correspond to the kinship. kinres@ibdseg (a data frame with identified IBD segments) and zoores@ibdp (a list of matrices with IBD probabilities per SNP and per class). This gives at one marker position, the probability that two haplotypes, one sampled in each individual, are IBD (and in that length class) at that position. It corresponds also to the predicted HBD values in their future progeny.

Some slots present in zoores objects are not reported because for each pair of individuals we would then have four values. If you are interested in these values for each pair of haplotypes, you can run zoorun with the ibd option. These slots are @mixc, @modlik, @modbic, @niter.

Here is a list with all the slots and their description:

1. kinres@npairs the number of pairs of individuals in the analysis,
2. kinres@ids a matrix containing the numbers of the analyzed pairs of individuals (their position in the data file),
3. kinres@krates the rates for the exponential distributions associated with each IBD or non-IBD class for all individuals,
4. kinres@realized a matrix with estimated realized IBD per layer (columns) for each pair of individuals (rows). These values are obtained with the Forward-Backward algorithm - fb option - and averaging over the four possible haplotype pairs (between the two individuals),
5. kinres@ibdp a list of matrices with the local probabilities of IBD in different layers (computed for every class and every pair of individuals). The IBD probability is the probability that two alleles sampled at that locus in the two individuals (one in each) are IBD with respect to that layer (e.g., the IBD segment must correspond to the layer). Each matrix has one row per layer / class and one column per snp. To access the matrix for pair (of individuals) i, use the brackets "[[]]", for instance kinres@ibdp[[i]],
6. kinres@ibdseg a data frame with the list of identified IBD segments with the Viterbi algorithm (the columns are the haplotype pair number, the chromosome number, the first and last SNP of the segment, the positions of the first and last SNP of the segment, the number of SNPs in the segment, the length of the segment, the IBD state of the segment),

7. kinres@sampleids is a vector with the names of the pair of samples (when provided in the zooin object through the zoodata function). The ids from the two individuals are separated by a "_".
8. kinres@haplotype_ids is a vector with the information on haplotype pairs in the analysis and used in the ibdseg table.
9. kinres@optimerr is a vector indicating whether optim ran with or without error (0 indicates successful completion / 1 indicates that the iteration limit has been reached / 10 indicates degeneracy of the Nelder–Mead simplex.51 and 52 indicate warnings from the #' "L-BFGS-B" method / 99 indicates numerical problem). See the optim R #' function for more details. In a kinship analysis, the zoorun model is run #' four times, with four values for optimerr. The maximum value is returned here. If this value is 0, this indicates that the run did not report any problem for each of the four pairs.

`zoomodel` *Define the model for the RZooRoH*

Description

Help the user to create a model for RZooRoH, including default parameters. The output is a zmodel object necessary to run RZooRoH.

Usage

```
zoomodel(
  predefined = TRUE,
  K = 10,
  mix_coef = rep(0, K),
  base_rate = 2,
  krates = rep(0, K),
  err = 0.001,
  seqerr = 0.001,
  step = FALSE,
  XM = rep(0, K),
  HBDclass = "SingleRate"
)
```

Arguments

`predefined`	Logical (TRUE or FALSE) to define whether rates of HBD and non-HBD-classes will be estimated by the model ("kl" model) or whether the rates of these classes are fixed and pre-defined by the user ("mixkl" model). The default value is "predefined = TRUE".
`K`	The number of layers with the nested 1R model implemented in 2022, this represents thus the number of HBD classes. By default, K is set to 10 but this is not optimal for all data sets. Hence, we recommend to the users to select their own value. If K is set to 1 and rates are estimated, RZooRoH will use the same rate for the HBD and the non-HBD class (so-called 1R model).

`mix_coef` The starting value for the mixing coefficients in each layer. The mixing coefficients determine the frequency of the segments from different classes, they determine the probability to start a new HBD segment in a given layer (after reaching that layer after a coancestry change). The mixing coefficients can also be interpreted as the rate of inbreeding in their respective layer and should take values between 0 and 1. The function expects K mixing coefficients (e.g. the number of layers). The default values are 0.01 for HBD classes (smaller values may be better for large values of K). In case the parameters are not estimated (e.g. when running the forward-backward or the Viterbi algorithm alone), these are the mixing coefficients used by the RZooRoH model. Is it possible to force some mixing coefficient to take the same value with the step option. In that case, the expected number of mixing coefficients might be different.

`base_rate` is a integer used to define the rates of successive layers or HBD classes (see krates below). This parameter is most useful when using a model with predefined rates. The rate of each HBD class will be equal to the base_rate raised to the exponent k (the class number). The non-HBD class will have the same rate as the last HBD class. For instance, with a base_rate of 2 and four layers, we have the following rates: 2, 4, 8, 16 and 16. Similarly, with a base_rate of 10 and three layers, we have 10, 100, 1000 and 1000. With this method, more HBD classes are defined for more recent ancestors (for which we have more information to estimate R) and less for ancient HBD classes (it doesn't make sense to try to distinguish R = 1000 from R = 1010). In addition, since the expected length of HBD segments is expected to be 1/R, the ratio between successive expected HBD lengths remains the same. This ratio also determines the ability of the model to distinguish segments from distinct classes. By keeping the ratio constant, the aptitude to discriminate between HBD classes is also constant. In addition, this method allows to cover a wide range of generations in the past, with more emphasis on recent ancestors. The default value for the base_rate is 2.

`krates` Is an array with a rate in each layer. The function expects K positive rates. These rates are parameters of the exponential distribution that together with the distance in centimorgans defines the probability to end a HBD segments between two markers (they correspond to the rates of coancestry changes). Each layer / HBD class has a distinct rate. Therefore, the expected length of HBD classes is defined by the rates. The expected length is equal to 1/R. These krates are associated with the age of the common ancestor of the HBD segment. The rate is approximately equal to the size of the inbreeding loop (twice the number of generations to the common ancestor) when the map is given in Morgans. This is not an exact dating but an approximation, in simplified conditions. By default, the rates are defined by the base_rate parameter (2, 4, 8, 16, ...).

`err` Indicates the error term, the probability to observe an heterozygous genotype in a HBD segment. The genotype could be heterozygous due to a mutation occuring on the path to the common ancestor. It can also be associated with a genotype calling error or a technical error. In case GP or GL formats are used (with genotyped probabilities or phred scores) or when an AD format is used (based on read counts), this error term still represents the probability to observe an heterozygous genotype in a HBD segment. When an heterozygous genotype was called with a probability equal to 1.00, this heterozygosity in an

HBD track might be associated to a mutation or to errors not accounted for by the model used to estimate the genotype probabilities (e.g., GATK). The emission probability to observe a heterozygous genotype in an HBD class will never go below the error term. The default value is 0.001.

`seqerr` This parameters is used only with the AD format. In the AD format the user gives the number of reads for both alleles. A simple model is then used to estimate the genotype probabilities based on the read counts. In that model, the seqerr represents the probability to have a sequencing error in one read. The default value is 0.001.

`step` Logical (TRUE or FALSE). This allows to use a step function to model the mixing coefficients. First a model with many layers is defined. For example, one layer for each even number (50 layers from 2 to 100 for example). Although such a model allows a finer resolution, it requires the estimation of many parameters (probably without enough information to estimate precisely all mixing coefficients). Therefore, we propose a stepfunction that forces mixing coefficients to be constant for several consecutive layers (by blocks of layers). When this option is used, K represents the total number of fitted layers, all their rates must be defined. Regarding the mixing coefficients, one mixing coefficient must be defined per step (block). An incidence matrix relating the full set of mixing coefficients from HBD classes from all layers to the steps or blocks must be provided.

`XM` Is an incidence matrix relating the full set of mixing coefficients to the reduced set of mixing coefficients (corresponding to steps or blocks).

`HBDclass` By default, this value is set to "SingleRate". This means that the layer is defined by a single rate (like an average or global rate for that layer). With the option "Interval", one rate is first defined for every past generation. If we assume discrete past generations (1, 2, 3, 4, 5, ...), the corresponding rates would approximately be equal to 2, 4, 6, ... We insist that this is a crude approximation. However, it allows easier interpretation of the results, for instance by understanding better how many generations are approximately captured per layer. One layer corresponds then to multiple "generations" and is defined as an interval going from generations g1 to g2. In this approach, we use only discrete generations and start at generation 1. The values in the krates vector correspond then to the last generation in each layer. For example, setting krates to (10,20,50,100) would define four layers from generations 1 to 10, 11 to 20, 21 to 50 and 51 to 100. To obtain the corresponding rates, these values are multiplied by two (inside the function). The rates used by ZooRoH would then go from 2 to 20, 22 to 40, 42 to 100 and 102 to 200.

Value

The function return an object that defines a model for RZooRoH and including the following elements: zmodel@typeModel equal to "kl", "mixkl" or "step_mixkl" according to the selected model, zmodel@mix_coef an array with mixing coefficients, zmodel@krates an array with the rates of the HBD classes, zmodel@err the parameter defining the probability to observe an heterozygous genotype in an HBD class, and zmodel@seqerr the parameter defining the probability of sequencing error per read, zmodel@XM the incidence matrix relating individual layers to blocks or steps, zmodel@typeClass indicating the type of HBD classes,

Examples

```
# To define a the default model, with 10 layers (10 HBD and 1 non-HBD class)
# and with pre-defined rates for HBD classes with a base of 2 (2, 4, 8, ...):

Mix10L <- zoomodel()

# To see the parameters of the defined model, just type:

Mix10L

# To define a model with four layers and using a base of 10 to define
# rates (10, 100, 1000, ...):

Mix4L <- zoomodel(K=4,base=10)

# To define a model with two classes, with estimation of rates for HBD classes
# and starting with a rate 10:

my.mod1R <- zoomodel(predefined=FALSE,K=1,krates=c(10))
```

`zooplot_hbdseg` *Plot HBD segments identified with the ZooROH model*

Description

Plot HBD segments identified with the ZooRoH model for one or several populations.

Usage

```
zooplot_hbdseg(
  input,
  chr = NULL,
  coord = NULL,
  minlen = 0,
  cols = NULL,
  plotids = TRUE,
  toplot = NULL,
  randomids = FALSE,
  nrandom = (rep(10, length(input))),
  seed = 100
)
```

Arguments

`input`	a named list with one or several zres objects obtained after running zoorun. The zres objects are the output of the zoorun function. For instance, putting

	list(name1 = zres1, name2 = zres2). The function will then use the names in the plot (in case several zres objects are used).
chr	the number of the chromosome where we are looking for HBD segments. This chromosome number refers to the position of the chromosome in the list of all chromsomes present in the input genotype data.
coord	a vector with the start and end position (in bp) of the region to plot.
minlen	the minimal length (in cM or Mb) of HBD segments to be plotted (set to 0 by default).
cols	a vector with the colors to be used for each population or zres object.
plotids	a logical indicating whether the IDs of the individuals are plotted on the graph (TRUE by default).
toplot	a list of vectors indicating the zres@ids to be plotted. This option can be used to select the individuals to plot. The list must contain one vector per population or zres object. By default, all individuals are plotted.
randomids	a logical indicating whether a randomset of individuals is plotted. This option allows to reduce the number of individuals in the plot. The option can not be used simultaneously with the toplot option. By default, randomids is FALSE.
nrandom	a vector indicating the number of individuals to be randomly sampled per population or per zres object when randomids is TRUE. By default, we select 10 individuals per zres object. This vector must have the same length as the input list.
seed	a value for the random seed used to sample individuals to plot (when the randomids option is TRUE).

Value

The function plots the HBD segments identified in the region, using different colors for different zres object. Each line represents a different individual.

---

zooplot_individuals *Plot individual curves with proportion of the genome in each HBD class or cumulated proportion in HBD classes with rates smaller than a threshold.*

Description

For each individual, the function plots the mean percentage of the genome in different HBD classes or the inbreeding coefficient obtained by summing autozygosity associated with HBD classes with a rate lower or equal to a threshold (e.g., including all HBD classes with longer and more recent HBD segments than a selected threshold).

Usage

```
zooplot_individuals(input, cumulative = TRUE, toplot = NULL, ncols = 2)
```

Arguments

input	a named list with one or several zres objects obtained after running zoorun. The zres objects are the output of the zoorun function. For instance, putting list(name1 = zres1, name2 = zres2). The function will then use the names in the plot (in case several zres objects are used).
cumulative	a logical indicating whether individual autozygosity is plotted per class (FALSE) or summed over all HBD class with a rate smaller than a value (these cumulated values are obtained for every rate defined in the model). By default, this value is TRUE. When FALSE, the percentages correspond to the individual genome-wide probabilities of belonging to each HBD-class or to the fraction of the genome in an autozygosity class. When TRUE, we obtain the probability of belonging to an HBD class with a rate smaller or equal than a threshold (here we use the pre-defined rates of the model as thresholds), averaged over the whole genome for each individual. This corresponds to report individual genomic inbreeding coefficients estimated with respect to different base populations obtained by selecting different thresholds T that determine which HBD classes are considered in the estimation of the genomic inbreeding coefficient (setting the base population approximately 0.5 * T generations ago).
toplot	A list of vectors indicating the zres@ids to be plotted. This option can be used to select the individuals to plot. The list must contain one vector per population or zres object. By default, all individuals are plotted.
ncols	when several populations are plotted, ncols determines how many results (graphs) are plotted per row.

Value

The function plots either the individual proportions of the genome associated with different HBD classes or individual genomic inbreeding coefficients estimated with respect to different base populations (from young to older). With both option, the average values are plotted in red.

---

zooplot_partitioning *Plot the partitioning of the genome in different HBD classes for each individual*

Description

Plot the partitioning of the genome in different HBD classes for each individual

Usage

```
zooplot_partitioning(
  input,
  cols = NULL,
  plotids = TRUE,
  toplot = NULL,
  randomids = FALSE,
```

```
  nrandom = NULL,
  seed = 100,
  ylim = c(0, 1),
  border = TRUE,
  nonhbd = TRUE,
  vertical = FALSE
)
```

Arguments

input	a named list with one or several zres objects obtained after running zoorun. The zres objects are the output of the zoorun function. For instance, putting list(name1 = zres1, name2 = zres2). The function will then use the names in the plot (in case several zres objects are used).
cols	A vector with the colors to be used for each class in the model.
plotids	A logical indicating whether the IDs of the individuals are plotted on the graph (TRUE by default).
toplot	A list of vectors indicating the zres@ids to be plotted. This option can be used to select the individuals to plot. The list must contain one vector per population or zres object. By default, all individuals are plotted.
randomids	A logical indicating whether a randomset of individuals is plotted. This option allows to reduce the number of individuals in the plot. The option can not be used simultaneously with the toplot option. By default, randomids is FALSE.
nrandom	A vector indicating the number of individuals to be randomly sampled per population or per zres object when randomids is TRUE. By default, we select 10 individuals per zres object. This vector must have the same length as the input list.
seed	A value for the random seed used to sample individuals to plot (when the randomids option is TRUE).
ylim	The limits of the y-axis.
border	Whether a border is plotted around each block of the barplot or not. When set to FALSE, it allows to get a less dense plot when many individuals are plotted.
nonhbd	Whether the a border is plotted around the non-hbd contribution. When set to FALSE, it allows to get a less dense plot when many individuals are plotted.
vertical	Whether the populations or zres labels are printed vertically or not.

Value

Individuals are presented with stacked barplots. Each vertical stack of bars represents one individual. Each class is represented with a bar of a different color. The height of the bar represents the proportion associated with the corresponding class. The total height of the stack is the total autozygosity.

`zooplot_prophbd` *Plot proportion of the genome associated with different HBD classes*

Description

Plot the mean percentage of the genome in different HBD classes or the inbreeding coefficient obtained by summing autozygosity associated with HBD classes with a rate lower or equal to a threshold (e.g., including all HBD classes with longer and more recent HBD segments than a selected threshold).

Usage

```
zooplot_prophbd(input, cols = NULL, style = "barplot", cumulative = FALSE)
```

Arguments

`input`	a named list with one or several zres objects obtained after running zoorun. The zres objects are the output of the zoorun function. For instance, putting list(name1 = zres1, name2 = zres2). The function will then use the names in the plot (in case several zres objects are used).
`cols`	a vector with the colors to be used for each population or zres object.
`style`	select "barplot", "lines" or "boxplot" for the graphic styles. Boxplot can be used with a single zres file or population.
`cumulative`	a logical indicating whether mean autozygosity is estimated per class (FALSE) or summed over all HBD class with a rate smaller than a value (these cumulated values are obtained for every rate defined in the model). By default, this value is FALSE. When FALSE, the percentages correspond to the mean individual genome-wide probabilities of belonging to each HBD-class or to the fraction of the genome in an autozygosity class. When TRUE, we obtain the mean probability of belonging to an HBD class with a rate smaller or equal than a threshold (here we use the pre-defined rates of the model as thresholds), averaged over the whole genome and all individuals. This corresponds to report mean genomic inbreeding coefficients estimated with respect to different base populations obtained by selecting different thresholds T that determine which HBD classes are considered in the estimation of the genomic inbreeding coefficient (setting the base population approximately 0.5 * T generations ago).

Value

The function plots either the average proportion of the genome associated with different HBD classes or the average genomic inbreeding coefficient estimated with respect to different base populations (from young to older).

zoorun *Run the ZooRoH model*

Description

Apply the defined model on a group of individuals: parameter estimation, computation of realized autozygosity and homozygous-by-descent probabilities, and identification of HBD segments (decoding). It is also possible to apply the model to a pair of phased haplotypes with the 'ibd' option.

Usage

```
zoorun(
  zoomodel,
  zooin,
  ids = NULL,
  parameters = TRUE,
  fb = TRUE,
  vit = TRUE,
  localhbd = FALSE,
  nT = 1,
  optim_methods = "L-BFGS-B",
  maxiter = 100,
  minmix = 1,
  maxr = 1e+08,
  ibd = FALSE,
  ibdpairs = NULL,
  haploid = FALSE,
  RecTable = FALSE,
  additional_zmix = NULL,
  trim_ad = FALSE,
  hemiprob = 0
)
```

Arguments

zoomodel	A valid zmodel object as defined by the zoomodel function. The model indicates whether rates of exponential distributions are estimated or predefined, the number of classes, the starting values for mixing coefficients and rates, the error probabilities. See "zoomodel" for more details.
zooin	A valid zdata object as obtained by the zoodata function. See "zoodata" for more details.
ids	An optional argument indicating the individual (its position in the data file) that must be proceeded. It can also be a vector containing the list of numbers that must be proceeded. By default, the model runs for all individuals.
parameters	Specifies whether the parameters are estimated by optimization with the L-BFGS-B method (default) from the optim function (optional argument - true

by default). If the user doesn't want to estimate the parameters he must set parameters=FALSE. In that case, the forward-backaward and Viterbi algorithms are run with the provided parameters.

`fb` A logical indicating whether the forward-backward algorithm is run (optional argument - true by default). The Forward-Backward algorithm estimates the local probabilities to belong to each HBD or non-HBD class. By default, the function returns only the HBD probabilities for each class, averaged genome-wide, and corresponding to the realized autozygosity associated with each class. To obtain HBD probabilities at every marker position, the option localhbd must be set to true (this generates larger outputs).

`vit` A logical indicating whether the Viterbi algorithm is run (optional argument - false by default). The Viterbi algorithm performs the decoding (determining the underlying class at every marker position). Whereas the Forward-Backward algorithms provide HBD probabilities (and how confident a region can be declared HBD), the Viterbi algorithm assigns every marker position to one of the defined classes (HBD or non-HBD). When informativity is high (many SNPs per HBD segments), results from the Forward-Backward and the Viterbi algorithm are very similar. The Viterbi algorithm is best suited to identify HBD segments. To estimate realized inbreeding and determine HBD status of a position, we recommend to use the Forward-Backward algorithm that better reflects uncertainty.

`localhbd` A logical indicating whether the HBD probabilities for each individual at each marker are returned when using the Forward-Backward algorithm (fb option). This is an optional argument that is false by default. In some cases, the ouput is large (memory or storage requirements!). It returns probability for each postion x HBD class x individual combination.

`nT` Indicates the number of parallel processes (workers) used when running RZooRoH in parallel (optional argument; one process by default, i.e. no parallelization). Parallelization relies on process forking and is limited to a single compute node: when submitting jobs on a cluster, make sure to request nT CPU cores on the same node (e.g. via –cpus-per-task on SLURM), not distributed across multiple nodes.

`optim_methods` Indicates which method the optim R function will use to estimate the parameters of the model ("L-BFGS-B" by default). The possible methods are "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent". Type "? optim" to have more information. In our experience, the "L-BFGS-B" method works well but the method achieving the best likelihood is variable (according to the data sets, the model, the priors, the constraints). The same goes for the efficiency (speed). When the zoorun does not converge, you can test with another method. Note that the only method allowing to put constraints on parameters is "L-BFGS-B" (other methods are unconstrained). It is also possible to provide multiple methods. These are then successively tested until one converges. This allows another method to be tested automatically in case of convergence issues. For example, you can test three methods setting optim_methods=c("L-BFGS-B","Nelder-Mead","SANN")). Note that convergence is considered problematic if optim returns and error or if it returns a flag of 10 or 51 (information is availble in the @optimerr slot).

`maxiter` Indicates the maximum number of iterations when estimating the parameters with the R optim function (optional argument - 100 by default). Iterations are not defined identically across methods. For instance, in one iteration of the "L-BFGS-B" method, the likelihood of the model, estimated with the forward algorithm, is evaluated multiple times. So, a value of 100 iterations is good for the "L-BFGS-B" method but larger values are required for some other algorithms. When mutliple optim_methods are provided, the same maxiter value will be used for each method. However, it is possible to provide multiple number of maxiter, one per method. For example, if you provide three methods (optim_methods=c("L-BFGS-B","Nelder-Mead","SANN")), you can provide three values maxiter=c(100,1000,10000).

`minmix` This indicates the minimal value for the mixing coefficients. By default it is set to 0 with the classical mixkl and kl models (unconstrained). However, when using the step option or the "Interval" HBDclass, the values is set to 1e-16 to avoid numerical problems. When running a KR model (with estimation of both rates and mixing coefficients), this constraint is only allowed with the "L-BFGS-B" method from optim.

`maxr` This indicates the maximum difference between rates of successive classes. It is an optional argument set to an arbitrarily large value (100000000). Adding such constraints might slow down the speed of convergence and we recommend to run first without this constraint (constraints are only allowed with the "L-BFGS-B" method from optim).

`ibd` A logical indicating whether the function will be used to compute IBD between pairs of phased haplotypes instead of HBD within individuals. In that case, the user must provide a matrix with the pairs of haplotypes that will be analyzed. This option can only be used if phased data are provided as input with the zformat set to "vcf" or "haps". This is an optional parameter set to false by default.

`ibdpairs` A matrix with four columns, indicating the pair of haplotypes being analyzed in an IBD analysis. Haplotypes are indicated by two columns, one column for the id of the individuals and a second column for the haplotype number within individual (1 or 2). The first and third columns indicate the id of the individuals carrying the first and second haplotype, respectively. The second and four columns indicates the haplotype numbers within the first and second individuals, respectively. With haploid data, the matrix must have only two columns indicating the number of the first and second haplotypes from the pairs. This is an optional parameter, the matrix must be provided only when the ibd option is true.

`haploid` This is an optional parameter indicating whether haplotypes belong to an haploid organisms or chromosome (false by default). It can be used only in combination with the 'ibd' option and requires phased data as input with the zformat set to "haps". When haploid is true, then the ibdpairs matrix has only two columns indicating simply the haplotype numbers. When haploid is true the number of haplotypes can be uneven while even numbers are required when haploid is set to false.

`RecTable` This is an optional parameter indicating whether a finite number of genetic distances are used (false by default). This function can be used only with the "Interval" HBDclass. The "Interval" option can be slow, in particular if large "in-

tervals" of generations are defined. To speed up computations, some variables are precomputed for a finite set of genetic distances, selected to cover a broad range of possible values. The real genetic distance between two genetic markers is then replaced by the closest value in the table (the difference between the true and used genetic distances being also lower than 10"%"). Note that this precomputed lookup table covers distances between consecutive markers up to 1000 cM (after applying the 'RecRate' parameter, if used). Distances between adjacent markers exceeding this value are not expected under realistic genetic maps but should be avoided.

`additional_zmix`
: This option enables additional starting values for the mixing coefficients to be provided. These are provided as a list of vectors of starting values. Each vector must have the same dimensions as the @mix_coef slot in the zoomodel. This option is only useful when some analyses do not converge (for one individual or a pair of haplotypes). Convergence is considered problematic if optim returns and error or if it returns a flag of 10 or 51 (information is availble in the @optimerr slot). If additional starting values for the mixing coefficients are provided, the analyses will automatically be relaunched with these alternative starting values. Sometimes, this solves the issue. When multiple optim_methods and additional mixing coefficients are provided, RZooRoH will work per method: it will start the first optim_method with the default starting values of the mixing coefficients and then with the additional mixing coefficients, before moving to the second method. Estimation stops as soon as one combination converges.

`trim_ad`
: This is an option still under evaluation (for testing only)

`hemiprob`
: This is an option still under evaluation (for testing only)

Value

The function return a zoores object with several slots accesses by the "@" symbol. The three main results are zoores@realized (the matrix with partitioning of the genome in different HBD classes for each individual), zoores@hbdseg (a data frame with identified HBD segments) and zoores@hbdp (a list of matrices with HBD probabilities per SNP and per class).

Here is a list with all the slots and their description:

1. zoores@nind the number of individuals in the analysis,
2. zoores@ids a vector containing the numbers of the analyzed individuals (their position in the data file). When analyzing IBD between pairs of haplotypes, the number correspond to position in the list of target IBD pairs (their position in the sampleids list),
3. zoores@mixc the (estimated) mixing coefficients per class for all individuals (related to inbreeding rates per layer or generations),
4. zoores@krates the (estimated) rates for the exponential distributions associated with each HBD or non-HBD class for all individuals (corresponding to the rates of coancestry changes),
5. zoores@niter the number of iterations for estimating the parameters - the number of calls of the forward algorithm (per individual),
6. zoores@modlik a vector containing the likelihood of the model for each individual,
7. zoores@modbic a vector containing the value of the BIC for each individual,

8. zoores@realized a matrix with estimated realized autozygosity per HBD class (columns) for each individual (rows). These values are obtained with the Forward-Backward algorithm - fb option). In case of IBD, the matrices contain realized IBD probabilities for pairs of haplotypes,
9. zoores@hbdp a list of matrices with the local probabilities to belong to an underlying hidden state (computed for #' every class and every individual). Each matrix has one row per class and #' one column per snp. To access the matrix from individual i, use the #' brackets "[[]]", for instance zoores@hbdp[[i]]. In case of IBD, the matrices contain local IBD probabilities for a pair of haplotypes,
10. zoores@hbdseg a data frame with the list of identified HBD segments with the Viterbi algorithm (the columns are the individual number, the chromosome number, the first and last SNP of the segment, the positions of the first and last SNP of the segment, the number of SNPs in the segment, the length of the segment, the HBD state of the segment). In case of IBD, the first column indicates the number of the pair of haplotypes (its position in the sampleids list),
11. zoores@optimerr a vector indicating whether optim ran with or without error (0 indicates successful completion / 1 indicates that the iteration limit has been reached / 10 indicates indicates degeneracy of the Nelder–Mead simplex, 51 and 52 indicate warnings from the "L-BFGS-B" method / 99 indicates numerical problem). See the optim R function for more details.
12. zoores@sampleids is a vector with the names of the samples (when provided in the zooin object through the zoodata function). When an IBD analysis is run, it indicates the pairs of haplotypes separated by "_". For diploid individuals, it combines individual ID, haplotype ID (1 or 2) for the two individuals.

Examples

```
# Start with a small data set with six individuals and external frequencies.
freqfile <- (system.file("exdata","typsfrq.txt",package="RZooRoH"))
typfile <- (system.file("exdata","typs.txt",package="RZooRoH"))
frq <- read.table(freqfile,header=FALSE)
typ <- zoodata(typfile,supcol=4,chrcol=1,poscol=2,allelefreq=frq$V1)
# Define a model with two HBD classes with rates equal to 10 and 100.
Mod2L <- zoomodel(K=2,base_rate=10)
# Run the model on all individuals.
typ.res <- zoorun(Mod2L, typ)
# Observe some results: likelihood, realized autozygosity in different
# HBD classes and identified HBD segments.
typ.res@modlik
typ.res@realized
typ.res@hbdseg
# Define a model with one HBD and one non-HBD class and run it.
Mod1R <- zoomodel(K=1,predefined=FALSE)
typ2.res <- zoorun(Mod1R, typ)
# Print the estimated rates and mixing coefficients.
typ2.res@krates
typ2.res@mixc
# Get the name and location of a second example file and load the data:
myfile <- (system.file("exdata","genoex.txt",package="RZooRoH"))
ex2 <- zoodata(myfile)
# Run RZooRoH to estimate parameters on your data with the 1 HBD and 1 non-HBD
# class (parameter estimation with optim).
```

```
my.mod1R <- zoomodel(predefined=FALSE,K=1,krates=c(10))
my.res <- zoorun(my.mod1R, ex2, fb = FALSE, vit = FALSE)
# The estimated rates and mixing coefficients:
my.res@mixc
my.res@krates
# Run the same model and run the Forward-Backward alogrithm to estimate
# realized autozygosity and the Viterbi algorithm to identify HBD segments:
my.res2 <- zoorun(my.mod1R, ex2)
# The table with estimated realized autozygosity:
my.res2@realized
# Run a model with 3 layers (3 HBD classes / 1 non-HBD class) and estimate
# the rates of HBD classes with one thread:
my.mod3L <- zoomodel(predefined=FALSE,K=3,krates=c(16,64,256))
my.res3 <- zoorun(my.mod3L, ex2, fb = FALSE, vit = FALSE, nT =1)
# The estimated rates for the 3 classes and the 20 individuals:
my.res3@krates
# Run a model with 4 layers and predefined rates.
# The model is run only for a subset of four selected individuals.
# The parameters are estimated, the Forward-Backward alogrithm is ued to
# estimate realized autozygosity and the Viterbi algorithm to identify
# HBD segments. One thread is used.
mix4L <- zoomodel(K=4,base=10)
my.res4 <- zoorun(mix4L,ex2,ids=c(7,12,16,18), nT = 1)
# The table with all identified HBD segments:
my.res4@hbdseg
```

Index

∗ **datasets**
  BBB_NMP_ad_subset, 3
  BBB_NMP_GP_subset, 4
  BBB_NMP_pl_subset, 4
  BBB_PE_gt_subset, 5
  BBB_samples, 6
  genoex, 9
  genosim, 9
  hbd1, 10
  hbd2, 11
  kintaf_mix4l, 11
  map1, 12
  rara_mix10l, 15
  soay_mix10l, 18
  typs, 18
  typsfrq, 19
  wilt_mix10l, 20

BBB_NMP_ad_subset, 3
BBB_NMP_GP_subset, 4
BBB_NMP_pl_subset, 4
BBB_PE_gt_subset, 5
BBB_samples, 6

ComputeFseg, 6
ComputeFsegAllT, 7
cumhbd, 8
cumkin, 8

genoex, 9
genosim, 9

hbd1, 10
hbd2, 11

kintaf_mix4l, 11

map1, 12
merge_kinres, 12
merge_zres, 13

predhbd, 13
probhbd, 14

rara_mix10l, 15
realized, 16
rohbd, 16

soay_mix10l, 18

typs, 18
typsfrq, 19

update_kinres, 19
update_zres, 20

wilt_mix10l, 20

zoodata, 21
zookin, 25
zoomodel, 29
zooplot_hbdseg, 32
zooplot_individuals, 33
zooplot_partitioning, 34
zooplot_prophbd, 36
zoorun, 37