The RZooRoH package

Tom Druet, Natalia Forneris, Amandine Bertrand, Naveen Kadri & Mathieu Gautier

2025-06-05

The `RZooRoH` package offers functions to identify Homozygous-by-Descent (HBD) segments and to estimate individual autozygosity (or inbreeding coefficients). HBD segments are created when an individual inherits two copies of the same chromosome segment from an ancestor. The two copies are inherited through different pathways, one copy was inherited from the mother, the second from the father. This happens in presence of inbreeding, when parents are related (they share a common ancestor). In absence of mutations, this creates long runs of homozygous genotypes (RoH). The length of the segments depends on the number of generations from the individual to the common ancestor (the generations from the two distinct paths must be summed) or the size of the inbreeding loop. Most often, multiple ancestors contribute to the autozygosity of an individual. These ancestors trace back to different generations in the past and are hence associated with inbreeding loops of variable size. As a result, the length of the HBD segments is expected to vary. Therefore, the model used in the package relies on multiple HBD classes related to the age of the segments (longer segments and smaller rates for recent autozygosity / recent common ancestor). The model has been extended to analysis of identity-by-descent (IBD) between chromosomes from different individuals and to kinship estimation. The `RZooRoH` package is available for most platforms (Linux, MS Windows and MacOSX) from the CRAN repository (https://CRAN.R-project.org/package=RZooRoH).

1 Installation

To install the package, open R and run:

```r
install.packages("RZooRoH")
```

2 Citations

To cite the RZooRoH package in publication, please use:

- T. Druet and M. Gautier (2017). A model-based approach to characterize individual inbreeding at both global and local genomic scales. Molecular Ecology, 26:5820-5841 (https://doi.org/10.1111/mec.14324).
- A.R. Bertrand, N.K. Kadri, L. Flori, M. Gautier and T. Druet (2019). RZooRoH: an R package to characterize individual genomic autozygosity and identify homozygous-by-descent segments. Methods Ecol Evol, 10:860–866 (https://doi.org/10.1111/2041-210X.13167).

A new model, based on succesive "layers", with improved properties has been developed and is presented in:

- T. Druet and M. Gautier (2022). A hidden Markov model to estimate homozygous-by-descent probabilities associated with nested layers of ancestors. Theor Popul Biol. 2022:145(1):38–51 (https://doi.org/10.1016/j.tpb.2022.03.001).

3 The multiple HBD classes model

3.1 The hidden Markov model with two classes (HBD *vs* non-HBD)

The model is an hidden Markov model (**HMM**) describing the genome of an individual as a succession of HBD and non-HBD segments. The unobserved HBD status is evaluated at each marker position. The two HBD status (the position is HBD or is not HBD) are not observed and referred to as hidden states. To compute the probability of a specific succession of HBD and non-HBD segments, the model requires the probability to continue or end the current segment (related to so-called transition probabilities) and the probability to observe the genotypes or the reads (with sequencing data) conditionally on the HBD status (so-called emission probabilities). In the case the current segment stops, we need also to define the probability to start a new HBD or non-HBD segment:

- The length of HBD segments is exponentially distributed (Thompson, 2013). We can use this to model the probability to continue or stop a segment. The probability to end a segment between two markers separated by d Morgans is e^{-Rd}, where R is the rate of the exponential distribution. The expected length of HBD segments is then 1/R Morgans (high rates corresponding to shorter segments).
- The probability to observe a genotype depends on the HBD status, the allele frequencies, the genotyping error rate and the mutation rate. In HBD segments, genotypes are expected to be mostly homozygous with higher probabilities to observe frequent alleles. In non-HBD segments, genotypes are expected to follow Hardy-Weinberg proportions. When genotypes are not know with high confidence (e.g., with low fold-sequencing data), the HMM framework allows to use the genotype probabilities and to integrate over the three possible genotypes.
- The probability to start a new HBD or non-HBD segment will be equal to the mixing proportion. An HMM is indeed a mixture distribution and the mixing proportions defines how frequent the different hidden states are.

More details on one HBD class HMM can be found in Leutenegger (2003), Vieira et al. (2016), Narasimhan et al. (2016) and Druet and Gautier (2017; 2022).

3.2 Extending to multiple HBD classes

Assuming a single HBD class amounts to consider that all HBD segments have the same expected length. This might be interpreted biologically as considering that all the autozygosity traces back to a single ancestor or several ancestors living in the same generation. A multiple HBD classes model assumes instead that each HBD class has its own expected length and frequency allowing to fit more realistic situations where ancestors contributing to autozygosity trace back to different generations in the past. This is the case in most populations (see Druet and Gautier (2017) or Sole et al. (2017) for examples).

The multiple HBD classes HMM still describes the genome of an individual as a succession of HBD and non-HBD segments. However, the HBD segments are categorized in different classes (e.g., very long, long, normal, short or very short HBD segments). Each of the classes defines an hidden state. The principle is the same as for the first HMM except that several classes of HBD segments are defined (based on their length). This model is fitted by successive layers, starting with the first layer corresponding to the most recent ancestors. In that layer, the genome is modeled with the model previously described (in section 3.1). However, the non-HBD segments can in turn be modeled as a mosaic of HBD and non-HBD segments corresponding to more remote ancestors. These segments are thus expected to be shorter (and the class has a higher rate parameter). This modelling can be repeated for multiple layers, each layer will have its own rate (with increasing values) and its own mixing coefficient (defining the rate of inbreeding inside the layer). The model relies on similar probabilities as before:

- The length of HBD segments is exponentially distributed. Now, each class has its own rate R_k, where k is the class number. The probability to end a segment from class k between two markers separated by d Morgans is $e^{-R_k d}$. The expected length of HBD segments from class k is then $1/R_k$ Morgans (high rates corresponding to shorter segments).
- The probability to observe a genotype depends on the HBD status, the allele frequencies, the genotyping error rate and the mutation rate. The same emission probabilities are used for all HBD classes (see above).
- The probability to start a new HBD or non-HBD segment within a layer will be equal to the mixing proportion in that layer. Each class k has its own mixing coefficient M_k.

A full description of the model is available in Druet and Gautier (2022).

3.3 Estimating HBD probabilities and identifying HBD segments with an HMM

We presented above some elements to compute the probability of a single succession of HBD and non-HBD segments but this is not our main interest. Indeed, we want to compute the probability of a possible sequence (of states) in order to find the best sequence or to estimate probabilities by integration over all sequences. The number of possible sequences of states increases rapidly with the number of markers $nsnp$ and is equal to K^{nsnp}, where K is the number of states (classes).

Fortunately, in the HMM framework we can efficiently compute the likelihood of the data and the probability to belong to each class at each marker position with the forward-backward algorithm (Rabiner, 1989). With the forward-backward algorithm, the probabilities are obtained by integration over all possible sequences of states. As a result we obtain at a marker position, the probability that the genome of the studied individual belongs to a HBD segment from class *k*. The probabilities are obtained by integrating over all possible length of HBD segments (over all possible window sizes).

Alternatively, it is also possible to use the Viterbi algorithm (Rabiner, 1989) to identify the most likely sequence of states or succession of HBD and non-HBD segments. In that case, every marker position is assigned to one of the K classes and stretches of markers assigned to the same class form HBD segments. We do not longer obtain probabilities.

3.4 Model and results interpretation

The different HBD classes are defined by their specific rates R_k. The length of HBD segments from class k is exponentially distributed with rate R_k and mean $1/R_k$. Classes with lower rates correspond to longer HBD segments from more recent common ancestors. Therefore, different HBD classes can be interpreted as HBD segments of different groups of ancestors tracing back to different generations in the past. The rate of the class is approximately equal to the size of the inbreeding loop in generations (Druet and Gautier, 2017). So, the rate of the class is approximately equal to twice the number of generations to the common ancestor. For instance, a class with a rate R_k equal to 10 would correspond approximately to ancestors five generations ago while a class with a rate R_k equal to 100 would correspond approximately to ancestors fifty generations ago. These are off course not precise estimations of age of HBD segments but rather qualitative measures. These approximations would work best when a single group of ancestors contribute to HBD or when a few group of ancestors contribute to inbreeding and are separated by many generations.

The HMM relies on two sets of parameters, the rates R_k and the mixing coefficients M_k. The interpretation of the rates R_k has been explained in the previous paragraph. With single HBD class models, the mixing coefficient can be interpreted as the inbreeding coefficient (see Leutenegger et al. (2003)). In the multiple layers models, the mixing coefficients correspond to the inbreeding rate in that layer (see Druet and Gautier (2022) for more details).

With the forward-backward algorithm, we obtain at each marker position the probability to belong to each state. We call the probability estimated at one marker position a local probability whereas global probabilities are obtained by averaging local probabilities over the genome, they are genome-wide estimates. The HMM provides local and global probabilities to belong to each class. Global probabilities provide the contribution from each class to the genome and for HBD classes, they provide an estimate of autozygosity associated with one class. We call the estimated contribution of one HBD class as the realized autozygosity.

The sum of probabilities to belong to each HBD class provides an estimate of the total HBD probability. The sum can be local and provides hence estimates of local HBD probabilities that can be used for homozygosity/autozygosity mapping experiments (Wang et al., 2009; Leutenegger et al., 2006). If the sum is genome-wide, then its provides an estimate of total autozygosity.

Inbreeding coefficients are estimated with respect to a base population that needs to be defined by the user. The different HBD classes and their rates R_k allow to define base populations since rates are related to the “age of the ancestors” (see first paragraph of this section). By summing the global probabilities from all HBD classes with a rate smaller or equal than a threshold T ($R_k \leq T$), we obtain an estimate of the inbreeding coefficient F when the base population is set approximately $0.5{*}T$ generations in the past (a few more generations). As mentioned before, this is an approximation.

Finally, when running the Viterbi algorithm, we obtain segments associated with different HBD classes (stretches of markers assigned to the same HBD class). We can define HBD segments as segments associated with any of the HBD classes or restrict the definition to HBD classes with a rate smaller than a threshold (eventually defining the base population).

In 2025, we implemented the algorithm presented by Harris et al. (2014), which allows to speed up the decoding of coalescent HMMs. The runtimes are linear in the number of hidden states instead of quadratic. This algorithm is used for parameter estimation (calling the forward algorithm) and for the forward-backward algorithm, but is not implemented in the Viterbi (which does not benefit from the speed-up). The advantage of the algorithm increases with the number of states. It is now possible to run models with 50-100 HBD classes for some specific applications (not recommended for standard usage).

4 Differences with other approaches

4.1 Some differences with window-based approaches identifying ROH

Several approaches to identify ROH (later interpreted as HBD segments) are based on gathering information in sliding windows. With rule-based methods (McQuillan et al., 2008), stretches of genotypes fulfilling certain criteria (e.g., number of SNPs, number of heterozygous and missing genotypes, marker density, window length, marker spacing) are interpreted as HBD segments. Ideally, the criteria should be optimized for every data set (population, genotyping technology, etc.). Such an approach is implemented in PLINK (Purcell et al., 2007). In likelihood-based approaches (Broman and Weber, 1999; Pemberton et al., 2012), LOD-scores are computed to classify windows as HBD or non-HBD. The marker allele frequencies and genotyping errors are used to compute the likelihood, making the approach less sensitive to marker recruitment bias. The optimal window size is determined by selecting the smallest value resulting in a clear bi-modal distribution of LOD scores. Likelihood-based approaches should be preferred to rule-based approaches since they better account for allele frequencies and genotyping errors.

The emission probabilities of the HMM use the allele frequencies and the genotyping error rates similarly to likelihood-based ROH. Therefore, they are less sensitive to marker recruitment bias or filtering criteria (minimum MAF). Sometimes rule-based ROH are run inappropriately with monomorphic markers. In HMM, these markers are automatically non-informative. In addition, HMM use information from the genetic map (distance between markers), taking automatically into account marker density or marker spacing and being more robust to variable recombination rates along the genome. HMM also automatically explore all possible lengths of HBD segments (they do not require the definition of an optimal window size) and provide HBD probabilities. Another benefit from the HMM is that they can work with genotype probabilities or likelihoods and with irregular marker spacing (exome, GBS). Therefore, they can also efficiently handle exome or whole-genome (including low-fold) sequence data (Magi et al., 2014; Narasimhan et al., 2016; Vieira et al., 2016).

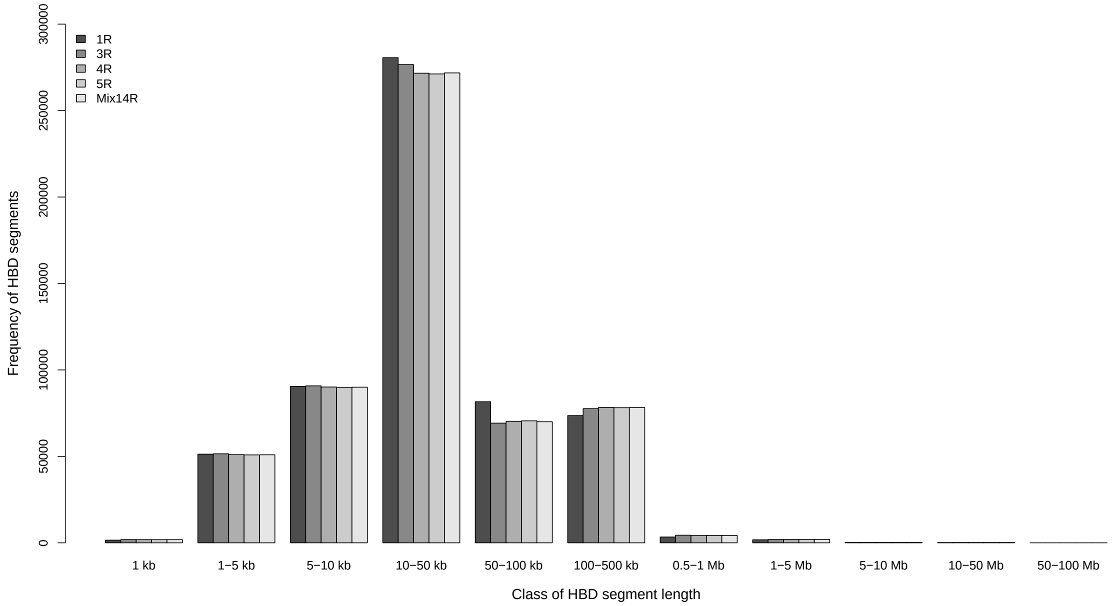


Figure 1: Distribution of length of HBD segments identified in Belgian Blue beef cattle with different models (results from Sole et al. (2017)). We observe that a model with 1 HBD class (1R) has more segments of intermediary size (10-100 kb). The model with 14 classes identifies more long segments (> 500 kb).

ROH-based approaches and HMM have been compared in a few simulations studies. As often, the simulated scenarios can influence the results (the selected marker density, a uniform marker spacing, the presence of genotyping errors or not, etc). In addition, the parameters of the ROH can be adapted to better fit the simulations. Therefore, comparisons should be interpreted with caution. Narasimhan et al. (2016) found that HMM had lower false positive rates (FPR) and false negative rates (FNR) compared to ROH estimated with PLINK. Druet and Gautier (2017) concluded that the differences between the three approaches was small when informativity was high (many SNPs per HBD segment) whereas the HMM performed better for shorter segments or at lower marker density in terms of the estimated realized inbreeding coefficient or the local autozygosity estimation (e.g. at a locus). By changing rules to call ROH (e.g., window size or number of heterozygous SNPs in ROH), it was possible to optimize the behavior of the window-based approaches. For instance, the FNR can be increased at the expense of a higher FPR by using more aggressive parameters (e.g., shorter windows).

The use of HMM is particularly valuable when information is sparser (Druet and Gautier, 2017) and HBD versus non-HBD classification is more uncertain (e.g., low fold-sequencing experiments, lower marker density, older and shorter HBD segments, biased genotyping arrays, etc.). It is also very useful when the information is more variable (variable allele frequencies, distances between markers, recombination rates, coverages, genotyping errors). For instance, Magi et al. (2014) showed that an HMM based approach outperforms PLINK when applied to whole-exome sequences data by considering the distances between consecutive SNPs. Vieira et al. (2016) demonstrated the importance to use genotype likelihoods (as integrated in some HMM approaches) instead of genotypes (as used in window-based approaches) when dealing with low-fold sequencing data. HMM estimating HBD probabilities provide information on the uncertainty associated with the inference when the information is degraded. Overall, the HMM approach is particularly beneficial in situations encountered in wild organisms including lower marker densities, low-fold sequencing data, marker recruitment bias, uneven marker spacing, variable genotyping error rates, variable recombination rates, etc. Some of these trends were confirmed in more recent studies, see for example Alemu et al. (2021), Lavanchy and Goudet (2023) or Forneris et al. (2025).

Finally, HMM present also conceptual differences with ROH-based methods since, by providing probabilities, they do not rely on the selection of various thresholds such as marker spacing, window size, minimum number of SNPs, etc. that should be re-defined for each data set. For example, in Forneris et al. (2025), the same HMM were used for data sets with different marker density or heterogenous marker spacing whereas parameters from rule-based methods needed optimization.

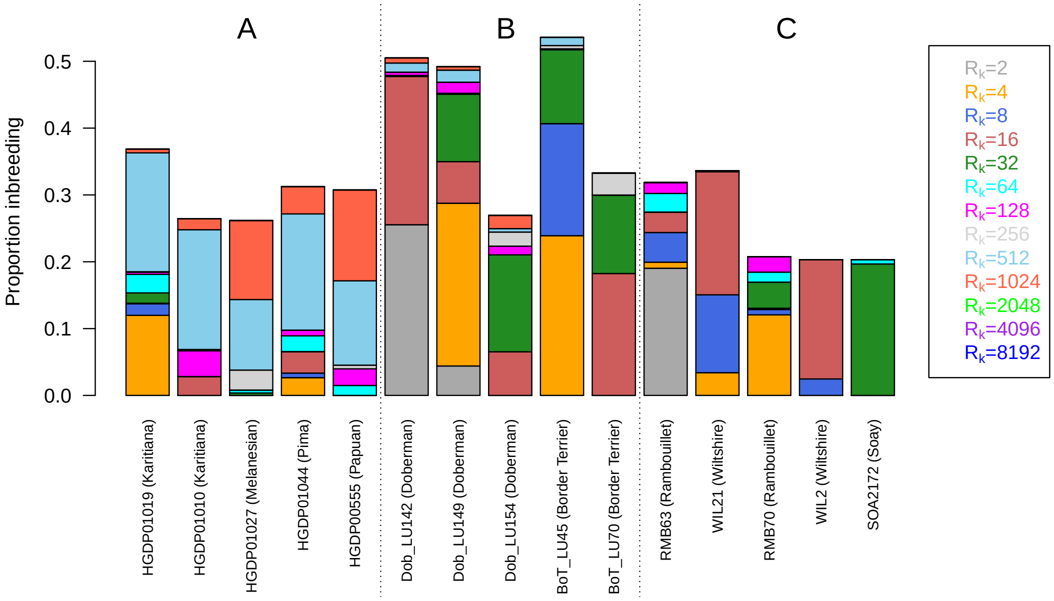


Figure 2: Estimated level of autozygosity per HBD class in five humans (A), five dogs (B) and five sheeps (C). Each colour is associated with a distinct class (defined by its rate). The heights of each colour bar represent the estimated level of autozygosity associated with the class, and the total height represents the total estimated autozygosity (results from Druet and Gautier (2017)). Three dogs have total values close to 0.5 and the method indicates that 25 percent of the genome is associated with HBD classes with rates equal to 2 or 4. These values are compatible with parent-offspring matings combined with additional autozygosity from more distance ancestors. The most inbred sheep has autozygosity levels higher than 0.30 but mainly associated with more distant ancestors (approximately 8 generations in the past). These examples illustrate that the partitioning in different HBD class might help to understand the relationship between the parents or past demographic events at the individual level.

4.2 Benefits of using multiple HBD classes

The multiple HBD classes model assumes that each HBD class has its own expected length and frequency (Druet and Gautier, 2017) allowing to fit realistic situations where ancestors contributing to autozygosity trace back to different generations in the past. These classes are related to the age of common ancestors associated with the HBD classes (see section 2.4): classes with longer segments (lower rates) correspond to more recent common ancestors and vice versa.

Here are some benefits of using multiple HBD classes:

- It results in a better fit of individual genetic data and more accurate estimations of autozygosity levels both locally and globally, particularly in complex populations. We showed through simulations that singles HBD class models might underestimate the autozygosity when multiple generations contribute to it (Druet and Gautier, 2017). Similarly, we illustrated with real cattle data that the use of single HBD class models results in lower estimates of autozygosity.
- With a single HBD class the distribution of length of identified HBD segments is more concentrated at intermediate values (Sole et al. (2017); see Figure 1). Indeed, the smallest segments are not

captured with a single class whereas long segments are fragmented in multiple smaller segments. Obtaining the correct length of HBD segments is essential to interpret the results, in particular to estimate the age of the ancestor.

- These extended models offer the possibility to reveal the recent demographic history by partitioning HBD segments in different age-related classes and to estimate the contribution of different generations in the past to the current autozygosity, at both an individual and a population scales (Druet and Gautier, 2017).
  - At individual scale, presence of long HBD segments in HBD classes with rates R_k of two to four indicates that the parents were highly related (parent-offspring matings or brother-sister matings). Age of the HBD classes and their importance help to determine the relationship between parents and the mating habits (see Figure 2 for an example). This might be particularly interesting in studies on wild populations where relationships between parents are often unknown.
  - At the population level, large contributions of a class to autozygosity indicates a reduced effective population size (Ne) at that time period possibly associated with a bottleneck or a founder effect. Similarly, low contribution to autozygosity suggests large Ne in a past period.
- The use of multiple HBD classes makes LD pruning less important because recent and ancient HBD segments are separated in distinct classes (Druet and Gautier, 2017). Indeed, we compared the results obtained with a single HBD class HMM applied with a pruning strategy (the data set with $>$ 600,000 SNPs was divided in 100 data sets with $\sim$ 6500 SNPs, 1% of the data) (Leutenegger et al., 2011) and those obtained with our model (using 14 HBD classes and without pruning). The results from both studies were consistent; for instance the autozygosity estimated by Leutenegger et al. (2011) were highly correlated to the recent autozygosity associated with the first four HBD classes (see Figure 3). More details are described in (Druet and Gautier, 2017).
- The multiple HBD class model is also more robust when using data with variable marker density such as reduced representation sequencing. In that case, small fragments of the genome have a high marker density whereas outside the fragments, the density is null. In Forneris et al. (2025), we observed that the multiple HBD class model was behaving well with such data whereas a single HBD class model was much less accurate (it might be useful to perform some LD pruning in that case).

5 Input data

5.1 Input data quality filtering

The user must filter his data before using RZooRoH. For example, it is assumed that SNPs with low call-rates, deviating strongly from Hardy-Weinberg equilibrium, with high genotyping error rates, etc. have been filtered out. RZooRoH works with bi-allelic markers and the missing genotypes must be coded correctly. The quality of the analysis depends also on the quality of the physical map or the genome assembly. When possible, it is recommended to remove regions that are more 'noisy' or not correctly positioned in the physical map. Errors in the map will break long HBD segments in shorter segments. Therefore, their length (and the associated interpretation) will be incorrect.

In addition to marker filtering, individuals with low call rates should also be excluded from the analysis.

5.2 Data format and conversion from PED or VCF files

Several data formats are accepted, including the Oxford GEN format (Marchini et al., 2007). SNPs are always organized by row and individuals per column. The first columns should include marker information and subsequent column genotype / sequence information per individual. The package runs on autosomes (e.g., no sexual chromosome) and for bi-allelic markers.

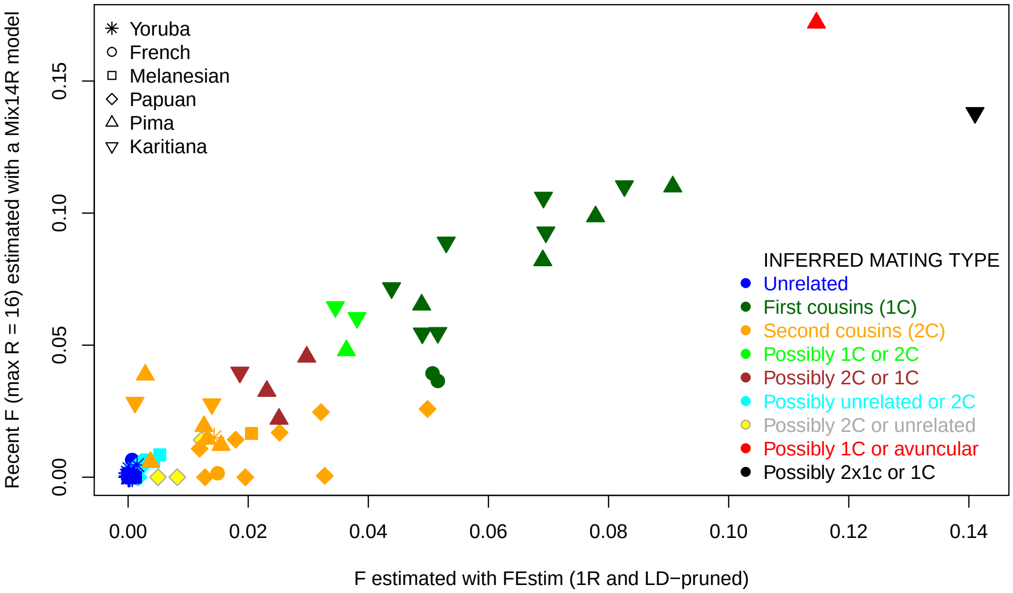


Figure 3: Comparison of individual autozygosity estimated with FEstim (1 HBD class) and a pruning strategy with recent autzoygosity classes estimated with a multiple HBD classes model (results from Druet and Gautier (2017)).

5.2.1 Fields for marker information The number of columns used to describe the markers is not fixed, by default we expect five columns with marker information (chromosome, marker name, position, first allele and second allele). Two columns are mandatory: the chromosome field and the position field. The other fields are not used. More or less columns can be provided and the order of the columns can be changed as the read function (`zoodata`) allows to specify the information.

The position field should ideally be in genetic distances. The package works with distances in cM multiplied by 1,000,000. When genetic distances are not available, physical distances in base pairs (bp) can be used, assuming 1 Mb = 1 cM as observed in some species. Internally, distances are converted to Morgans (division by 100,000,000) and rates can be interpreted as a function of number of generations to the ancestor. If positions are expressed on a different scale, then the model will adjust by rescaling the rates by the same magnitude (in case the rates are estimated).

SNP with a null position will be discarded. Ideally, this should be done prior to load data in `RZooRoH`.

5.2.2 Fields for genotype / sequence information After the columns for the marker information, come the columns with the genotype / sequence information. One, two or three columns are expected per individual according to the format:

- GT format: 1 column per individual containing the genotypes expressed as allele dosages, 0 for AA, 1 for AB and 2 for BB. Missing genotypes are indicated by a value of 9.
- GL format: three columns per individual containing genotype likelihoods in phred scores for the three possible genotypes REF/REF, REF/ALT, ALT/ALT. This field is available in some VCF files (obtained after variant calling with sequencing data). For missing genotypes, the three values should be 0.

- GP format: three columns per individual containing genotypes probabilities for the three possible genotypes AA, AB and BB. This format corresponds to the Oxford GEN format. For missing genotypes, the three probabilities should be 0.

- AD format: two columns per individual containing the number of reads for the first allele and for the second allele. This field is also available in some VCF files. For missing genotypes, values of 0 are used. This format is not recommended for high sequencing depth. At high sequencing depth, genotypes are called accurately and a GT (or eventually a GP) format is optimal. When AD values are too high, it can result in numerical problems.

- VCF format: this corresponds to a phased VCF file (for example obtained with Beagle). There is one column per individual indicating the two alleles (0 for REF and 1 for ALT) separatd by "|" to indicate phasing. For instance "0|1", "1|1" or "1|0". Missing genotypes are indicated as ".|.". This format is for diploid individuals only.

- HAPS format: this is also a format for phased data with two columns per individual (one per haplotype). Alleles are coded as 0 and 1. This is the only format that can be used for haploid individuals. RZooRoH reads HAPS data more efficiently that the VCF format (BCFtools can be used to convert VCF files in HAPS format - see below).

5.2.3 Examples for the six data formats

Example of a file in **GT format**. There are four columns to describe the markers: chromosome, position (in bp), first allele and second allele. Then we have genotypes for ten individuals (we observe only 0's and 1's, AA and AB). Note that this is not the default format, the second column with marker names is missing. Therefore, additional information will be provided when reading the data.

```
file1 <- system.file("exdata","BBB_PE_gt_subset.txt",package="RZooRoH")
myfile1 <- read.table(file1,header=FALSE)
head(myfile1)
#>     V1    V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14
#> 1 chr1  6665  G  A  0  0  0  0  0   0   0   0   0   0
#> 2 chr1  9149  G  A  0  0  0  1  1   0   0   1   0   1
#> 3 chr1 13812  T  C  1  1  0  1  1   0   0   0   1   1
#> 4 chr1 29575  C  G  0  0  0  0  0   0   0   1   0   0
#> 5 chr1 31490  T  C  0  0  0  0  0   0   0   1   0   0
#> 6 chr1 33632  C  T  0  0  0  0  0   0   0   1   0   0
```

Example of a file in **GL format**. There are five columns to describe the markers: chromosome, marker name, position (in bp), first allele and second allele (default format). Then we have genotype likelihoods in phred scores (three values per individual) for ten individuals. We print the first three individuals. The phred scores can be converted in genotype probabilities (see https://samtools.github.io/hts-specs/VCFv4.1.pdf for more information).

```
file2 <- system.file("exdata","BBB_NMP_pl_subset.txt",package="RZooRoH")
myfile2 <- read.table(file2,header=FALSE)
head(myfile2[,1:14])
#>     V1          V2    V3 V4 V5 V6 V7  V8 V9 V10 V11 V12 V13 V14
#> 1 chr1           .  9149  G  A  0 12 213  0   3  41   0   0   0
#> 2 chr1 rs208929886 35740  T  A 37  3   0  0   0   0   0   0   0
#> 3 chr1 rs208268304 35742  C  G 37  3   0  0   0   0   0   0   0
#> 4 chr1 rs384017208 36011  G  A  0  6  78  0   0   0   0   0   0
#> 5 chr1 rs132895573 36337  G  A  0  6  78 32   0  73   0   0   0
#> 6 chr1 rs208669904 36440  A  G  0  3  40 41   3   0   0   0   0
```

Example of a file in **GP format**. There are also five columns to describe the markers as above. Then we have genotype probabilities for AA, AB and BB genotypes (three values per individual) for ten individuals. We print the first three individuals. This corresponds to the oxford GEN format where the marker information columns are chromosome, marker name, position, first allele and second allele.

```
file3 <- system.file("exdata","BBB_NMP_GP_subset.txt",package="RZooRoH")
myfile3 <- read.table(file3,header=FALSE)
head(myfile3[,1:14])
#>      V1             V2   V3 V4 V5       V6       V7       V8       V9      V10
#> 1 chr10  chr10:126_G_A  126  G  A 0.000000 0.000000 0.000000 0.531308 0.335233
#> 2 chr10  chr10:297_C_T  297  C  T 0.888184 0.111816 0.000000 0.984398 0.015602
#> 3 chr10 chr10:1173_C_T 1173  C  T 0.624537 0.313010 0.062454 0.666125 0.333854
#> 4 chr10 chr10:1184_C_A 1184  C  A 0.799240 0.200760 0.000000 0.000793 0.998891
#> 5 chr10 chr10:1194_C_A 1194  C  A 0.001670 0.333303 0.665027 0.000000 0.200760
#> 6 chr10 chr10:1200_T_G 1200  T  G 0.000021 0.333854 0.666125 0.000000 0.200760
#>        V11 V12 V13 V14
#> 1 0.133459   0   0   0
#> 2 0.000000   0   0   0
#> 3 0.000021   0   0   0
#> 4 0.000316   0   0   0
#> 5 0.799240   0   0   0
#> 6 0.799240   0   0   0
```

For instance, the first individual has missing values for the first marker (the three genotype probabilities are null) and the second individual has genotype probabilities of 0.53, 0.34 and 0.13 for genotypes AA, AB and BB. For the fourth marker, the second individual has a high probability to be heterozygous AB (0.998). The third individual has missing information for the six printed genotypes. Note that these data has been obtained with an average coverage of 1x.

Example of a file in **AD format**. The same five columns are used to describe the markers. Then we have read counts for the two alleles (two values per individual) for ten individuals. We print the first five individuals. For instance, the first individual has four REF and zero ALT alleles for the first markers and one ALT allele for the third marker.

```
file4 <- system.file("exdata","BBB_NMP_ad_subset.txt",package="RZooRoH")
myfile4 <- read.table(file4,header=FALSE)
head(myfile4[,1:15])
#>     V1          V2    V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15
#> 1 chr1           .  9149  G  A  4  0  1  0   0   0   2   1   1   0
#> 2 chr1 rs208929886 35740  T  A  0  0  0  0   0   0   0   1   0   0
#> 3 chr1 rs208268304 35742  C  G  0  1  0  0   0   0   0   1   0   0
#> 4 chr1 rs384017208 36011  G  A  2  0  0  0   0   0   1   0   2   0
#> 5 chr1 rs132895573 36337  G  A  2  0  2  1   0   0   1   0   1   1
#> 6 chr1 rs208669904 36440  A  G  1  0  0  1   0   0   1   0   0   1
```

Example of a file in **VCF format**. Here, we refer only to phased VCF files (the package does not read general VCF files). Nine column provide information on markers. Then we have phasing data for four individuals. The symbol “|” indicates that the data has been phased. Note that the headers from the VCF files are not printed.

```
file5 <- system.file("exdata","TAF_phased_ex.vcf",package="RZooRoH")
myfile5 <- read.table(file5,header=FALSE)
head(myfile5)
#>   V1     V2                              V3 V4 V5 V6   V7 V8 V9 V10 V11 V12 V13
#> 1  1 135098          Hapmap43437-BTA-101873  G  A  . PASS  . GT 1|1 1|0 0|0 0|0
#> 2  1 267940              ARS-BFGL-NGS-16466  A  G  . PASS  . GT 0|0 0|0 0|0 0|0
#> 3  1 393248 Hapmap34944-BES1_Contig627_1906  A  G  . PASS  . GT 1|1 1|0 0|0 0|0
#> 4  1 471078              ARS-BFGL-NGS-98142  G  A  . PASS  . GT 0|0 0|1 0|0 1|1
#> 5  1 516404          Hapmap53946-rs29015852  G  A  . PASS  . GT 0|0 0|0 1|1 0|0
#> 6  1 533815             ARS-BFGL-NGS-114208  G  A  . PASS  . GT 0|0 0|0 1|1 0|0
```

Example of a file in **HAPS format**. There are five columns to describe the markers: chromosome, marker name, position (in bp), first allele and second allele (default format). Then we have 22 haplotypes with REF and ALT alleles coded as 0 and 1, respectively (“.” for missing). In the present example, there a 22 haploid individuals. In case of diploid individuals, we would have two haplotypes per individual (two columns per individual). We print the haplotypes from seven haploid individuals.

```
file6 <- system.file("exdata","Ref22_EX.haps",package="RZooRoH")
myfile6 <- read.table(file6,header=FALSE)
head(myfile6[,1:18])
#>              V1 V2  V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18
#> 1 chromosome_01  . 506  T  C  0  0  0  0   0   0   0   0   0   0   0   0   0
#> 2 chromosome_01  . 516  T  C  0  0  0  0   0   0   0   0   1   1   0   0   0
#> 3 chromosome_01  . 552  A  G  0  0  0  0   0   0   0   0   1   1   0   1   0
#> 4 chromosome_01  . 587  C  T  0  0  0  0   0   0   0   0   0   1   0   0   0
#> 5 chromosome_01  . 610  A  G  0  0  0  0   1   0   0   0   0   0   0   0   0
#> 6 chromosome_01  . 629  A  G  0  0  0  0   0   0   1   0   0   0   0   0   0
```

5.2.4 Converting from ped or VCF files If your file is in ped or VCF format, you can convert them to the Oxford GEN format with PLINK (https://www.cog-genomics.org/plink/1.9/) or BCFtools (http://samtools.github.io/bcftools/bcftools.html#convert). `RZooRoH` is then able to read the Oxford GEN format with the GP format.

For ped files, 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 (bash command):

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

Sometimes, the tools set missing genotype probabilities to 0.333, 0.333 and 0.333. Previously this was not considered missing. This had no consequence on emission probabilities but could slightly affect estimation of allele frequencies. Now, when the three genotype probabilities are higher than 0.33, we consider the genotype missing. Nevertheless, it is recommended to set missing values to 0 0 0.

BCFtools can also be used to extract the desired fields from a VCF file of an **haploid organism**, providing then an “HAPS” zformat. For example:

```
bcftools query -f ’%CHROM %ID %POS %REF %ALT[ %GT]\n’ Ref22.vcf.gz > Ref22.haps
```

5.3 Reading the data

An analysis with `RZooRoH` requires three mains steps: reading the data, defining the model and running the model. Here we describe the first step but first the package must be loaded:

```
library(RZooRoH)
```

The input data must be loaded with the `zoodata` function that creates a `zooin` object containing all the information for further analysis. With the `zoodata` function the user can specify the name of the data file, the genotype / sequence data format, the format of the marker information, minor allele frequencies (**MAF**) filtering rules and eventually the name of files with individuals IDs or with allele frequencies estimated previously (e.g., with a larger data set).

5.3.1 Specifying the data file and genotype / sequence format

The name of the data file is specified with the `genofile=filename` option and the genotype format with the `zformat` option. Six values are possible as described in section 5.2.2: “gt”, “gp”, “pl”, “ad”, “vcf” and “haps” (the last two formats are for phased data only). The “gt” format is used when the `zformat` is not specified.

For instance, to read the file in the GP format:

```
file3 <- system.file("exdata","BBB_NMP_GP_subset.txt",package="RZooRoH")
BBB_GP <- zoodata(genofile = file3, zformat = "gp")
#> [1] "Number of positions in original file ::"
#> [2] "1000"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "1000"
```

The function tells that it has red 1000 lines (markers) and none were filtered out. Note that specifying `genofile =` is not required and the same result is obtained with this shorter command:

```
BBB_GP <- zoodata(file3, zformat = "gp")
#> [1] "Number of positions in original file ::"
#> [2] "1000"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "1000"
```

To read a file in the AD format, the command would be:

```
file2 <- system.file("exdata","BBB_NMP_ad_subset.txt",package="RZooRoH")
BBB_AD <- zoodata(file2, zformat = "ad")
#> [1] "Number of positions in original file ::"
#> [2] "1000"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "1000"
```

5.3.2 Specifying the format for marker information As indicated in section 5.2.1, the `zoodata` function requires marker information that comes in the first columns of the file. Two fields are required, a column with the chromosome information and a column with the position. By default, `zoodata` assumes five columns with marker information and that the chromosome is indicated in the first position whereas the position is in the third column. In the two above examples, the marker information had the default format and there was no need to use options associated with format of marker information fields. With a phased "vcf" format, the function assumes nine columns with marker information (with chromosome in the first column and the marker position in the second column).

When a different format is used, it can be specified with the `supcol`, `poscol` and `chrcol` options. The `supcol` option indicates the number of columns present before the first genotype / sequence data, the `poscol` option indicates the column with the genetic position information and the `chrcol` indicates the column with the chromosome information.

For instance, the "BBB_PE_GT_subset.txt" file contains genotypes in the GT format, four columns for marker information with the chromosome and genetic positions indicated in first and second columns respectively. To read the data we run the following command (note that the "gt" format does not need to be specified since it is the default format):

```
file1 <- system.file("exdata","BBB_PE_gt_subset.txt",package="RZooRoH")
BBB_GT <- zoodata(file1, supcol = 4, poscol = 2, chrcol = 1)
#> [1] "Number of positions in original file ::"
#> [2] "1000"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "1000"
```

5.3.3 Minimal allele frequency threshold With the `min_maf` option, the user can specify a threshold for MAF filtering. For instance, if we want to keep only marker with a MAF higher than 0.05:

```
BBB_GT <- zoodata(file1, supcol = 4, poscol = 2, chrcol = 1, min_maf = 0.05)
#> [1] "Number of positions in original file ::"
#> [2] "1000"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "929"
```

Now, 929 markers remain after filtering. Note that the allele frequencies were estimated only with ten individuals. Some comments on filtering are providing in section 5.4.

5.3.4 Estimation of alleles frequencies - the EM algorithm

By default, RZooRoH use the original functions to estimate allele frequencies. These were relatively simple functions. In a later version, a better EM algorithm was added and can be called with the option `freqem=TRUE"`. For example:

```
BBB_AD2 <- zoodata(file2, zformat = "ad", freqem = TRUE)
#> [1] "Number of positions in original file ::"
#> [2] "1000"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "1000"
```

The approach is recommended for low-fold sequencing for example, and more so with the PL or GP format (the original approximation with the AD format being closer to the EM). The approach is ignored with the GT format. For high confidence genotypes (e.g., genotyping arrays, high-coverage sequencing data), it is not necessary to use this EM approach as genotypes are known.

Note that with a low number of individuals, the allele frequencies can not be estimated correctly and this impacts the accuracy of the method. In particular, with a single individuals, all homozygous positions would be uninformative (as they will be considered monomorphic).

5.3.5 Additional external files (sample names and allele frequencies)

The `zoodata` function can extract information from two additional files with the `samplefile` and the `allelefreq` options.

The `samplefile` option is used to indicate a file containing the names of the sample in the genotype / sequence data file. These names are not required and will be added to the results (for instance for plotting). The file should contain only one column with the names of the samples.

For instance, to add the name of the samples to the last data set:

```
mysamples <- system.file("exdata","BBB_samples.txt",package="RZooRoH")
BBB_GT <- zoodata(file1, supcol = 4, poscol = 2, chrcol = 1, samplefile = mysamples)
#> [1] "Number of positions in original file ::"
#> [2] "1000"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "1000"
```

The `allelefreq` option is used to indicate a file containing the allele frequencies of the first allele. The file should contain only one column with these allele frequencies. An example of the use of this option is given later (in section 7.2.3). The use of external frequencies can be useful when these were obtained from a larger and more informative data set. For instance, when you want to run `RZooRoH` for a few individuals but estimates of allele frequencies are available from a large sample. Similarly, if you have only a few individuals sequenced but you have also frequencies obtained from a pool of individuals. The option allows also to use distinct frequencies such as estimates of ancestral allele frequencies. In addition, it avoids to re-estimate the frequencies for every run. Finally, skipping estimation of allele frequencies can save memory.

Finally, this option is also useful when splitting the data set in order to run several RZooRoH procedures in parallel. Storing the allele frequencies allows then to extract only the genotypes from the target individuals. RZooRoH will then use much less memory because there will be no need to load the full genotype file. This can be valuable when working with whole-genome sequence data.

5.3.6 The haploid option This optional argument indicates whether we work with haploid data and requires the use of the "haps" zformat. Such data can only be processed later with the `ibd` option in the `zoorun` function. With the `haploid` option, an uneven number of haploids can be provided in the "haps" file.

```
myfile6 <- system.file("exdata","Ref22_EX.haps",package="RZooRoH")
data6 <- zoodata(myfile6, zformat = "haps", haploid = TRUE)
#> [1] "Number of positions in original file ::"
#> [2] "35"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "35"
```

5.3.7 Structure of the zooin object The `zooin` object is intended for internal use. It contains the nine slots necessary for further analysis: genos, bp, chrbound, nind, nsnps, nchr, zformat, sample_ids. These can be accessed by using the "@" symbol. For instance, to obtain the genotype table from the `zooin` object called BBB_GT you need to type BBB_GT@genos.

The zooin@genos is a matrix containing genotypes, genotype probabilities, read counts (the format is stored in the zooin@zformat variable). For the previous examples:

```
head(BBB_GT@genos)
#>      V5 V6 V7 V8 V9 V10 V11 V12 V13 V14
#> [1,]  0  0  0  0  0   0   0   0   0   0
#> [2,]  0  0  0  1  1   0   0   1   0   1
#> [3,]  1  1  0  1  1   0   0   0   1   1
#> [4,]  0  0  0  0  0   0   0   1   0   0
#> [5,]  0  0  0  0  0   0   0   1   0   0
#> [6,]  0  0  0  0  0   0   0   1   0   0

head(BBB_GP@genos[,1:6])
#>            V6       V7       V8       V9      V10      V11
#> [1,] 0.000000 0.000000 0.000000 0.531308 0.335233 0.133459
#> [2,] 0.888184 0.111816 0.000000 0.984398 0.015602 0.000000
#> [3,] 0.624537 0.313010 0.062454 0.666125 0.333854 0.000021
#> [4,] 0.799240 0.200760 0.000000 0.000793 0.998891 0.000316
#> [5,] 0.001670 0.333303 0.665027 0.000000 0.200760 0.799240
#> [6,] 0.000021 0.333854 0.666125 0.000000 0.200760 0.799240
```

The zooin@nind, zooin@nsnps and zooin@nchr represent the number individuals, the number of SNPs (after filtering) and the number of chromosomes. In the previous example, the data contains 10 individuals, 1000 SNPs and only one chromosome:

```
c(BBB_GT@nind, BBB_GT@nsnps, BBB_GT@nchr)
#> [1]   10 1000    1
```

The zooin@chrbound is a matrix with one row per chromosome and two values per row: the number of the first marker in that chromosome and the number of the last marker in that chromosome. The chromosome identifier (name or number) is stored in zooin@chrnames. The zooin@bp is an array with the marker genetic positions. The zooin@freqs contains the estimated or loaded allele frequencies for the first allele.

```
head(cbind(BBB_GT@bp, BBB_GT@freqs))
#>      [,1] [,2]
#> [1,]  6665 0.00
#> [2,]  9149 0.20
#> [3,] 13812 0.30
#> [4,] 29575 0.05
#> [5,] 31490 0.05
#> [6,] 33632 0.05
```

Finally, the zooin@sample_ids contains the sample names:

```
BBB_GT@sample_ids
#>  [1] "Maxx7"    "Kopi"     "Ever5337" "Skippy"   "Unkown"   "GIGA"
#>  [7] "Arlequin" "Black"    "Kenza"    "Minus"
```

5.4. Comment on data filtering

One important property of models using the allele frequencies (likelihood-based ROH or HMM) is that they account for marker allele frequencies and are therefore less sensitive to filtering rules. For rule-based ROH it would be important to remove monomorphic markers (unless they were segregating in the base population) and eventually markers with low MAF. In the HMM, monomorphic markers are automatically ignored (because the emission probabilities are identical for HBD and non-HBD states) and when homozygous genotypes are observed, the support for HBD is weighted by the allele frequency. For instance, the probability to observe an homozygous AA is ${f_A}^2$ in non-HBD segments and approximately f_A for HBD segments. So, the emission probability in non-HBD state is approximately equal to the emission probability in HBD state multiplied by f_A. If f_A is high, the probabilities are close with little support in favor of HBD whereas for low f_A, the support for HBD is higher.

Overall, the HMM approach is not too sensitive to filtering on MAF. We compared the filtering rules on the real data sets used in Druet and Gautier (2017) and found little differences in setting `min_maf` to 0, 0.01 or 0.05. Under HWE and in absence of genotyping errors, homozygosity at rare alleles should strongly support HBD. When the frequency gets very low (e.g. 1e-3), the supports becomes stronger. However, it is also possible that the population is not completely under HWE or that there are genotyping errors. There is also more chance that an homozygous rare allele is a genotyping error (because the chance to observe such genotypes in absence of errors is low). This could occur in presence of deletions (the individual is not homozygous for the rare allele but hemizygous). Therefore, it is not clear whether the rare alleles must be kept with the emission probabilities we use. As the model relies on HBD segments (series of homozygous markers), if this homozygous rare allele is not in a long stretch of homozygous markers, it will contribute to the most ancient HBD class (the shortest segments). Conversely, if that marker is in a long stretch of homozygous markers, the segment would be called even when that marker is filtered out. So, in both cases it does not impact too much HBD estimation in the more recent HBD classes (those of higher interest).

Similarly, our multiple HBD class model is quite robust to LD structure (or absence of LD filtering). With one HBD class HMM, it is sometimes necessary to perform LD pruning to capture only the long HBD fragments and not the small HBD fragments associated to background LD. For instance, Leutenegger et al. (2011) selected subsets of 1% of their data. With our model, the long HBD segments and the small HBD segments go into distinct classes. Applying our model without data filtering, we obtained estimates similar to Leutenegger et al. (2011) in the recent HBD classes whereas additional autozygosity was associated to more ancient HBD classes. See also section 3.2 (and Figure 2) and Druet and Gautier (2017) for more details.

Our models achieved also good performances without LD pruning on data simulated with population genetics models, see Druet and Gautier (2017) for more details.

6 Defining the model

With RZooRoH you can define different models. The most important parameters are the number of classes K and the rates R_k for each class. These rates can either be fixed by the user ("pre-defined") or estimated by RZooRoH for every individual. The optimal model to select depends on the objectives of the analysis and on the data set.

6.1 Models with one HBD class and one non-HBD class: 1L model

The one HBD class HMM is a special case of the models for which the rates R_k are estimated. In that case, the same rate is used for HBD and non-HBD classes as in Leutenegger et al. (2003), Narasimhan et al. (2016) or Vieira et al. (2016). We recommend to use such a model only when all the autozygosity or HBD tracks trace back to one ancestor or several ancestors living in the same generation. That model might also require some LD pruning prior to analysis.

6.2 Models without pre-defined rates: KL models (RZooRoH will estimate the rates R_k)

If the goal is to identify all HBD segments that can be captured with the marker density and to estimate the total autozygosity, then a model with a few HBD classes would be indicated. In that case, the optimal number of classes is a function of the marker density. For instance, Solé et al. (2017) showed with cattle data that for low-density arrays only one or two HBD classes are needed whereas for sequencing data, three or four HBD classes were recommended. The BIC can be used to compare models with different number of classes K and to determine the optimal K for each individual (Druet and Gautier, 2017).

This strategy with an optimized number of classes would also be recommended to perform a length-based classification of HBD segments in main categories (e.g. long segments associated with recent relatedness, intermediate segments and short segments associated with LD patterns) as in Pemberton et al. (2012). The main categories make the LD pruning unnecessary.

Optimizing K is also more efficient computationally although convergence is more difficult with R_k rates estimation. The use of the BIC criteria to select K combined with the estimate of the R_k also reduces the risk to fit an inappropriate model and miss some autozygosity (because classes are too distant or do not cover the whole range of segments that can be captured with the current marker density).

However, this strategy is not recommended if the user wants to determine which generations of ancestors contribute to present autozygosity (e.g. to identify past bottlenecks, identify offspring from closely related individuals). In addition, such a strategy with estimation of R_k rates makes comparisons across individuals uneasy because different individuals will have different R_k. Their autozygosity will therefore be partitioned in different HBD classes and eventually estimated with respect to different base populations.

Overall, we don't recommend to use the KL models unless the user has very specific objectives. We recommend to use MixKL models with pre-defined rates (the user can select them). Parameters are easier to estimate and the model is more robust.

6.3 Models with pre-defined rates: `MixKL` models

In a so-called `MixKL` model, the R_k rates are pre-defined by the user and no longer estimated. Then, `RZooRoH` will only estimate the mixing coefficients that will influence the number of segments in each class.

6.3.1 Selecting the number of classes and their rates The user should select a set of classes that cover the whole range of HBD segments that can be captured with the genotyping array:

- The smallest rate (most recent class) should not be smaller than 1. A value of two corresponds approximately to a common ancestor present one generation ago as with selfing. So, values lower than two should ideally not be used.
- The highest rate depends on the marker density. For instance, there is not enough data to capture HBD segments smaller than the average marker spacing. With 10 SNPs per cM, the average spacing would be 0.1 cM corresponding to the average segment length with a R_k rate of 1,000. Therefore, there is no need to put classes with rates higher than 1,000. Previous analyses with simulated and real data sets (Druet and Gautier, 2017; Solé et al., 2017) suggest that setting a few classes with too high R_k doesn't affect the result because these classes are not used (they stay empty) although computational costs are increased. On the other hand, setting to few classes (to small R_k for the last class) is not optimal because the last class will capture its HBD segments and those from the unmodeled older classes with a higher rate.
- Finally, the number of classes K should allow to cover the range from the first to the last rate. With too many classes, the computational burden will dramatically increase and there is probably not enough information to distinguish segments for classes with very similar rates. We like to use series of rates with constant ratio between successive rates (expected lengths are divided by a constant value from one class to the next). We recommended to use a ratio of two or higher to limit the overlap between exponential distributions. For instance the power of two [2, 4, 8, 16, 32, 64, . . . , 512, 1024, 2048, 4096] allows to have more classes for long HBD segments that for shorter segments. There is more information to distinguish two exponential distributions with rates 2 and 4 (expected length 25 and 50 cM) than with rates 998 and 1000 (expected length 0.1002 and 0.1000 cM).
- Other series such as the power of five [5, 25, 125, 625, 3125, . . .] or ten [10, 100, 1000, 10000] can cover the same range with less classes (reducing the computational cost). However, some resolution is lost because classes regroup more generations of ancestors. For instance, rates of 4, 8, 16 would result in categories for grand-parents, ancestors present 4 or 8 generations ago whereas rates of 5, 25 and 125 would result in categories for ancestors around 2.5 generations ago, 12.5 generations ago and more than 60 generations. With [10, 100, 1000, . . .], the resolution would be even poorer with one group for very recent ancestors (centered around 5 generations ago) and one group for more ancients ancestors (centered around 50 generations ago).

The rules and models proposed above should limit the risk to miss autozygosity because the first rate is to high, the last rate is too low or two rates are too distant from each other.

A more complex strategy to select the pre-defined HBD classes would be to first run `KL` models and select the optimal number of classes K with the BIC. Then, we could select as rates R_k for the `MixKL` model the median values of R_k estimated with the optimal `KL` model (the median of R_1's for R_1, the median of R_2's for R_2, etc.). With that strategy, we would obtain a parsimonious `MixKL` model but the model selection step would be time consuming. It would also limit the risk to miss some autozygosity because the model is inappropriate.

6.3.2 Benefits of using pre-defined rates Compared to `KL` models (with estimation of the rates), `MixKL` models:

- Allow an easier comparison across individuals because they all have the same R_k and the same HBD classes. It seems to be better to use the same set of classes for all individuals.
- Offer more resolution to determine which ancestors contribute to the autozygosity of individuals because most often models with pre-defined rates have more classes. With more classes, we can better identify offspring from closely related parents (with a common ancestor in very recent past) and obtain more precise estimates of demographic events such as bottleneck or founder effects.
- Allow to estimate inbreeding coefficients with respect to the same base population for all individuals (since the HBD classes are the same).
- Fit as well the data as models selected with BIC (Druet and Gautier, 2017).
- Converge often better.
- Have higher computational costs (large models than run longer and require more memory). However, they are not systematically slower because parameters are estimated in fewer iterations (only the mixing coefficients need to be estimated).

We recommend the use of pre-defined models for these reasons and set "pre-defined" as default value for the `zoomodel` function.

6.4 Using `zoomodel` to define your model

6.4.1 Options The `zoomodel` function creates a `zmodel` object needed to run our model. The function allows to define whether the R_k rates are pre-defined (default) or not with the `predefined` option, the number of layers K (option $K = \ldots$). The values of the R_k can either be defined with the `base_rate` option (default = 2). Then, the successive rates will be equal to b^k where b is the selected base rate and k the class number. Alternatively, the `krates` option can be used to specify the K rates. The `mix_coef` determines the starting values for the mixing coefficients. In addition to these parameters, the user can provide the genotyping error rate ϵ with option `err` (the probability to observe a heterozygous genotype in an HBD class) or sequencing error rates (the probability to have a sequencing error in one read, for the "ad" format) with option `seqerr`.

6.4.2 More advanced options: stepfunctions and layers defined as intervals

6.4.2.1 Step functions If the user fits a large number of layers, it is possible to use a step function to estimate the mixing coefficients: the mixing coefficients of neighboring layers are equal (the mixing coefficient are equal for groups or blocks of consecutive layers). For instance, if 50 layers are defined, this function could allow to force the mixing coefficients of layers 1 to 10, 11 to 20, 21 to 30, 31 to 40 and 41 to 50 to be identical. This would require the estimation of five different mixing coefficients instead of 50 and will result in improved convergence (there is probably not enough information to estimate 50 coefficients, especially if the rates from the different layers are close to each other). To use this option, `step` must be set to 'TRUE' and an incidence matrix must be specified using "XM = M" (where M is the name of your incidence matrix). The incidence matrix connects all K defined layers (rows) to the reduced set of mixing coefficients (columns). All elements of the matrix should be zero, with 1's added at position (i,j) to indicate that layer i (row) uses mixing coefficient j (column). There should be one "1" in each row (no more, no less).

For example, let's define a model with ten layers with rates equal to {2,4,8,10,12,14,16,18,20} and decide that the mixing coefficients are constant for the first two layers, for layers 3 to 5 and for the last five layers. The incidence matrix would then be:

```
MyMat = matrix(0,10,3)
MyMat[(1:2),1]=1;MyMat[(3:5),2]=1;MyMat[(6:10),3]=1
MyMat
#>       [,1] [,2] [,3]
#>  [1,]    1    0    0
#>  [2,]    1    0    0
#>  [3,]    0    1    0
#>  [4,]    0    1    0
#>  [5,]    0    1    0
#>  [6,]    0    0    1
#>  [7,]    0    0    1
#>  [8,]    0    0    1
#>  [9,]    0    0    1
#> [10,]    0    0    1
```

This matrix can then be used with the `zmodel` function, the number of layers must be set to 10, we need to specify 10 rates but give only starting values for each of the three mixing coefficients. Importantly, the `step` option must be TRUE and the incidence matrix must be provided as follows:

```
StMix10L <- zoomodel(K=10,krates=c(2,4,6,8,10,12,14,16,18,20),mix_coef = c(0.01,0.01,0.01),
                     step = TRUE, XM = MyMat)
```

We can verify that the zmodel@typeModel is “step_mixkl”.

```
StMix10L@typeModel
#> [1] "step_mixkl"
```

This `step` option would be useful when fitting a large number of layers. For example, with the improved computational efficiency of the model, it is possible to run models with 50 or 100 layers. In this case, we could fit layers that would roughly correspond to generations. For example, for generations 1 to 100, the rates should be from 2 to 200 (with a step of 2). Of course, this would be an approximation and there is no guarantee that a given layer would actually capture a given generation. However, it makes interpretation easier in terms of number of generations to the common ancestor and within a layer.

Estimating 100 mixing coefficients would be difficult, especially when the rates are very close to each other. In addition, mixing coefficients reflect inbreeding rates that are a function of effective population size, which should not change abruptly in each past generation. Therefore, using a step function in which layers are grouped by 5 or 10 would be useful (the inbreeding rates remain the same for several generations).

This option must be used with caution, as the computational requirements are much higher, while the usefulness of the estimated parameters needs to be further evaluated.

With this option, the mixing coefficients are related to the inbreeding rates per generation and are thus expected to be smaller than for standard `MixKL` models where the layers include multiple generations (and the mixing coefficient is defined for the group of generations).

This option is only possible with pre-defined rates (not with `KL` models).

6.4.2.2 Layers defined as intervals The option `HBDclass = "Interval"` also allows to model many layers (one per generation of ancestors), but with a different strategy. Instead of fitting each generation as a single layer, the `"Interval"` option models layers as all discrete generations in intervals. Layers are then defined to include all discrete generations from G1 to G2 (the boundaries). For each generation we use a rate of twice the generation. We remind that the relationship between generations and rates is not an exact fit and that this is an approximation. However, this approach make the results easier to interpret (in terms of age or number of generations in an HBD class).

In some aspects, our model is similar to PSMC models (which also use discrete classes). In fact, to compute the transition probabilities between our newly defined states, we can apply the rules developed by Harris et al. (2014) (described in their first appendix). They use continuous time, while we use discrete generations (as we work in more recent times) and work with sums instead of integrals. In fact, this option integrates different probabilities (probabilities of being in a certain state, transition probabilities, recombination probability in a layer, etc.) in the interval.

Compared to the `step` option, the `"interval"` approach has the advantage of defining fewer states (classes). This has some computational advantages. However, the computation of probabilities and transitions by summing over all generations in an interval is also time consuming. With the `step` approach, we still need to fit one class per "generation" and this limits the number of generations that can be fitted (the absence of older generations may impact the estimation in more recent layers), while with the `interval` approach, we can include more ancient generations. By fitting the layers, the `step` option is more accurate (no approximation in estimation probabilities of being in a certain state, transition probabilities, recombination probability in a layer, etc.) while some minor approximations are used in the `interval` approach. Both approaches gave very similar results when we compared them on different real and simulated data sets.

The `Interval` approach can also be slower (e.g. compared to the standard `MixKL` model), especially when many generations are fitted. The `zoorun` function has an option called `RecTable`, which allows faster computation.

When `Interval` is TRUE, the values specified with the `krates` parameters correspond to the last generations of the intervals. The first interval starts at generation 1 and subsequent intervals start after the last generation of the previous interval. The rates used by the model are twice the values of each generation.

For example, to define a model with HBD classes defined as intervals from generations 1 to 5, 6 to 10, 11 to 20 and 21 to 50:

```
Mix4I <- zoomodel(K=4,HBDclass ="Interval",krates=c(5,10,20,50),mix_coef=rep(0.001,4))
```

As with the `step` option, mixing coefficients are related to the inbreeding rates per generation and are therefore expected to be smaller than for standard `MixKL` models where layers include multiple generations.

This `HBDclass` is only possible with pre-defined rates (not with `KL` models) and is not compatible with the `step` option.

6.4.3 Output: the `zmodel` object The returned `zmodel` object contains seven slots: the zmodel@typeModel can be "mixkl" (for predefined = TRUE), "kl" (for predefined = FALSE) or "step_mixkl", the zmodel@mix_coef contains the starting values for the mixing coefficients of each class, the zmodel@krates contains the (starting) values for the R_k rates, the zmodel@err contains the genotyping error rate, the zmodel@seqerr contains the sequencing error rate, zoomodel@XM is the incidence matrix (when the `step` option is used) and zoomodel@typeClass indicates whether standard layers are defined or 'intervals' are used.

6.4.4 Some examples of model definition with `zoomodel` To define a default model, with 10 layers classes (10 HBD and 1 non-HBD class) with a ratio between successive rates of two:

```
mix10L <- zoomodel()
mix10L
#> An object of class "zmodel"
#> Slot "typeModel":
```

```
#> [1] "mixkl"
#>
#> Slot "mix_coef":
#>  [1] 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
#>
#> Slot "krates":
#>  [1]    2    4    8   16   32   64  128  256  512 1024
#>
#> Slot "err":
#> [1] 0.001
#>
#> Slot "seqerr":
#> [1] 0.001
#>
#> Slot "XM":
#>       [,1]
#>  [1,]    1
#>  [2,]    1
#>  [3,]    1
#>  [4,]    1
#>  [5,]    1
#>  [6,]    1
#>  [7,]    1
#>  [8,]    1
#>  [9,]    1
#> [10,]    1
#>
#> Slot "typeClass":
#> [1] "SingleRate"
```

To access a specific slot use the @ symbol:

```
mix10L@krates
#>  [1]    2    4    8   16   32   64  128  256  512 1024
```

To define a model with pre-defined rates for 4 layers (4 HBD and 1 non-HBD class) with a ratio between successive rates of ten:

```
mix4L <- zoomodel(K=4,base=10)
mix4L
#> An object of class "zmodel"
#> Slot "typeModel":
#> [1] "mixkl"
#>
#> Slot "mix_coef":
#> [1] 0.01 0.01 0.01 0.01
#>
#> Slot "krates":
#> [1]    10   100  1000 10000
#>
#> Slot "err":
```

```
#> [1] 0.001
#> 
#> Slot "seqerr":
#> [1] 0.001
#> 
#> Slot "XM":
#>      [,1]
#> [1,]    1
#> [2,]    1
#> [3,]    1
#> [4,]    1
#> 
#> Slot "typeClass":
#> [1] "SingleRate"
```

To define the same model but setting genotyping error rates to 0.01 and sequencing error rates to 0.005:

```
mix5R <- zoomodel(K=5,base=10,err=0.01,seqerr=0.005)
mix5R@err
#> [1] 0.01
mix5R@seqerr
#> [1] 0.005
```

To define a model with three layers, with estimation of the rates and specifying the initial rates to 16, 64 and 256:

```
my.mod3L <- zoomodel(predefined=FALSE,K=3,krates=c(16,64,256))
my.mod3L
#> An object of class "zmodel"
#> Slot "typeModel":
#> [1] "kl"
#> 
#> Slot "mix_coef":
#> [1] 0.01 0.01 0.01
#> 
#> Slot "krates":
#> [1]  16  64 256
#> 
#> Slot "err":
#> [1] 0.001
#> 
#> Slot "seqerr":
#> [1] 0.001
#> 
#> Slot "XM":
#>      [,1]
#> [1,]    1
#> [2,]    1
#> [3,]    1
#> 
```

```
#> Slot "typeClass":
#> [1] "SingleRate"
```

To change the starting values of the mixing coefficients:

```
my.mod3L <- zoomodel(predefined=FALSE,K=3,krates=c(16,64,256),
                     mix_coef=c(0.03,0.04,0.13))
my.mod3L@mix_coef
#> [1] 0.03 0.04 0.13
```

Finally, to define a model with one HBD-class and common rate for the HBD and non-HBD classes (similar to the original model from Leutenegger et al. (2003)):

```
my.mod1L <- zoomodel(predefined=FALSE,K=1,krates=c(10))
my.mod1L
#> An object of class "zmodel"
#> Slot "typeModel":
#> [1] "kl"
#>
#> Slot "mix_coef":
#> [1] 0.01
#>
#> Slot "krates":
#> [1] 10
#>
#> Slot "err":
#> [1] 0.001
#>
#> Slot "seqerr":
#> [1] 0.001
#>
#> Slot "XM":
#>      [,1]
#> [1,]    1
#>
#> Slot "typeClass":
#> [1] "SingleRate"
```

7 Running zoorun

The zoorun function allows to estimate the parameters of the model, to estimate the global and locus specific realized autozygosity, to partition it in the different HBD classes and to identify the HBD segments. With phased data, it can also be used to compute identity-by-descent (**IBD**) probabilities or identify IBD segments between pairs of haplotypes.

7.1 General options

The zoorun requires two essential elements, the zmodel and zdata objects that provide respectively the model and the data. These objects are obtained by running prior to the analysis the zoomodel and zoodata functions as described in sections 5 and 6. By default, the analysis will be performed on all individuals but the user can provide a list of individuals with the optional `ids` option (see examples below). The `ids` are the position of the individuals in the data file (columns, 1 for the first individual, 2 for the second, etc).

In addition, the analysis can be run in parallel by specifying the number of threads with the `nT` option (equal to 1 by default).

7.2 Parameter estimation

7.2.1 Methods for parameter estimation Parameter estimation is required prior to estimating the realized autozygosity or to identifying the HBD segments. Therefore, parameter estimation is set to true by default (parameters = TRUE). When the parameters have been estimated in a previous run, their value can be indicated in the `zmodel` and the function can be run with parameters = FALSE. This is one of the rare situations where the parameter estimation can be skipped. This approach can also be used if the user wishes to use a set of pre-defined fixed parameters (e.g., to speed up computations).

Originally, the parameter estimation, for the mixing coefficients M_k and eventually the rates R_k, was performed with the EM algorithm as described in Druet and Gautier (2017). In `RZooRoH`, the estimation of parameters is performed with optimization procedures implemented in the `optim` R package, with the L-BFGS-B method (by default). This procedure can converge is less iterations (with models with few parameters) and at a lower computational cost (one iteration to obtain the likelihood required by `optim` needs to compute only the forward variables whereas the EM algorithm requires both forward and backward variables).

In order to work with unconstrained parameters and to obtain ordered HBD classes (with increasing rates of exponential distributions), we defined new parameters (Zucchini and MacDonald, 2009). The parameters are for example the rate differences (between successive classes) and not the rates :

$$\eta_k = \begin{cases} log(R_k - R_{k-1}) & \text{if 1 < k} \leq \text{K} \\ log(R_k - 1) & \text{if k = 1} \end{cases} \tag{1}$$

$$\tau_k = log(\frac{M_k}{1 - M_k}) \qquad \text{if k} \leq \text{K} \tag{2}$$

Using the back-transformation, it can be verified that the rates R_k from HBD classes are always positive and ordered, and that the mixing coefficient M_k are constrained between 0 and 1.

For the first class, the rate is $1 + exp(\eta_1)$ forcing the minimum rate to be 1 (and not 0).

7.2.2 Options for parameter estimation Several optimization methods are available in the `optim` R package. These can be selected using the `optim_method` option. The possible methods are “Nelder-Mead”, “BFGS”, “CG”, “L-BFGS-B”, “SANN” and “Brent”. Type “? optim” for more information. In our experience, the “L-BFGS-B” method works well, but the method that achieves the best likelihood is variable (depending on data sets, models, priors, constraints, etc.).

With the “L-BFGS-B” method, it is possible to impose some constraints on the parameters (this is not possible with the other methods). For the R_k rate parameters, the optional `maxr` represents the maximum differences between the rates of two consecutive classes. For the M_k, the `minmix` option can be used to specify the minimum value. By default, it is set to 0 for standard `KL` and `MixKL` models and to 1e-16 with the `step` option or when the `HBDclass` is set to ‘Interval’ (to avoid numerical problems).

It is also possible to specify the maximum number of iterations using the `maxiter` option. Iterations are not defined identically for all 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. Thus, a value of 100 iterations is good for the “L-BFGS-B” method, but larger values are required for some other algorithms.

If optim fails to converge, we recommend trying other methods (using the optim_method option), trying different starting values of the parameters or different constraints.

7.2.3 Examples We start by loading a data set with six individuals from a cattle population genotyped for a low-density array. An additional file with the allele frequencies estimated on a larger sample is also available (this provides an illustration on how to use the allelefreq option).

```
freqfile <- (system.file("exdata","typsfrq.txt",package="RZooRoH"))
typfile <- (system.file("exdata","typs.txt",package="RZooRoH"))
frq <- read.table(freqfile,header=FALSE)
bbb <- zoodata(typfile,supcol=4,chrcol=1,poscol=2,allelefreq=frq$V1)
#> [1] "Number of positions in original file ::"
#> [2] "6370"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "6370"
```

We will then estimate parameters with the my.mod1R model defined in section 6.4.4. We start with default options for the first individual.

```
bbb_results <- zoorun(my.mod1L, bbb, ids = c(1))
#> final  value 6127.228984
#> converged
```

We repeat parameter estimation setting the maximum rate to 100 and the minimum mixing coefficient to 1e-8.

```
bbb_results1 <- zoorun(my.mod1L, bbb, ids = c(1,2), maxr = 100, minmix = 1e-8)
#> final  value 6127.228984
#> converged
#> final  value 5767.659061
#> converged
```

We perform parameter estimation with another method (Nelder-Mead) and increase the maximum number of iterations to 1000. Note that constraints are only possible when “optim_method” is “L-BFGS-B”.

```
bbb_results2 <- zoorun(my.mod1L, bbb, ids = c(1,2), optim_method = "Nelder-Mead",
                       maxiter = 1000)
#>   Nelder-Mead direct search function minimizer
#> function value for initial parameters = 6150.251745
#>   Scaled convergence tolerance is 9.16459e-05
#> Stepsize computed as 0.459512
#> BUILD              3 6150.251745 6147.589741
#> EXTENSION          5 6148.024244 6141.714123
#> LO-REDUCTION       7 6147.589741 6141.714123
#> EXTENSION          9 6141.763985 6129.142208
#> LO-REDUCTION      11 6141.714123 6129.142208
#> LO-REDUCTION      13 6130.760022 6128.143499
#> LO-REDUCTION      15 6129.142208 6127.738319
#> HI-REDUCTION      17 6128.143499 6127.613612
#> HI-REDUCTION      19 6127.738319 6127.257206
#> LO-REDUCTION      21 6127.613612 6127.257206
#> HI-REDUCTION      23 6127.391712 6127.257206
#> LO-REDUCTION      25 6127.348366 6127.235084
#> HI-REDUCTION      27 6127.257206 6127.235084
#> HI-REDUCTION      29 6127.248864 6127.230681
#> HI-REDUCTION      31 6127.235084 6127.230681
#> HI-REDUCTION      33 6127.233216 6127.230457
#> HI-REDUCTION      35 6127.230681 6127.229760
#> HI-REDUCTION      37 6127.230457 6127.229058
#> LO-REDUCTION      39 6127.229760 6127.229058
#> HI-REDUCTION      41 6127.229156 6127.229058
#> HI-REDUCTION      43 6127.229133 6127.229022
#> Exiting from Nelder Mead minimizer
#>     45 function evaluations used
#>   Nelder-Mead direct search function minimizer
#> function value for initial parameters = 5771.954769
#>   Scaled convergence tolerance is 8.60088e-05
#> Stepsize computed as 0.459512
#> BUILD              3 5774.072032 5770.258735
#> EXTENSION          5 5771.954769 5769.926988
#> REFLECTION         7 5770.258735 5768.639992
#> HI-REDUCTION       9 5769.926988 5768.639992
#> EXTENSION         11 5769.207484 5767.920196
#> REFLECTION        13 5768.639992 5767.808816
#> HI-REDUCTION      15 5767.920196 5767.808816
#> HI-REDUCTION      17 5767.870572 5767.684962
#> HI-REDUCTION      19 5767.808816 5767.684962
#> HI-REDUCTION      21 5767.699563 5767.679018
#> HI-REDUCTION      23 5767.684962 5767.668976
#> HI-REDUCTION      25 5767.679018 5767.661651
#> HI-REDUCTION      27 5767.668976 5767.661651
#> LO-REDUCTION      29 5767.663073 5767.659384
#> HI-REDUCTION      31 5767.661651 5767.659384
#> HI-REDUCTION      33 5767.659726 5767.659384
#> HI-REDUCTION      35 5767.659417 5767.659210
#> HI-REDUCTION      37 5767.659384 5767.659085
#> HI-REDUCTION      39 5767.659210 5767.659085
#> Exiting from Nelder Mead minimizer
#>     41 function evaluations used
```

7.3 Estimating realized autozygosity (with partitioning in different HBD classes)

To estimate the realized autozygosity in each HBD class (the percentage of the genome from an individual associated with a specific HBD class), one has simply to set the option `fb` (for Forward-Backward algorithm) to true (the default value). These values are indeed estimated with the Forward-Backward algorithm (see section 3.3) that allows to compute at each marker position the probability to belong to each of the defined classes in the model. For instance, the realized autozygosity has been automatically estimated in the two previous examples (in the 7.2.3 section).

The realized autozygosity in each HBD class is equal to the genome-wide average of the probabilities to belong to each HBD class computed at each marker position. These locus specific HBD probabilities can be obtained by setting the `localhbdp` option to TRUE. By default this value is set to FALSE because it generates large outputs (number of SNPs x number of classes x number of individuals). These values would be interesting for specific applications such as autozygosity mapping experiments.

7.4 Identifying HBD segments

To identify HBD segments we rely on the Viterbi algorithm (see section 3.3). The Viterbi algorithm identifies the most likely sequence of HBD and non-HBD classes along the genome. At each marker, the genome is assigned to one class and HBD segments can be identified as stretches of consecutive markers assigned to the same HBD class. To use the Viterbi algorithm, one has simply to set the `vit` option to TRUE (the default value). As before, the HBD segments were automatically identified in the two previous examples (in the 7.2.3 section).

Note that we prefer to work with HBD probabilities and the forward-backward algorithm than with the HBD segments and the Viterbi algorithm. We do not recommend to estimate the realized autozygosity as the sum of the length of HBD segments divided by the length of the genome (these estimators were found to be less accurate than those obtained with the Forward-Backward algorithm). Similarly, for autozygosity mapping experiments, we recommend to work with local HBD probabilities to take uncertainty into account. The HBD segments should be used only in applications requiring HBD segments or for visualization purposes.

7.5 Estimation of identy-by-descent (IBD) between two phased haplotypes

The ZooRoH model is most often applied to homologous chromosome pairs within individuals. This does not require to know phases, as we only need to know whether the two alleles are identical (homozygous) or not (heterozygous). However, the same model could conceptually be applied to any pair of chromosomes, even those belonging to distinct individuals. It would then provide information on IBD and its partitioning in different length based classes. However, this requires phasing information (haplotypes must be known). With statistical approaches, there is no guarantee that the phasing is error-free. We have shown that this approach can work in Forneris et al. (2025).

If phased data are provided (see also `zoodata` and `zoomodel`), `zoorun` can be used to perform IBD analysis with the zoomodel on pairs of haplotypes. To do this, the optional `ibd` option must be set to TRUE and the haplotype pairs to be analyzed must be provided with the `ibdpairs` parameter. The `ibdpairs` must specify a **matrix** containing all the pairs to be analyzed. A haplotype is defined by two values, the individual ID (its position in the input file) and its haplotype number (1 for the first haplotype, 2 for the second haplotype). A pair of haplotypes is thus specified with four values: ID of the first individual, haplotype number of first individual (1 or 2), ID of the second individual and haplotype number of the second individual (1 or 2).

As with the zoodata function, we can inform the function that the analysis is done on haploid data by setting the optional `haploid` option to TRUE. In this case there is only one haplotype per individual (both have the same ID corresponding to their position in the input file). Therefore, the matrix specified with `ibdpairs` must contain only two columns: the number of the first individual/haplotype and the number of the second individual/haplotype.

When using phased diploid data, the analysis will be sensitive to phasing errors and their frequency. So-called switch errors will reduce the length of identified IBD segments and affect the partitioning of IBD into different classes (IBD being associated with more ancient IBD classes).

7.6 More efficient computation when `HBDclass` is set to `Interval`

As mentioned above, the `Interval` approach can be time consuming if intervals contain too many generations. Setting the `RecTable` option to TRUE can reduce these computations by making small approximations of the genetic distances. In fact, computations will only be performed for a subset of genetic distances at regular intervals such as 100, 200, 300, . . . , 900, 1000, 1100, 1200, . . . , 9900, 10000 bp. The spacing increases for higher genetic distances: by 1kb {11, 12, 13, . . . , 99, 100 kb}, then by 10 kb {110, 120, 990, 1000 kb}, then by 100 kb {1, 1.1, 1.2 . . . , 10 Mb}, then by 1 cM {10, 11, 12, . . . , 100 cM}. The true genetic distance is then replaced by the closest value from the table, resulting in an approximation of less than 10% (except for very short distances < 1 kb).

The benefit of this approach increases as the number of unique genetic distances in the data set increases, so the benefit is greater at high marker densities. In fact, this approach requires the calculation of about 500 values. If the genetic map contains 5000 markers, the benefit will be modest, but with sequence data the benefit will be higher.

We recommend using `Interval` with the `RecTable` option, because without this option, computation time can increase dramatically and the approximation has little effect on the estimated parameters (we have compared both options on several real and simulated data sets with different models).

7.7 More examples

To obtain the locus specific HBD probabilities for all individuals with the previous example.

```
bbb_results3 <- zoorun(my.mod1L, bbb, localhbd = TRUE)
#> final  value 6127.228984
#> converged
#> final  value 5767.659061
#> converged
#> final  value 5763.247991
#> converged
#> final  value 5809.962884
#> converged
#> final  value 5758.887693
#> converged
#> final  value 5383.740548
#> converged
```

To run a model with two HBD classes with pre-defined rates equal to 10 and 100 on the same data set.

```
Mod2L <- zoomodel(K=2,base_rate=10)
bbb_mod2l <- zoorun(Mod2L, bbb, localhbd = TRUE)
#> final  value 6127.569545
#> converged
#> final  value 5768.378100
#> converged
#> final  value 5763.347984
#> converged
#> final  value 5811.388094
#> converged
#> final  value 5760.214682
#> converged
#> final  value 5383.757491
#> converged
```

To run an IBD analysis on a group of phased diploid individuals, we can use “file5’ previously defined and containing phased that for four cows.

```
DTAF <- zoodata(file5,zformat="vcf")
#> [1] "Number of positions in original file ::"
#> [2] "91"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "91"
```

Then, we need to specify which pairs of haplotypes must be analyzed, for example four pairs:

- The first haplotype of the third individual with the first haplotype of the fourth individual
- The second haplotype of the third individual with the first haplotype of the fourth individual
- The second haplotype of the first individual with the first haplotype of the second individual
- The second haplotype of the second individual with the first haplotype of the fourth individual

```
TPairs <- as.matrix(cbind(c(3,3,1,2),c(1,2,2,2),c(4,4,2,4),c(1,1,1,1)))
TPairs
#>      [,1] [,2] [,3] [,4]
#> [1,]    3    1    4    1
#> [2,]    3    2    4    1
#> [3,]    1    2    2    1
#> [4,]    2    2    4    1
```

Finally, we run `zoorun` with the `Mod2L` model setting the `ibd` option to TRUE and specifying the matrix.

```
TIBD <- zoorun(Mod2L, DTAF, ibd = TRUE, ibdpairs = TPairs)
#> final  value 22.017591
#> converged
#> final  value 22.017591
#> converged
#> final  value 32.209552
#> converged
#> final  value 22.395340
#> converged
```

If we want to apply an IBD analysis to a set of haploid organisms, we can use the 22 individuals present in file6 (“haps” zformat). To estimate the IBD relationship between pair of individuals (1,21),(3,5),(7,10),(22,15) and (6,16), we need to specify the `haploid` option with both `zoodata` and `zoorun`:

```
D22 <- zoodata(file6,zformat="haps",haploid = TRUE)
#> [1] "Number of positions in original file ::"
#> [2] "35"
#> [1] "Number of positions after MAF filtering  ::"
#> [2] "35"
Dpairs <- as.matrix(cbind(c(1,3,7,22,6),c(21,5,10,15,16)))
DIBD <- zoorun(my.mod1L,D22,ibd=TRUE,ibdpairs=Dpairs,haploid=TRUE)
#> final  value 14.736189
#> converged
#> final  value 8.447185
#> converged
#> final  value 10.340587
#> converged
#> final  value 26.001391
#> converged
#> final  value 7.510843
#> converged
```

8 Description of results

Results are grouped in a zres object with 12 slots. These slots can be accessed with the @ symbol. Some slots describe the samples in the analysis:

- ..@nind is simply the number of analyzed individuals in the run (or pairs of haplotypes with `ibd`);
- ..@ids is an array with the numbers of the analyzed individuals (or pairs of haplotypes with `ibd`);
- ..@sampleids is an array with the name of the samples (if they were provided) or with their number (pairs of haplotypes names are obtained by combining individuals names and haplotype numbers (1 or 2));
- ..@optimerr is an array indicating whether `optim` ran without error (for each individual / pair of haplotypes). 0 indicates successful completion, 1 indicates that the iteration limit has been reached, 51 and 52 indicate warnings from the "L-BFGS-B" method, 99 indicates numerical problem. See the `optim` R function for more details.

For instance, for the analysis with two individuals:

```
bbb_results2@nind
#> [1] 2
bbb_results2@ids
#> [1] 1 2
```

And for the analysis with IBD between haplotypes from diploid individuals:

```
TIBD@nind
#> [1] 4
TIBD@ids
#> [1] 1 2 3 4
TIBD@sampleids
#> [1] "3_1_4_1" "3_2_4_1" "1_2_2_1" "2_2_4_1"
```

8.1 Parameters (likelihood and convergence)

Other slots of the zres object contain information on the parameter estimation: the estimated parameters, the log(likelihood) of the model, the BIC at convergence and the number of iterations:

- ..@mixc is a matrix with *nind* rows (the number of individuals) and K columns containing the estimated mixing coefficients M_k for each individual (in rows) and each class (in columns). If the `HBDClass == "Interval"`, only one mixing coefficient is reported for each group of layers;
- ..@krates is a matrix with *nind* rows (the number of individuals) and K columns containing the (estimated) R_k for each individual (in rows) and each class (in columns);
- ..@modlik is an array with the estimated log(likelihood) of the model at convergence for each individual;
- ..@modbic is an array with the estimated BIC of the model at convergence for each individual;

- ..@niter is an array with the number of iterations for parameter estimation for each individual (the total number of calls of the likelihood function).

Interpretation of the parameters is described in section 3.4.

To obtain the mixing coefficients M_k with the model with three classes and pre-defined rates:

```
bbb_mod2l@mixc
#>            [,1]         [,2]
#> [1,] 0.01024741 5.425527e-02
#> [2,] 0.03687469 3.103977e-03
#> [3,] 0.02961828 2.686068e-02
#> [4,] 0.03491979 5.825863e-03
#> [5,] 0.02206654 2.404143e-02
#> [6,] 0.09631071 5.533909e-06
```

Since rates R_k were pre-defined, they are all the same and correspond to the values specified by the model:

```
bbb_mod2l@krates
#>      [,1] [,2]
#> [1,]   10  100
#> [2,]   10  100
#> [3,]   10  100
#> [4,]   10  100
#> [5,]   10  100
#> [6,]   10  100
```

With the first model, the R_k vary across individuals (note that with one HBD class the same rate is used for the HBD and the non-HBD class as in Leutenegger et al. (2003)):

```
bbb_results3@krates
#>        [,1]
#> [1,] 73.283
#> [2,]  6.218
#> [3,] 24.438
#> [4,] 24.021
#> [5,] 35.496
#> [6,] 10.417
```

We can extract the log(likelihood) or the BIC as follows:

```
cbind(bbb_results3@modlik,bbb_results3@modbic)
#>           [,1]     [,2]
#> [1,] -6127.229 12271.98
#> [2,] -5767.659 11552.84
#> [3,] -5763.248 11544.01
#> [4,] -5809.963 11637.44
#> [5,] -5758.888 11535.29
#> [6,] -5383.741 10785.00
```

These likelihoods or BIC are most useful to compare models or parameter estimation procedures.

8.2 Realized autozygosity per class

The realized autozygosity per HBD class, or the partitioning of the genome into different HBD classes is one of the main outputs of the model. It is returned in the ..@realized slot as a matrix with $K + 1$ columns (the number of HBD classes and the non-HBD class) and *nind* lines (one per individual in the analysis). The values [i,j] are the genome-wide contributions (averaged over all positions) of class j to the genome of individual i. For IBD analysis, individuals are replaced by pairs of haplotypes.

```
bbb_mod2l@realized
#>             [,1]         [,2]      [,3]
#> [1,] 0.006002374 5.554389e-02 0.9384537
#> [2,] 0.043370796 2.986221e-03 0.9536430
#> [3,] 0.025913316 2.688936e-02 0.9471973
#> [4,] 0.025730270 6.792653e-03 0.9674771
#> [5,] 0.014428373 2.624476e-02 0.9593269
#> [6,] 0.100384227 5.210138e-06 0.8996106
```

The first individual has 0.60% of its genome in the first HBD class ($R_k = 10$), 5.55% in the second HBD class ($R_k = 100$) and 93.85% in the non HBD-class ($R_k = 100$). The second individual has more autozygosity in the first class (0.0434 or 4.34%) and the sixth individual has the highest level in the first class (10.04%).

With the one-HBD class model, there are only tow columns representing autozygosity versus allozygosity.

```
bbb_results3@realized
#>            [,1]      [,2]
#> [1,] 0.05852653 0.9414735
#> [2,] 0.04253223 0.9574678
#> [3,] 0.04469846 0.9553015
#> [4,] 0.03333417 0.9666658
#> [5,] 0.03563491 0.9643651
#> [6,] 0.10069500 0.8993050
```

In the case of a one-HBD class model, the mixing coefficients M_k and the realized autozygosity are more related but not identical. The first representing the proportion of segments that are HBD and the second the proportion of loci that are HBD.

```
cbind(bbb_results3@realized,bbb_results3@mixc)
#>            [,1]      [,2]       [,3]
#> [1,] 0.05852653 0.9414735 0.05798916
#> [2,] 0.04253223 0.9574678 0.04705781
#> [3,] 0.04469846 0.9553015 0.04472968
#> [4,] 0.03333417 0.9666658 0.03031428
#> [5,] 0.03563491 0.9643651 0.03517329
#> [6,] 0.10069500 0.8993050 0.09471100
```

A zres object obtained by running a model with more classes on 110 individuals from the Soay population genotyped for 47,365 SNPs can be loaded with the package. The default model with 10 layers was used to perform the analysis:

```
mix10L <- zoomodel()
mix10L
#> An object of class "zmodel"
#> Slot "typeModel":
#> [1] "mixkl"
#> 
#> Slot "mix_coef":
#>  [1] 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
#> 
#> Slot "krates":
#>  [1]    2    4    8   16   32   64  128  256  512 1024
#> 
#> Slot "err":
#> [1] 0.001
#> 
#> Slot "seqerr":
#> [1] 0.001
#> 
#> Slot "XM":
#>       [,1]
#>  [1,]    1
#>  [2,]    1
#>  [3,]    1
#>  [4,]    1
#>  [5,]    1
#>  [6,]    1
#>  [7,]    1
#>  [8,]    1
#>  [9,]    1
#> [10,]    1
#> 
#> Slot "typeClass":
#> [1] "SingleRate"
```

The results are stored in the soay_mix10l zres object. To obtain the realized autozygosity for the first ten individuals:

```
round(soay_mix10l@realized[1:10,],3)
#>       [,1]  [,2]  [,3]  [,4]  [,5]  [,6] [,7] [,8] [,9] [,10] [,11]
#>  [1,]    0 0.044 0.020 0.000 0.000 0.178    0    0    0     0 0.758
#>  [2,]    0 0.000 0.000 0.001 0.039 0.140    0    0    0     0 0.819
#>  [3,]    0 0.000 0.000 0.000 0.028 0.206    0    0    0     0 0.766
#>  [4,]    0 0.000 0.000 0.025 0.024 0.169    0    0    0     0 0.783
#>  [5,]    0 0.007 0.001 0.000 0.057 0.149    0    0    0     0 0.785
#>  [6,]    0 0.000 0.000 0.000 0.090 0.145    0    0    0     0 0.765
#>  [7,]    0 0.000 0.000 0.000 0.059 0.167    0    0    0     0 0.774
#>  [8,]    0 0.000 0.000 0.021 0.045 0.147    0    0    0     0 0.786
#>  [9,]    0 0.000 0.000 0.055 0.000 0.179    0    0    0     0 0.766
#> [10,]    0 0.000 0.000 0.061 0.000 0.171    0    0    0     0 0.768
```

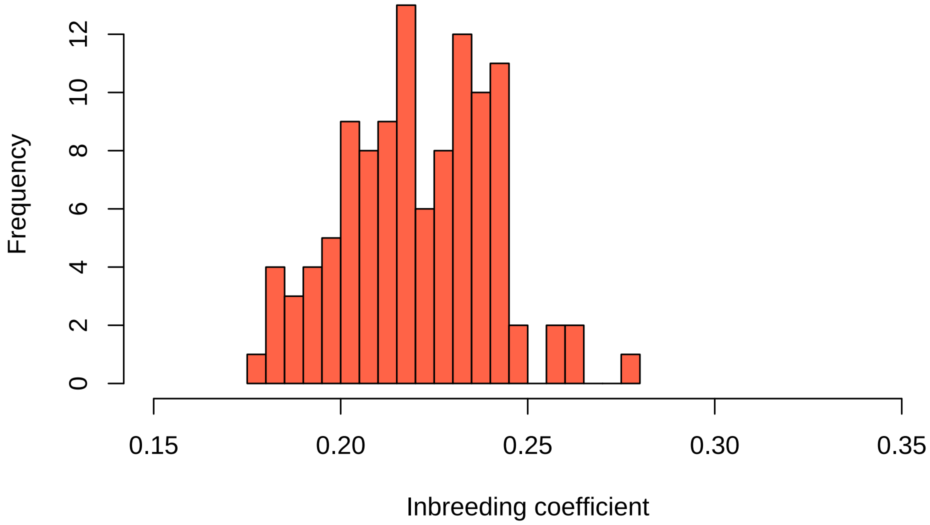
The columns represent the probability to belong to each class (averaged genome-wide). The columns represents the HBD classes with rates R_k equal to 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 and the non-HBD class (last column). The autozygosity is higher than 0.20 for most individuals (the non-HBD class accounts for less than 0.80) and is concentrated in HBD classes 5 and 6 (with rates 32 and 64).

8.3 Defining inbreeding coefficients (with respect to a base population)

The objective is not always to obtain the partitioning of autozygosity in different HBD classes (contribution from ancestors tracing back to different generations in the past). We are sometimes interested in the inbreeding coefficients (or the overall autozygosity). To estimate an inbreeding coefficient, we must define a base population. Segments inherited from ancestors present in generations more remote than the reference population will no longer be consider autozygous. With our model, we can decide that all HBD classes with a rate larger than a threshold T are not consider as autozygous. This amounts to set the base population at approximately 0.5*T generations in the past and to estimate an inbreeding coefficient F_{G-T}, for instance F_{G-50} would include all HBD classes with $R_k \leq 50$.

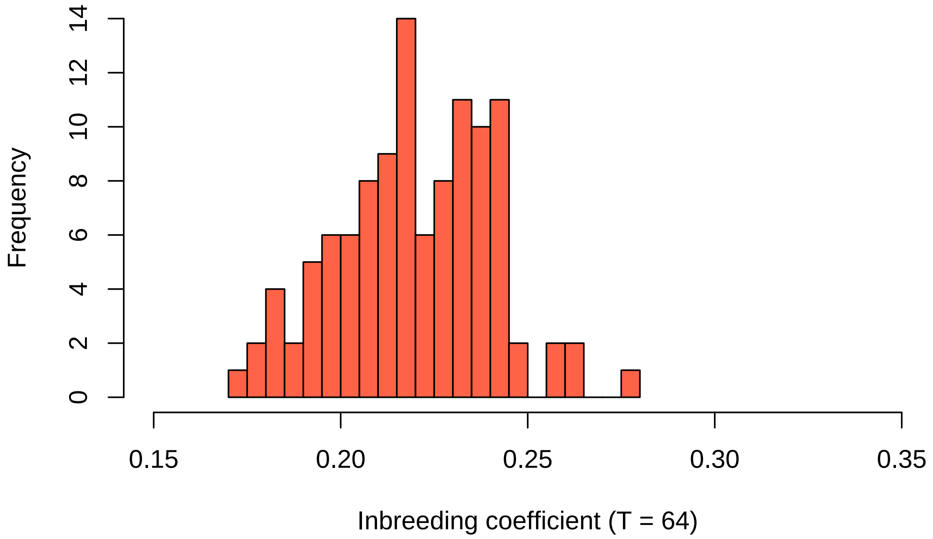
To obtain an inbreeding coefficient including all HBD classes, autozygosity must be summed over all HBD classes or can be obtained as 1 minus the non-HBD proportion.

```r
x <- 1-soay_mix10l@realized[,11]
dpar <- par()
hist(x,nc=20,main="",xlab="Inbreeding coefficient",xlim=c(0.15,0.35),col='tomato')
```



To obtain an inbreeding coefficient with respect to a threshold T, for instance at 64, autozygosity must be summed over the corresponding classes, with the cumsum function (or other functions).

```r
x <- t(apply(soay_mix10l@realized[,1:6],1,cumsum))
hist(x[,6],nc=20,main="",xlab="Inbreeding coefficient (T = 64)",
     xlim=c(0.15,0.35),col='tomato')
```



8.4 Local HBD probabilities

The locus specific HBD probabilities (probabilities to belong to an HBD class / state at a marker position) are stored in the ..@hbdp slot. This slot is a list of matrices, one matrix per individual in the analysis. Each matrix has a dimension K x $nsnp$ (the number of classes x number of SNPs). Element[i,j] represents the probability to be in class i at marker j. To access the matrix of one individual in the list, you must use "[[x]]". For instance, to extract HBD probabilities for the first 15 markers for individual 3:

```
t(bbb_mod2l@hbdp[[3]][,1:15])
#>                [,1]         [,2]      [,3]
#>  [1,] 4.588761e-06 3.563349e-05 0.9999598
#>  [2,] 1.135774e-05 6.979208e-04 0.9992907
#>  [3,] 7.340363e-06 5.028960e-04 0.9994898
#>  [4,] 2.152122e-08 1.298829e-06 0.9999987
#>  [5,] 2.347703e-07 1.193325e-05 0.9999878
#>  [6,] 7.024566e-05 3.724684e-03 0.9962051
#>  [7,] 3.813432e-04 2.212207e-02 0.9774966
#>  [8,] 4.207788e-04 2.476160e-02 0.9748176
#>  [9,] 4.763857e-04 2.865323e-02 0.9708704
#> [10,] 5.121785e-04 3.132471e-02 0.9681631
#> [11,] 5.301553e-04 3.302695e-02 0.9664429
#> [12,] 4.029736e-07 2.202109e-05 0.9999776
#> [13,] 6.517123e-07 4.878025e-05 0.9999506
#> [14,] 9.195540e-09 7.102927e-07 0.9999993
#> [15,] 2.532642e-08 1.977341e-06 0.9999980
```

Note that for readability, we transposed the results and lines are flipped in columns. After transposing, we obtain three columns, the probabilities for HBD class 1 ($R_k = 10$), for HBD class 2 ($R_k = 100$) and the non-HBD class. The HBD probabilities remain below 0.01 for most markers. An example of HBD segments (with high HBD probability) is found for individual 6:

```
t(bbb_mod2l@hbdp[[6]][,4700:4720])
#>            [,1]         [,2]         [,3]
#>  [1,] 0.9996947 2.597226e-08 0.0003052948
#>  [2,] 0.9997088 2.597553e-08 0.0002911718
#>  [3,] 0.9996862 2.561641e-08 0.0003137446
#>  [4,] 0.9996602 2.530187e-08 0.0003397778
#>  [5,] 0.9991953 1.834651e-08 0.0008046605
#>  [6,] 0.9997909 2.065980e-08 0.0002090340
#>  [7,] 0.9998113 2.072322e-08 0.0001886933
#>  [8,] 0.9997878 1.914133e-08 0.0002121389
#>  [9,] 0.9998018 1.872535e-08 0.0001981374
#> [10,] 0.9997735 1.789255e-08 0.0002264567
#> [11,] 0.9997756 1.789017e-08 0.0002243651
#> [12,] 0.9997265 1.759819e-08 0.0002735084
#> [13,] 0.9997584 1.817232e-08 0.0002416143
#> [14,] 0.9997434 1.847064e-08 0.0002565321
#> [15,] 0.9997235 1.865022e-08 0.0002764539
#> [16,] 0.9995594 1.903599e-08 0.0004405984
#> [17,] 0.9995729 2.084540e-08 0.0004270310
#> [18,] 0.9995346 2.101655e-08 0.0004653470
#> [19,] 0.9992074 2.058797e-08 0.0007926270
#> [20,] 0.9990529 2.062256e-08 0.0009471247
#> [21,] 0.9989748 2.034070e-08 0.0010251465
```

8.5 HBD segments

The HBD segments identified with the Viterbi algorithm are stored in the ..@hbdseg, a data frame with one line per identified HBD segment and with nine columns accessed with the $ symbol:

- ..@hbdseg$id is the number of the individual in which the HBD segments is located;
- ..@hbdseg$chrom is the chromosome number of the HBD segments (this chromosome number refers to the position of the chromosome in the list of all chromosomes present in the input genotype data);
- ..@hbdseg$start_snp is the number of the SNP at which the HBD segment starts (the SNP number within the chromosome);
- ..@hbdseg$start_end is the number of the SNP at which the HBD segment ends (the SNP number within the chromosome);
- ..@hbdseg$start_pos is the position at which the HBD segment starts (within the chromosome);
- ..@hbdseg$start_end is the position at which the HBD segment ends (within the chromosome);
- ..@hbdseg$number_snp is the number of consecutive SNPs in the HBD segment;
- ..@hbdseg$length is the length of the HBD segment (for instance in bp or in cM/1000000);
- ..@hbdseg$HBDclass is the HBD class associated with the HBD segment.

With the `ibd` option, the table contains IBD instead of HBD segments and the individual numbers are replaced by the numbers of the haplotype pairs (their position in the `ibdpairs` matrix).

The table of HBD segments identified in the Belgian Blue cattle subset with the `mod3r` model is:

```
dim(bbb_mod2l@hbdseg)[1]
#> [1] 47
head(bbb_mod2l@hbdseg[,1:8])
#>    id chrom start_snp end_snp start_pos   end_pos number_snp  length
#> 2   1     1         7      19    845494   2049400         13 1203907
#> 21  1     4       230     252 101677815 111368926         23 9691112
#> 1   1     6         1       9    202769   1014246          9  811478
#> 22  1     8       112     117  47179793  49524000          6 2344208
#> 23  1    11       204     210  94101466  95637394          7 1535929
#> 11  1    13         1      19    313308   3764223         19 3450916
```

There are 47 identified HBD segments (approximately 8 per individual). The first and second HBD segments are respectively 1.2 and 9.7 MB long and contain 13 and 23 SNPs. They belong both to the first individual and are located at the beginning of chromosome 1 and the end of chromosome 4.

To have summary statistics of length of HBD segments (in bp or in number of SNPs):

```
summary(bbb_mod2l@hbdseg$length)
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#>   627467  2783941  8958274 10518823 12071964 48659487
summary(bbb_mod2l@hbdseg$number_snp)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    6.00   16.00   21.00   29.91   32.50  145.00
```

The largest HBD segment is more than 48 Mb long, the average length is 10.5 Mb and the mean number of SNP per identified HBD segment is 29.9.

In the sheep population genotyped at higher density, the distribution of HBD segment presents some differences.

```
dim(soay_mix10l@hbdseg)[1]
#> [1] 16075
head(soay_mix10l@hbdseg[,1:8])
#>    id chrom start_snp end_snp start_pos   end_pos number_snp  length
#> 2   1     1       243     289  15944770  18888844         47 2944075
#> 4   1     1       312     328  20926901  22120789         17 1193889
#> 6   1     1       622     760  40342997  48373831        139 8030835
#> 8   1     1       787     800  50682135  51833033         14 1150899
#> 10  1     1      1222    1264  80119112  82137439         43 2018328
#> 12  1     1      1660    1704 108405452 111474679         45 3069228
```

There are 16,075 identified HBD segments (approximately 146 per individual). The first and second HBD segments are respectively 2.9 and 1.2 Mb long and contain 47 and 17 SNPs. They belong both to the first individual and are located on chromosome 1. The first individual has several segments on the first chromosome.

To obtain summary statistics of length of HBD segments (in bp or in number of SNPs):

```r
summary(soay_mix10l@hbdseg$length)
#>     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
#>    63092   1597052   2566229   3529729   4297704  74191678 
summary(soay_mix10l@hbdseg$number_snp)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>    2.00   26.00   40.00   53.21   64.00 1093.00
```

The largest HBD segment is more than 74 Mb long, the average length is 3.5 Mb and the mean number of SNP per identified HBD segment is 53.2.

8.6 Accessor functions (easier access to slots)

We propose several accessor functions to interact more conveniently with the zres object, in particular with the most useful slots (..@realized, ..@hbdp and ..@hbdseg).

8.6.1 For realized autozygosity To extract the table of realized autozygosity, you can use the function `realized` that takes as argument the name of the zres object and the numbers of the classes to be extracted (all by default). 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 value of the corresponding R_k rate. 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". To extract the entire table for the results in BBB cattle with the mod2l model:

```r
realized(bbb_mod2l)
#>          R_10        R_100    NonHBD
#> 1 0.006002374 5.554389e-02 0.9384537
#> 2 0.043370796 2.986221e-03 0.9536430
#> 3 0.025913316 2.688936e-02 0.9471973
#> 4 0.025730270 6.792653e-03 0.9674771
#> 5 0.014428373 2.624476e-02 0.9593269
#> 6 0.100384227 5.210138e-06 0.8996106
```

To extract the contribution of the first six HBD classes for the sheep analysis:

```r
head(round(realized(soay_mix10l,seq(1,6)),5))
#>       R_2     R_4     R_8    R_16    R_32    R_64
#> 1 0.00001 0.04360 0.02010 0.00000 0.00000 0.17785
#> 2 0.00001 0.00001 0.00006 0.00124 0.03925 0.14035
#> 3 0.00000 0.00000 0.00000 0.00000 0.02803 0.20565
#> 4 0.00000 0.00000 0.00009 0.02460 0.02379 0.16885
#> 5 0.00015 0.00678 0.00120 0.00002 0.05743 0.14909
#> 6 0.00000 0.00000 0.00000 0.00000 0.09019 0.14510
```

8.6.2 For inbreeding coefficients To obtain the inbreeding coefficient F_{G-T} as the sum of contributions from all HBD classes with a $R_k \leq T$, you can use the `cumhbd` function that takes as arguments the name of the zres object and the value of the threshold T (all HBD classes by default, when no threshold is specified).

To estimate the inbreeding coefficient in the sheep analysis by setting T equal to 100:

```r
F100 <- cumhbd(soay_mix10l, 100)
summary(F100)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.1749  0.2073  0.2197  0.2201  0.2352  0.2773
```

To estimate the inbreeding coefficient in the BBB cattle data and the analysis with the mod2l model including either the first HBD class or both of them:

```r
F10 <- cumhbd(bbb_mod2l, 10)
F100 <- cumhbd(bbb_mod2l, 100)
cbind(F10,F100)
#>              F10       F100
#> [1,] 0.006002374 0.06154626
#> [2,] 0.043370796 0.04635702
#> [3,] 0.025913316 0.05280267
#> [4,] 0.025730270 0.03252292
#> [5,] 0.014428373 0.04067313
#> [6,] 0.100384227 0.10038944
```

To estimate the inbreeding coefficient in the BBB cattle data but for the analysis performed with the 1 HBD class model. Computations are performed with respect to three values for T (20, 50 and 100) and with an estimated rate R_1 for each individual:

```r
F20 <- cumhbd(bbb_results3, 20)
#> [1] "cumhbd is not recommended for a model with different rates per individual!"
F50 <- cumhbd(bbb_results3, 50)
#> [1] "cumhbd is not recommended for a model with different rates per individual!"
F100 <- cumhbd(bbb_results3, 100)
#> [1] "cumhbd is not recommended for a model with different rates per individual!"
cbind(F20,F50,F100,bbb_results3@krates[,1])
#>             F20        F50       F100
#> [1,] 0.00000000 0.00000000 0.05852653 73.283
#> [2,] 0.04253223 0.04253223 0.04253223  6.218
#> [3,] 0.00000000 0.04469846 0.04469846 24.438
#> [4,] 0.00000000 0.03333417 0.03333417 24.021
#> [5,] 0.00000000 0.03563491 0.03563491 35.496
#> [6,] 0.10069500 0.10069500 0.10069500 10.417
```

We first observe that the function does not recommend to estimate an inbreeding coefficient when the R_k rates are estimated. This is because in that situation, each individual has different HBD classes and then the number of classes included in the estimation would vary according to the individuals. Here, only individuals 2 and 6 have a non-zero inbreeding coefficient when setting T equal to 20. All individuals have a non-zero inbreeding coefficient with T equals to 50 with the exception of the first individual. Finally, all the estimated autozygosity is considered when T equals 100.

8.6.3 For HBD segments The `rohbd` function can help to extract all HBD segments for a group of individuals and a genomic region. By default HBD segments for all individuals and the entire genome are extracted. The individuals are specified by ids = x where x is a vector with the individual numbers. The region is specified by chrom = y where y is the chromosome number. The coordinates can also be specified with the startPos and endPos arguments (these are not used if the chrom is NULL).

Two methods of extraction can be used with the inside logical arguments. The function extracts only those segments with start and end positions within the specified region (inside = TRUE) or extracts all HBD segments that overlap the region (inside = FALSE). Output is a data.frame with the same format as the ..@hbdseg slot (see description in 7.4).

To extract all HBD segments identified in the sheep population overlapping the region from 10 to 20 Mb on chromosome 25.

```
roh25_10_20 <- rohbd(zres = soay_mix10l, chrom = 25,
                     startPos= 10000000,endPos = 20000000, inside = FALSE)
dim(roh25_10_20)
#> [1] 88  9
head(roh25_10_20[,1:8])
#>      id chrom start_snp end_snp start_pos  end_pos number_snp  length
#> 420   2    25       132     196   9657834 13757998         65 4100165
#> 617   2    25       225     278  15495515 18657658         54 3162144
#> 4193  5    25       229     315  15856692 21870388         87 6013697
#> 2314  7    25       259     315  17689768 21870388         57 4180621
#> 4175  8    25       265     295  17973845 20026839         31 2052995
#> 6155  9    25       244     285  16935804 18991436         42 2055633
summary(roh25_10_20$length)
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
#>   935966  2308713  3134047  4392371  5340193 16846298
```

There are 88 HBD segments overlapping the region, with a mean length of 4.4 Mb and the longest segments is more than 16 Mb. If we restrict the extraction on HBD segments inside the region:

```
roh25_10_20 <- rohbd(zres = soay_mix10l, chrom = 25,
                     startPos= 10000000,endPos = 20000000, inside = TRUE)
dim(roh25_10_20)
#> [1] 38  9
head(roh25_10_20[,1:8])
#>       id chrom start_snp end_snp start_pos  end_pos number_snp  length
#> 617    2    25       225     278  15495515 18657658         54 3162144
#> 6155   9    25       244     285  16935804 18991436         42 2055633
#> 4203  12    25       244     291  16935804 19655167         48 2719364
#> 41412 13    25       225     266  15495515 17997283         42 2501769
#> 22411 14    25       184     196  12822033 13757998         13  935966
#> 41710 14    25       244     285  16935804 18991436         42 2055633
summary(roh25_10_20$length)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>  935966 2055633 2323904 2443090 2711511 4863075
```

There are only 38 HBD segments and the longest is less than 5 Mb in size. Extraction, can also be performed per individual for the entire genome. For instance, for individuals 56, 15, 97, 103 and 108 we find 755 HBD segments (the first individual column is now 15):

```
roh25_10_20 <- rohbd(zres = soay_mix10l, ids = c(15,56,97,103,108))
dim(roh25_10_20)
#> [1] 755   9
```

```
head(roh25_10_20[,1:8])
#>      id chrom start_snp end_snp start_pos   end_pos number_snp   length
#> 2180 15     1       177     204  11540961  13301343         28  1760383
#> 4100 15     1       252     314  16471993  21164217         63  4692225
#> 690  15     1      1046    1149  68918632  76227410        104  7308779
#> 890  15     1      1878    2075 124296584 137186574        198 12889991
#> 1090 15     1      2079    2115 137752062 139783950         37  2031889
#> 1279 15     1      2592    2658 172252836 176779276         67  4526441
```

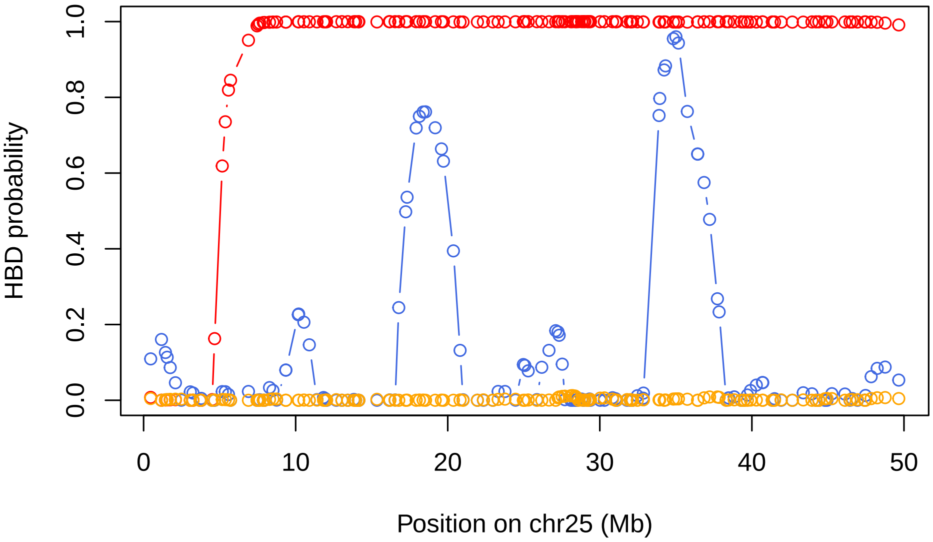
8.6.4 For local HBD probabilities (locus specific)

The `probhbd` function help to extract the local HBD probabilities for one individual in a specific region. HBD probabilities represent large amount of data. Therefore, we extract only a total HBD probability by summing over all HBD classes. As for the inbreeding coefficient, the user can specify a threshold T to determine which HBD classes should be used in the sum and considered as autozygous (all by default).

The function takes as arguments the `zres` object but also the `zooin` object (that contains some information on the markers, their position and the chromosome number). The individual is indicated with the id = . argument. The same arguments as for the `rohbd` function are used to declare the genomic region (chrom, startPos, endPos). By default all positions are extracted.

Here is an example to extract HBD probabilities for the sixth individual for the BBB cattle data, on chromosome 19 from position 10 Mb to 20 Mb and plot it (we need also to extract the position from the zooin@bp slot). In addition, we extract HBD probabilities for two other individuals:

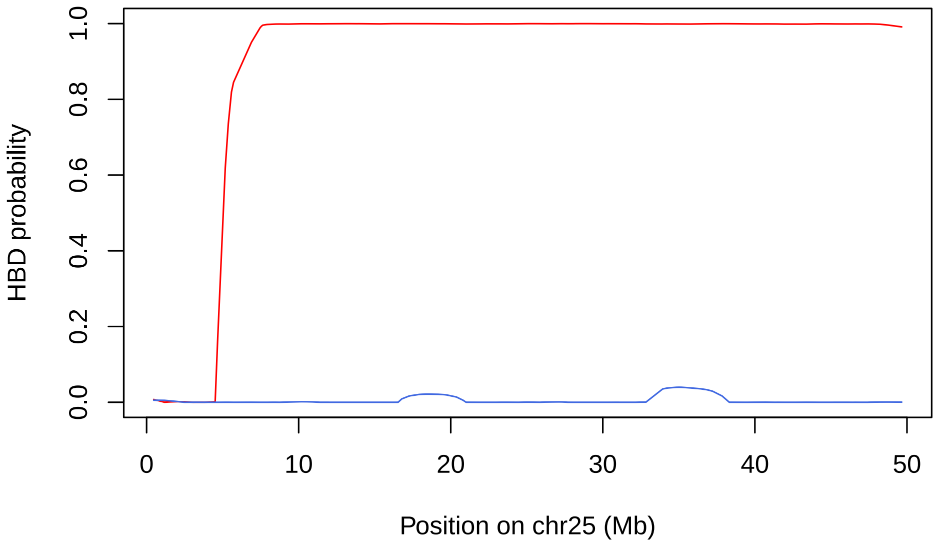
```
y6 <- probhbd(zres = bbb_mod2l, zooin = bbb, id = 6, chrom = 19,
              startPos = 0, endPos = 50000000)
x <- bbb@bp[bbb@chrbound[19,1]:bbb@chrbound[19,2]]
x <- x[x >=0 & x<= 50000000]/1000000
plot(y6~x,type='b',ylim=c(0,1),ylab='HBD probability',col='red',
     xlab='Position on chr25 (Mb)')
y1 <- probhbd(zres = bbb_mod2l, zooin = bbb, id = 1, chrom = 19,
              startPos = 0, endPos = 50000000)
y2 <- probhbd(zres = bbb_mod2l, zooin = bbb, id = 2, chrom = 19,
              startPos = 0, endPos = 50000000)
par(new=TRUE)
plot(y1~x,type='b',ylim=c(0,1),ylab='',col='royalblue',xlab='',axes=FALSE)
par(new=TRUE)
plot(y2~x,type='b',ylim=c(0,1),ylab='',col='orange',xlab='',axes=FALSE)
```



We observe that individual 6 has a long HBD segment, whereas there is no evidence of HBD segments in individual 2. For the first individual, we see two small regions were the HBD probabilities reach approximately 0.80 and 0.95. These probabilities suggest presence of HBD segment that would not have been captured with window-based approaches because they contain too few markers.

The use of the threshold T determines which classes are included. In the previous example, we used two HBD classes. By setting T = 20, we would use only one HBD class with $R_k = 10$ (and not the class with $R_k = 100$):

```
y6b <- probhbd(zres = bbb_mod2l, zooin = bbb, id = 6, chrom = 19,
              startPos = 0, endPos = 50000000,T=20)
y1b <- probhbd(zres = bbb_mod2l, zooin = bbb, id = 1, chrom = 19,
              startPos = 0, endPos = 50000000,T=20)
plot(y6b~x,type='l',ylim=c(0,1),ylab='HBD probability',col='red',
     xlab='Position on chr25 (Mb)')
par(new=TRUE)
plot(y1b~x,type='l',ylim=c(0,1),ylab='',col='royal blue',xlab='',axes=FALSE)
```



The curve is almost identical for individual 6 because the long HBD segment is associated to the first HBD class (rate $R_k = 10$). The HBD probabilities for the first individual are now equal to 0, because the evidence for short segments previously observed is associated with the second HBD class, with a higher rate (100) corresponding to shorter segments.

8.6.5 Using accessor functions with the IBD analysis The same accessor functions can be used after a `zoorun` analysis performed on pairs of haplotypes (`ibd`) option. The interpretation of the results must be adapated as IBD replaces HBD and individuals (and their number) are replaced by pairs of haplotypes.

8.6.6 Combining accessor functions with standard summary functions The accessors can be combined with standard summary functions to obtain summary of the results (realized autozygosity, inbreeding coefficients and the HBD segments). We already had some examples with histograms of the inbreeding coefficients (see 8.6.2).

To obtain a summary of the realized inbreeding:

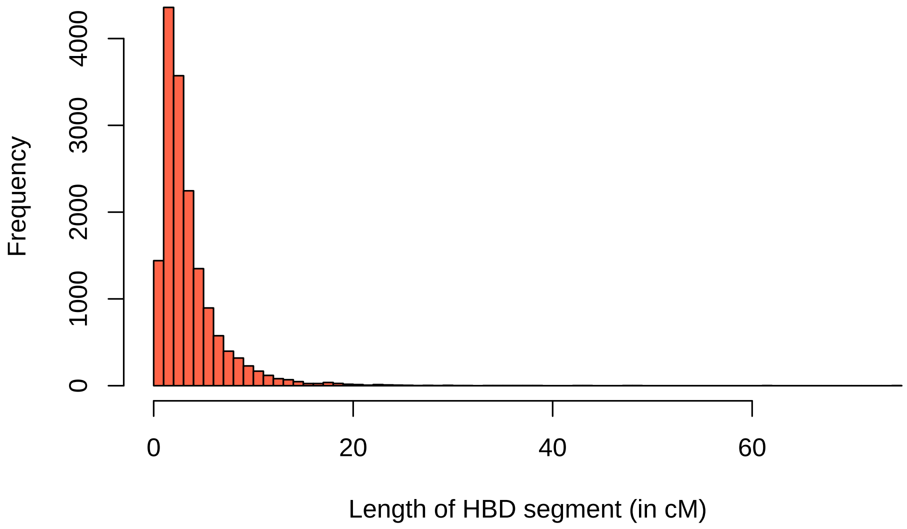
```
summary(realized(soay_mix10l))
#>       R_2                  R_4                  R_8
#>  Min.   :0.000e+00   Min.   :0.000e+00   Min.   :0.000e+00
#>  1st Qu.:4.390e-09   1st Qu.:2.000e-08   1st Qu.:1.000e-07
#>  Median :4.483e-08   Median :9.000e-08   Median :1.350e-06
#>  Mean   :2.191e-06   Mean   :1.172e-03   Mean   :3.000e-03
#>  3rd Qu.:3.591e-07   3rd Qu.:8.200e-07   3rd Qu.:5.320e-05
#>  Max.   :1.499e-04   Max.   :5.792e-02   Max.   :4.368e-02
#>       R_16                 R_32                 R_64               R_128
#>  Min.   :0.000e+00   Min.   :0.000e+00   Min.   :0.07455   Min.   :0.000e+00
#>  1st Qu.:2.890e-06   1st Qu.:2.902e-05   1st Qu.:0.14606   1st Qu.:0.000e+00
#>  Median :4.193e-03   Median :2.257e-02   Median :0.16848   Median :1.100e-08
#>  Mean   :1.840e-02   Mean   :3.252e-02   Mean   :0.16499   Mean   :5.840e-04
#>  3rd Qu.:3.633e-02   3rd Qu.:5.506e-02   3rd Qu.:0.18314   3rd Qu.:5.100e-07
#>  Max.   :7.116e-02   Max.   :1.645e-01   Max.   :0.22566   Max.   :2.824e-02
#>      R_256               R_512               R_1024              NonHBD
#>  Min.   :0.000e+00   Min.   :0.000e+00   Min.   :0.000e+00   Min.   :0.7227
#>  1st Qu.:0.000e+00   1st Qu.:0.000e+00   1st Qu.:0.000e+00   1st Qu.:0.7648
#>  Median :0.000e+00   Median :0.000e+00   Median :0.000e+00   Median :0.7803
#>  Mean   :5.171e-09   Mean   :1.926e-09   Mean   :1.537e-09   Mean   :0.7793
#>  3rd Qu.:2.130e-11   3rd Qu.:9.000e-13   3rd Qu.:1.040e-12   3rd Qu.:0.7927
#>  Max.   :3.270e-07   Max.   :1.637e-07   Max.   :1.349e-07   Max.   :0.8214
```

Similarly, for inbreeding coefficients with T equal to 100:

```
summary(cumhbd(soay_mix10l,100))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.1749  0.2073  0.2197  0.2201  0.2352  0.2773
```

The `summary` function can also be used with the `hbdseg` data.frame (for the entire frame or for some variables). We can also use a function to plot an histogram of length of HBD segments:

```
allseg <- rohbd(zres = soay_mix10l)
summary(allseg$length)
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#>    63092  1597052  2566229  3529729  4297704 74191678
hist(allseg$length/1000000,xlab="Length of HBD segment (in cM)",main="",
     col='tomato',nc=100)
```



8.6.7 Merging and updating zres objects Finally, the `merge_zres` function enables the merging of several zres objects obtained from the same dataset, while the `update_zres` function enables the modification of the values in a zres object using new results obtained for the same individuals (for example, when a convergence issue occurred in the first run).

9 Plotting

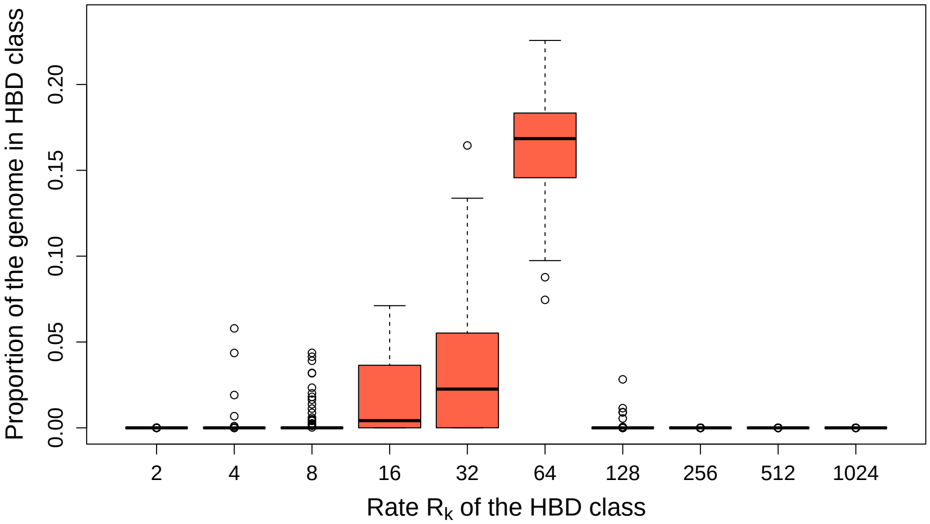
Four plotting functions are helpful to interpret the results. They plot either the partitioning of the genome in HBD class at the population or the individual level, the contribution of different classes to the genome or HBD segments. We recommend to use the functions representing proportion of the genome in different HBD classes or partitioning of the genome in HBD classes **only with models with pre-defined rates** (see section 6).

9.1. Proportion of the genome associated with different HBD classes (population)

The `zooplot_prophbd` represents the proportion of the genome associated with the different HBD classes at the population level. The input is a named list of zres object (list(name1 = zres1, name2 = zres2, . . .)). The list can contain one or several populations. When provided, the names are used in the plot. Three format can be used with the `style` arguments: "barplot", "lines" and "boxplot" (boxplot can only be use with a single zres object).

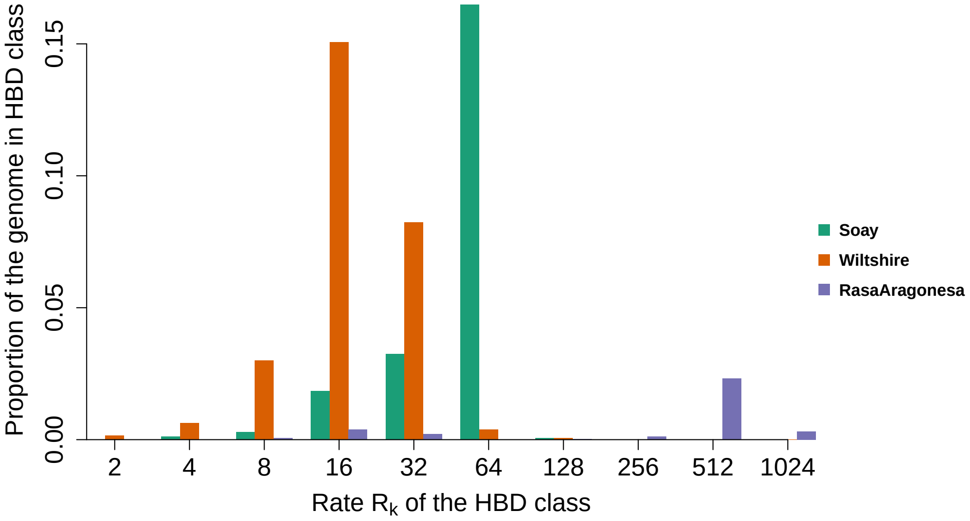
To plot a single population with boxplots:

```
zooplot_prophbd(list(Soay = soay_mix10l), cols = 'tomato', style = 'boxplot')
```



To plot three populations with barplots:

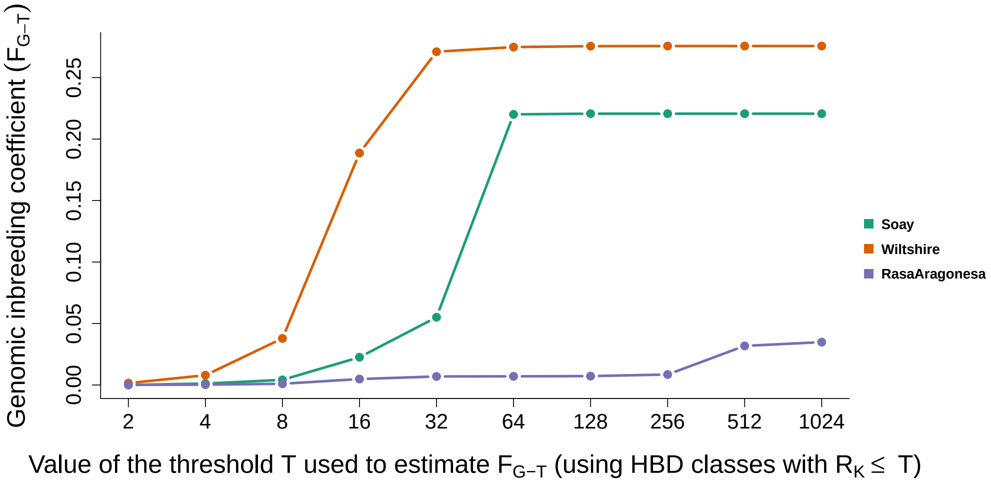
```
zooplot_prophbd(list(Soay=soay_mix10l,Wiltshire=wilt_mix10l,
                     RasaAragonesa=rara_mix10l),style='barplot')
```



The plot can also represents cumulative proportions: the fraction of the genome in all HBD classes with rates $R_k \leq T$. For the plot, we use the rate of the HBD classes as values for T. So, we obtain the fraction of the genome in the first HBD class, the two first HBD classes, the first three HBD classes, etc. These values can be interpreted as inbreeding coefficients estimated with respect to different base populations (see sections 3.4, 8.3, 8.6.2 or Druet and Gautier (2017) and Solé et al. (2017) for more details).

To plot the average inbreeding coefficients (cumulative values) for three populations with lines:

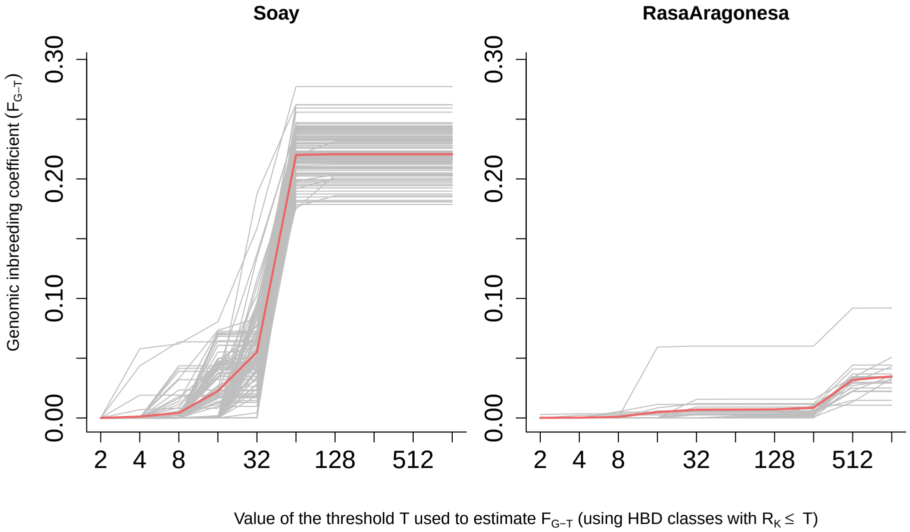
```
zooplot_prophbd(list(Soay=soay_mix10l,Wiltshire=wilt_mix10l,
                     RasaAragonesa=rara_mix10l),style='lines', cumulative = TRUE)
```



9.2. Proportion of the genome associated with different HBD classes (individuals)

The `zooplot_individuals` function represents the same results as the `zooplot_prophbd` function but at an individual level. Each individual is represented by a line (no barplots or boxplots). The function plots either the proportion of the genome in different HBD classes or cumulative proportions (see above) if `cumulative` is set to `TRUE`. The average value of the population is added in red. As before, the input is a named list of zres object (list(name1 = zres1, name2 = zres2, . . .)). The list can contain one or several populations. When provided, the population names are used in the plot. The `ncols` argument determines the number of graphs (populations) plotted next to each other.

```
pop2 <- list(Soay=soay_mix10l,RasaAragonesa=rara_mix10l)
zooplot_individuals(pop2, cumulative = TRUE)
#> Warning in zooplot_individuals(pop2, cumulative = TRUE):
#> All individuals are plotted; use toplot to select individuals
```



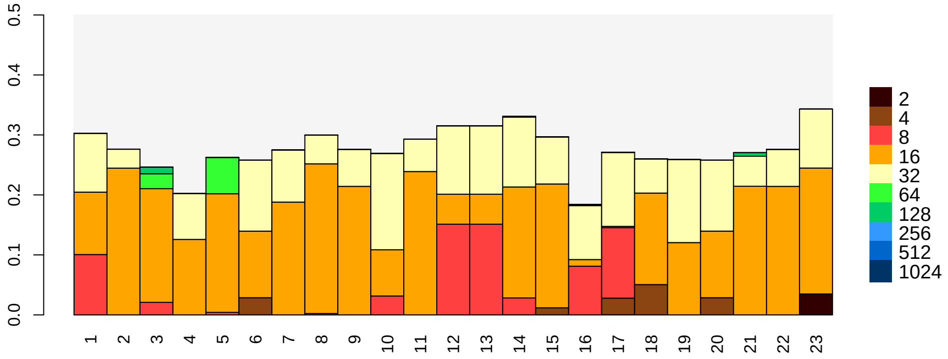
We observe than Soay sheep present higher levels of autozygosity and that patterns are relatively homogeneous in both populations.

9.3. Partitioning individual genomes in different HBD classes

The `zooplot_partitioning` function represents for each individual the proportion of the genome in each HBD class. Each individual is represent as a stacked barplot. The total height represents the total autozygosity level. The contribution of each HBD class (in % of the genome) is represented as a bar of a different color (black, brown, red for the most recent HBD classes).

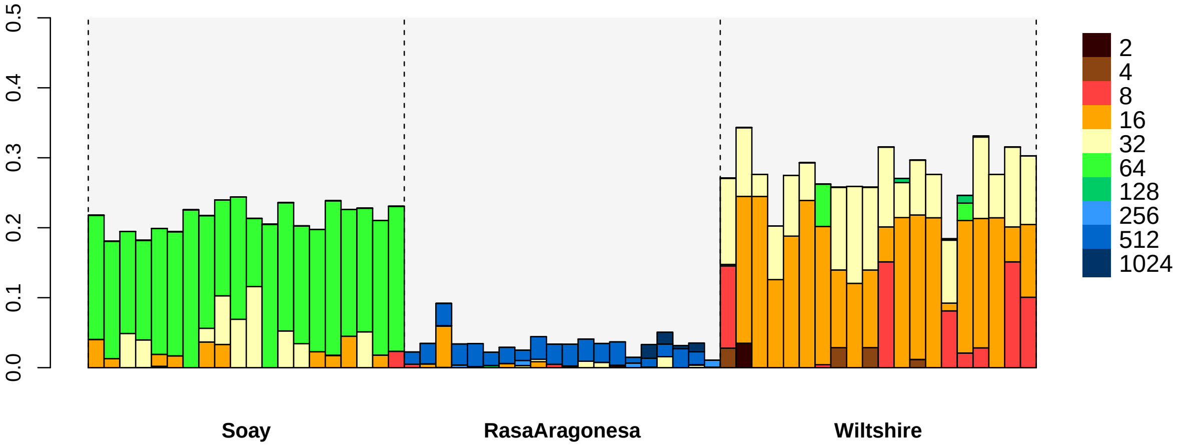
The input is again a named list of zres object (list(name1 = zres1, name2 = zres2, . . .)). The list can contain one or several populations. When provided, the population names are used in the plot. The user can provide the colors of the bars (argument `cols`), can choose to plot the ids or not (argument `plotids`), can give a list of individuals to plot (argument `toplot`, a list of vectors containing the individuals to plot for each population), can choose to plot a random sample of individuals (argument `randomids` set to TRUE with `nrandom` a vector containing the number of individuals to select for each population and `seed` being the random seed). The `ylim` argument indicates the minimal and maximal values of the y-axis, the `border` argument indicates whether a border is drawn around each bar, the `nonhbd` argument indicates whether a border is drawn around the non-hbd contribution and the `vertical` argument specifies if sample names are written vertically or not. To plot the partitioning in the Wiltshire population:

```
zooplot_partitioning(list(Wiltshire=wilt_mix10l), ylim = c(0,0.5), nonhbd = FALSE)
```



Next we plot the three sheep population. To reduce the number of individuals, we select randomly 20 individuals per population and we don't print the ids.

```
pop3 <- list(Soay=soay_mix10l,RasaAragonesa=rara_mix10l,Wiltshire=wilt_mix10l)
zooplot_partitioning(pop3, randomids = TRUE, nrandom = c(20,20,20), plotids = FALSE,
                     ylim=c(0,0.5), nonhbd = FALSE)
#> Warning in zooplot_partitioning(pop3, randomids = TRUE, nrandom = c(20, :
#> Random seed 100 is used to sample individuals.
```



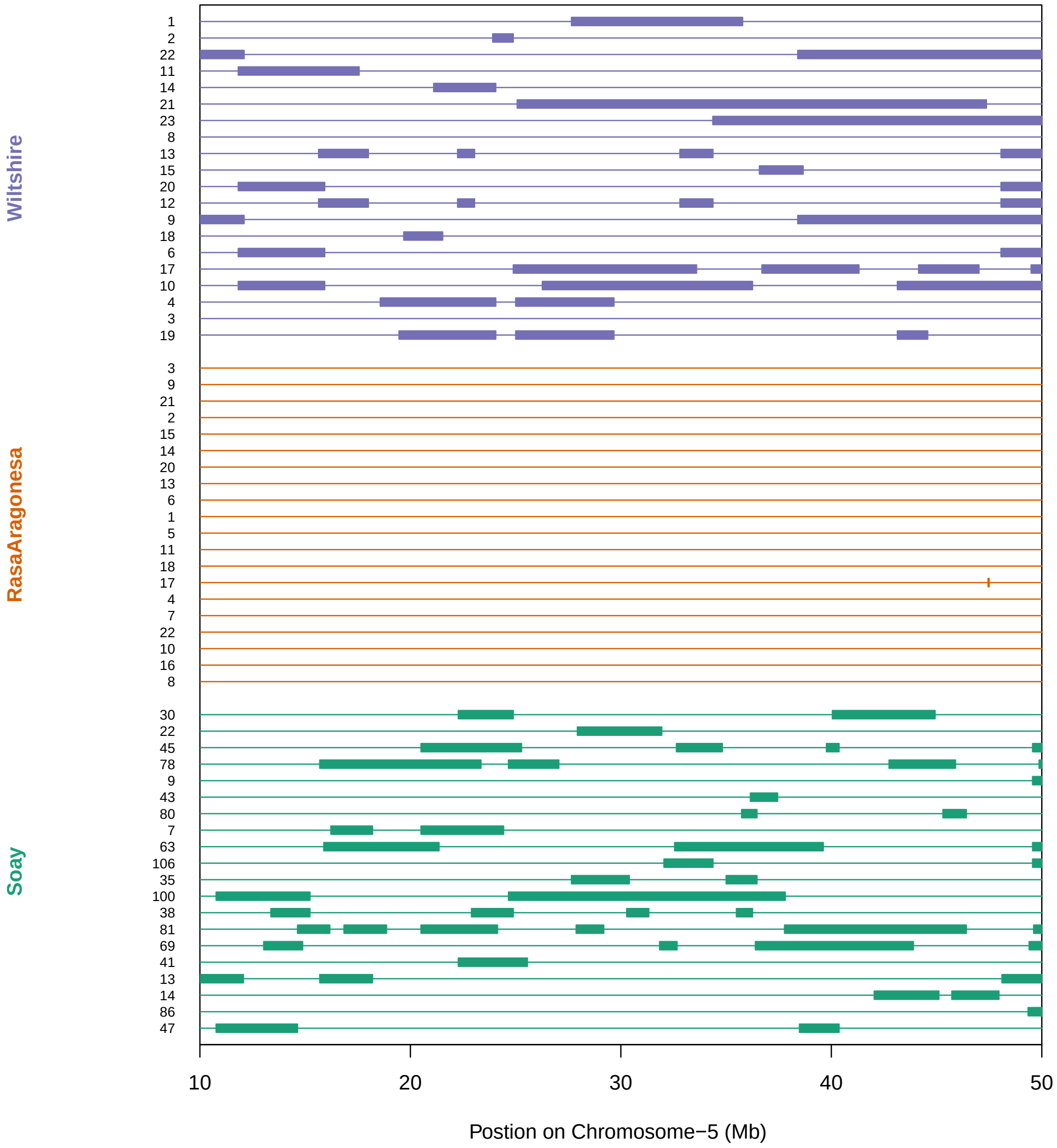
Soay and Wiltshire populations present higher levels of autozygosity. The Wiltshire individuals present more recent autozygosity (brown, red, orange bars) compared to the Soay individuals where green ($R_k = 64$) dominates, indicating that the Wiltshire individual have more recent common ancestors causing autozygosity.

9.4. Plotting identified HBD segments

The zooplot_hbdseg function plots the identified segments for a selected region. The region is specified with the chr and coord arguments. A minimal segment size can be selected with the minlen arguments.

Some arguments are identical to the zooplot_partitioning function: those to select individuals, randomids, nrandom, seed, toplot, plotids. The cols argument allows to specify the color used for each population or zres object in the input list.

```
zooplot_hbdseg(pop3, randomids = TRUE, nrandom = c(20,20,20),
               chr=5,coord=c(10000000,50000000))
#> random seed  100  is used to sample ids
```



10 Impact of data quality and informativity - some caution with whole genome sequecing and reduced representation sequecing

The genotyping technology will impact the amount of available data to characterize autozygosity and identify HBD segments. The method will be more efficient with more SNPs per HBD segment, with less genotyping / sequencing errors, when markers are more informative (higher minor allele frequency - MAF) or when sequencing is performed at higher coverage. There are no simple guidelines indicating how well the method will perform as it depends on many factors specific to data sets. Still we performed some simulations to asses the impact of different factors on the mean absolute errors (MAE) of HBD probabilities, the absolute value of the difference between the HBD probability and the true HBD status (Druet and Gautier, 2017). The results are summarized in Figure 4, that can help to understand the impact of lower marker density, marker informativity or sequencing coverage and to estimate the number of SNP required per HBD segment to obtain descent estimates.

With genotyping data, the HBD segments are accurately identified with 50 or 100 SNPs per segments (as for window-based RoH) and with 20 SNPs per segment the method is still efficient but in that case, false positive and false negative rates are higher and the use of HBD probabilities is recommended (it is risky to use window-based RoH with only 20 SNPs per window). If we use sequencing data, we observe that with low cover, the number of marker per HBD segments must be increased to obtain similar accuracy.

We can observe that genotyping error rates have a small impact because they are accounted for in the model. Still, with higher error rate (0.01) we observe that the number of false positive HBD segments increases because heterozygous genotypes are less penalized, the model allows approximately one heterozygous SNP per 100 SNPs (the accuracy in HBD segments remains however unchanged). Similarly, when the overall autozygosity is higher the risk of false positive HBD segments increases because since HBD segments are more frequent, the probability to observe HBD segments is higher. In that case, we also observe that the power to identify the HBD segments increases in parallel.

In addition to these simulations, we show results obtained with low-fold sequencing in Belgian Blue beef cattle to illustrate with real data that the approach can indeed capture HBD segments even with low-fold sequencing. Figure 5 represents for 46 individuals HBD segments obtained with genotyping arrays (~ 2-3 SNPs per Mb, ~ 10 SNPs per Mb and ~ 200 SNPs per Mb) and with whole genome sequencing (10x, 0.5x and 1x). The results from the 1x are similar to those of the highest SNPs density (in terms of number of segments and their length) whereas the results from the 0.5x lie in between those obtained with ~10 SNPs per Mb and those obtained with ~ 200 SNPs per Mb.

Although the method has been shown to be efficient when marker density or marker spacing is variable (such as in reduced representation sequencing data), analysis on **real reduced representation sequencing data (GBS) requires some caution**. Indeed, we observed higher genotyping error rates, possibly due to allelic dropout. If genotyping error rates become higher (and the levels are unknown), it becomes more difficult to correctly estimate HBD (the same would apply to rule-based RoH). Allelic dropout would result in heterozygous genotypes being incorrectly reported as homozygous genotypes, thereby increasing homozygosity levels. We did observe that estimated HBD levels increased, but mainly in more distant HBD classes (background HBD levels in short segments). This would be expected if genotyping errors were randomly distributed. In this case, relying on the more recent HBD segments would be robust. In addition, the reduced representation sequencing data contains **some variants that were not identified by whole-genome sequencing of the same individuals**. The presence of these variants in the analysis reduced the quality of the HBD estimatation.

Another important consideration is **the quality of the genetic or physical map**. If the data still contains some "poor quality" regions or incorrectly located segments, long HBD segments may be broken into shorter ones. Therefore, the distribution of HBD segment lengths will be incorrect and so will the partitioning into different HBD classes. We observed on some cattle whole genome sequencing data that with limited data filtering, **HBD levels were concentrated in HBD classes of shorter length**, whereas strict filtering allowed some more recent HBD segments to be identified. In many cases, it is better to keep fewer high-quality markers (without genotyping errors, without map errors, etc) than keep more markers. Whole-genome sequence data is not required to capture inbreeding (even associated with ancestors present 100 generations ago) and high-density marker maps are sufficient (with low-fold sequencing, the properties are somewhat different and more markers are needed).

11 Predicting HBD in future offspring and kinship estimation with `zookin`

The kinship between two individuals can be estimated by applying the `zooroh` model to the four possible pairs of parental haplotypes (one chromosome from each individual) and then averaging these IBD probabilities. These are in fact the probability that two alleles sampled at random in the two individuals are IBD. This can provide genome-wide estimates, but also locus-specific information. The kinship between two parents is also an estimator of the future inbreeding of their offspring. This approach is described in more details in Forneris et al. (2025) and has been shown to provide good estimates. The length-based classification of IBD will also provide additional information (as for HBD segments).

This approach can be done by using the `zoorun` function with the `ibd` option and specifying the four haplotype pairs. The results of the four haplotype pairs must then be combined to estimate relatedness of the pairs of individuals. To facilitate this process, we have implemented the `zookin` function, which is very similar to the `zoorun` function, and takes almost the same arguments.

11.1 Running the model

The options / parameters that are identical are `zoomodel` (mandatory) specifying the model, `zooin` (mandatory) indicating the name of the `zdata` object, `parameters`, `fb`, `vit`, `localhbd`, `nT`, `optim_method`, `maxiter`, `minmix`, `maxr` and `RecTable`. See the `zoorun` function for more information. The specific option is `kinpairs` that is used to specify a matrix with the pairs of individuals to be analyzed.

Since this function calls `zoorun` with the `ibd` option, it is only possible to use phased data (`zformat` equals to "vcf" or "haps"). The function is not compatible with `KL` models (because the rates would be different for every haplotype pair).

For example, in order to estimate the kinship with the phased cattle data (TAF), we start by defining the matrix with the pairs of individuals we want to analyze and then use the `zookin` function. Here we estimate the relationships between the four individuals:

```
targets <- as.matrix(cbind(c(1,1,1,2,2,3),c(2,3,4,3,4,4)))
TKIN <- zookin(Mod2L, DTAF, kinpairs = targets)
#> final  value 32.209552
#> converged
#> final  value 23.680273
#> converged
#> final  value 32.209552
#> converged
#> final  value 23.680273
#> converged
#> final  value 24.673525
#> converged
#> final  value 24.673525
#> converged
#> final  value 24.673525
#> converged
#> final  value 24.673525
#> converged
#> final  value 30.741951
#> converged
#> final  value 23.862595
```

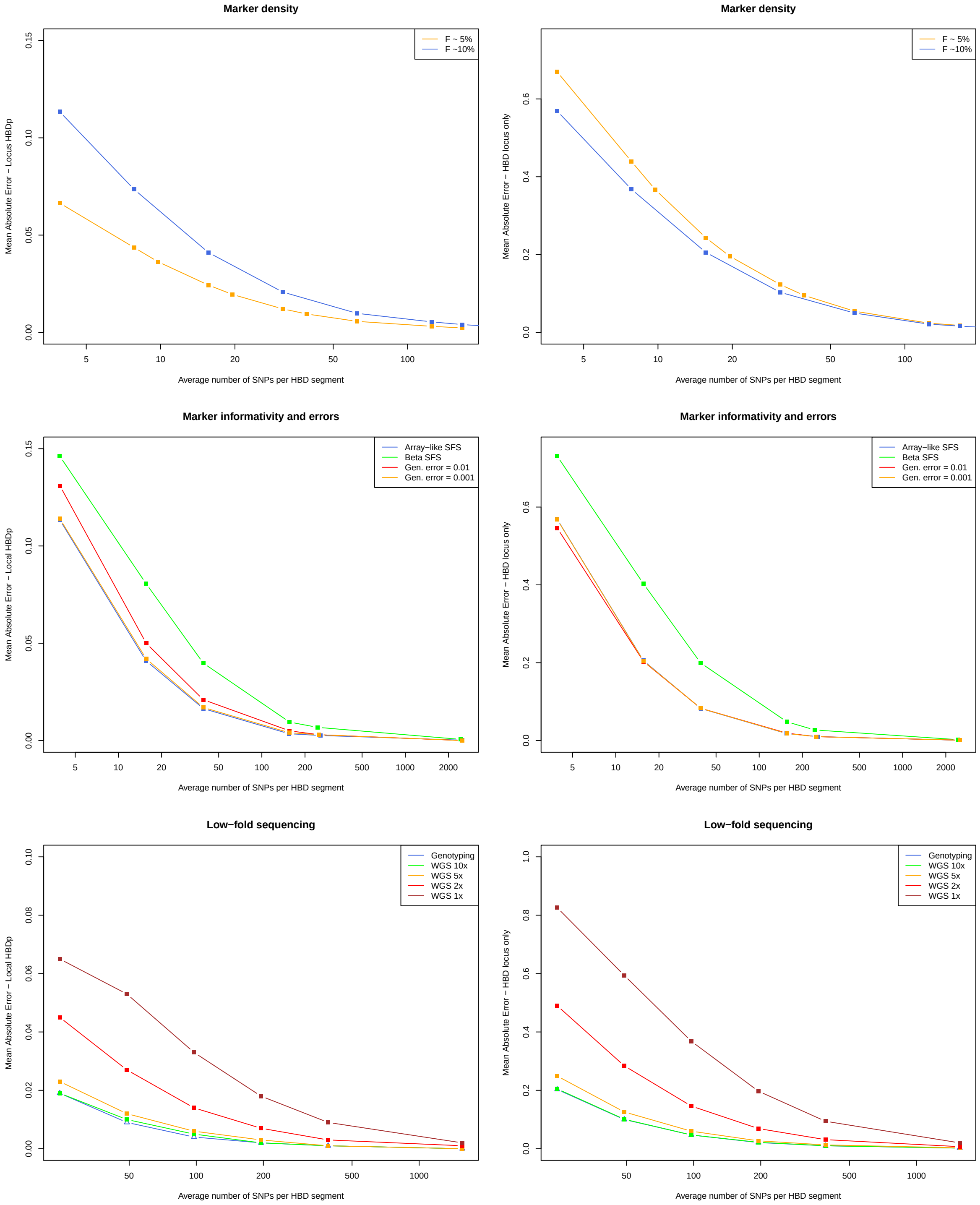


Figure 4: Impact on marker informativity on mean absolute error (MAE) for local HBD probabilities for all locus (left column) or for HBD locus only (right column) based on results from results from Druet and Gautier (2017). The figure shows the impact of average number of SNPs per HBD segment, site frequency spectrum (SFS), presence of genotyping errors or coverage with whole-genome sequencing (WGS) data.

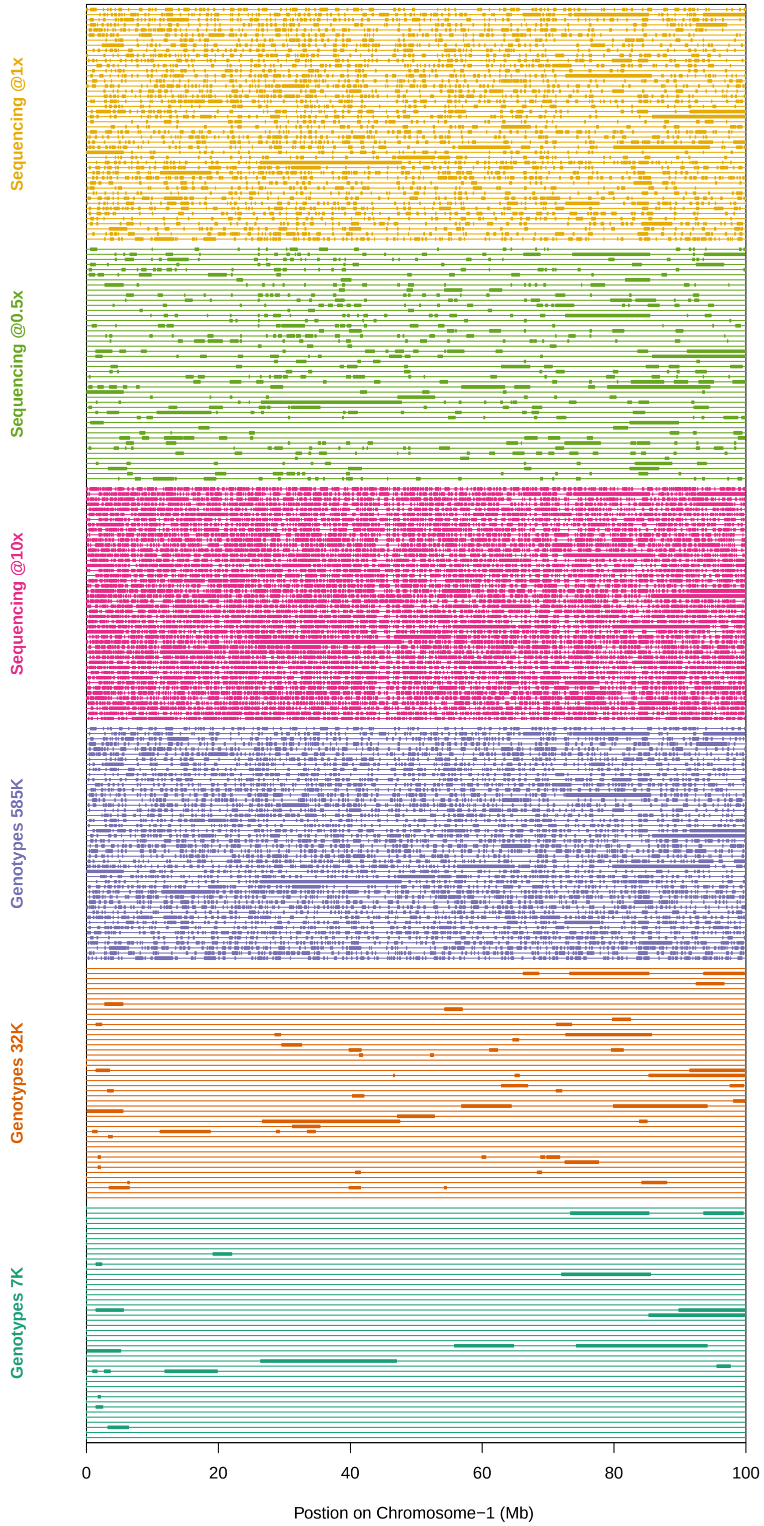


Figure 5: HBD segments identified on chromosome 1 using different genotyping densities or using whole-genome sequencing at different coverages in 46 Belgian blue sires.

```
#> converged
#> final  value 30.741951
#> converged
#> final  value 23.862595
#> converged
#> final  value 21.835269
#> converged
#> final  value 21.835269
#> converged
#> final  value 18.401270
#> converged
#> final  value 18.401270
#> converged
#> final  value 22.310363
#> converged
#> final  value 21.024339
#> converged
#> final  value 22.395340
#> converged
#> final  value 17.282520
#> converged
#> final  value 22.017591
#> converged
#> final  value 18.342449
#> converged
#> final  value 22.017591
#> converged
#> final  value 18.342449
#> converged
```

The code for the full data including 18 cows from the population would be:

```
taf3 <- zoodata("taf_phased.vcf",zformat="vcf", min_maf = 0.001)
mix4L <- zoomodel(K=4,base_rate=5)
kintaf_mix4l <- zookin(mix4L,taf3,kinpairs=mykinpairs,vit=FALSE,nT=4)
```

The `zdata` object contains 23679 polymorphic SNPs. We specify that all the pairs must be analysed (153 pairs of individuals in total).

11.2 Output

Results are grouped in a kres object, similar to the zres but only 8 slots were kept. These slots can be accessed with the @ symbol. Some slots describe the samples in the analysis:

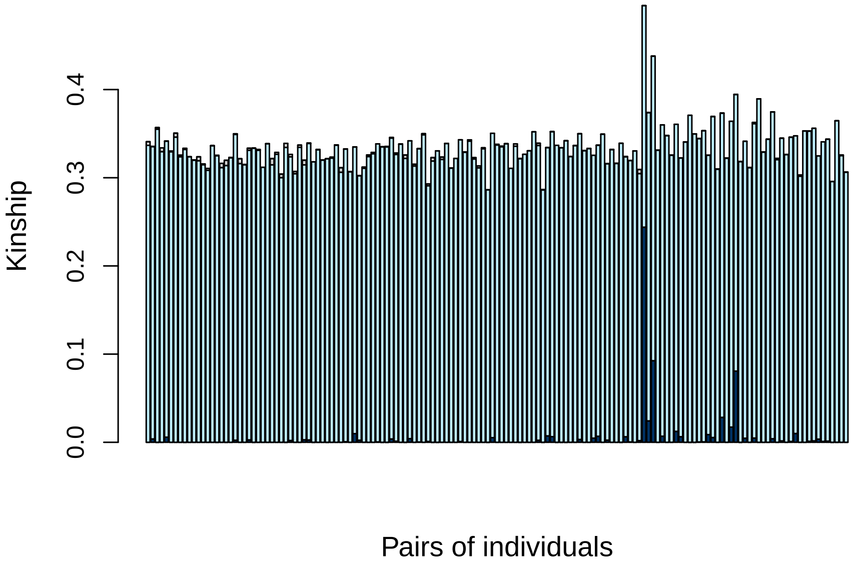
- ..@npairs is simply the number of pairs of individuals analyzed in the run;
- ..@pairs is an array with the pairs of analyzed individuals (it is basically the `kinpairs` matrix);
- ..@sampleids is an array with the name of the pairs of samples (the ids of the two individuals are combined with a "_" symbol);
- ..@haplotype_ids is a vector with the information on haplotype pairs in the full analysis and used in the `ibdseg` table (see below).

The @krates slot gives the rates of the IBD classes in the model. The results are available in the last three slots:

- ..@realized is a matrix with the genome-wide realized kinship partitioned in the different IBD classes / layers. This is the genome-wide average probability that two alleles sampled at a locus in the two individuals (one in each) are IBD with respect to that layer (the IBD segments must belong to the layer). To obtain the additive relationship, the kinship must be multiplied by two;
- ..@ibdseg is a table similar to `hbdseg` except that it contains IBD segments estimated with the Viterbi algorithm rather than HBD segments. Accordingly, the individual numbers are replaced by the number of the haplotype pairs;
- ..@ibdp is 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@hbdp[[i]].

For instance, we can use a barplot to plot the realized kinship for 153 pair of individuals in the TAF population (Gautier et al., 2024).

```
par(mar=c(5.1,4.1,2.1,2.1),mfrow=c(1,1),mfcol=c(1,1))
barplot(t(kintaf_mix4l@realized),col=c('#003366','lightblue1',"lightgrey","whitesmoke"),
        xlab="Pairs of individuals",ylab="Kinship",cex.lab=1.2)
```



For each pair of individuals, the IBD is partitioned in the four IBD classes with rates equal to {5,25,125,625}. The last class is not used while the realized kinship levels in the third class are minor. We observe that for all pairs of individuals, there is a kinship level above 0.30 associated with the second IBD class and that for three pairs of individuals, we observe additional recent kinship levels > 0.05. The background kinship level is consistent with the demographic history of this population, which experienced a bottleneck 15-20 generations ago (see Gautier et al. (2024) for more details). The partitioning into different IBD classes allows also to separate the recent kinship from the background level. The highest kinship level corresponds to a first-order relationship (parent-offspring or full-sibs), whereas the other two levels may correspond, for example, to half-sibs.

Locus-specific IBD information may help to better interpret these results. We can apply the `zookin` model with the `localhbd` option only for these three pairs as follows:

```
rel <- kintaf_mix4L@realized[,1] > 0.05
kintaf_local <- zookin(mix4l,taf3,kinpairs=mykinpairs[rel,],localhbd=TRUE)
```

Accessor functions could help to extract the information (see below).

11.3 Specific accessor function

Two accessor functions have been defined for **kres** objects obtained from a **zookin** analysis. First, the **cumkin** function is similar to **cumhbd** and estimates the kinship as the sum of all IBD classes with a $R_k \leq T$. It takes as arguments the name of the **kres** object and the value of the threshold T.

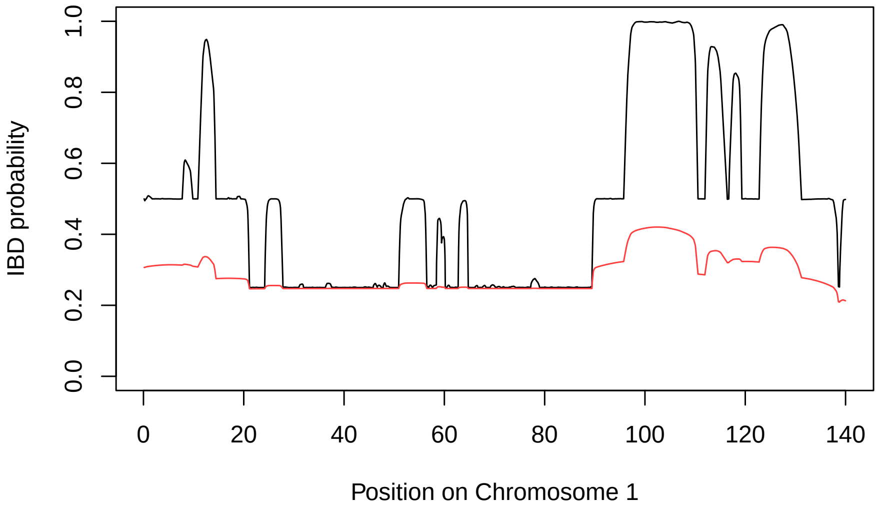
The second function, called **predhbd**, is similar to the **probhbd** function. It helps to extract the local IBD probabilities for a pair of individuals in a given region. We only extract a total IBD probability by summing over all IBD classes. As for the inbreeding coefficient, the user can specify a threshold T to determine which IBD classes should be used in the sum (all by default). This IBD probability corresponds to the probability that two alleles randomly sampled in each individual are IBD at that position, which also corresponds to the future HBD level of their future offspring at that position (see Forneris et al. (2025) for more details).

For example, we can use the accessor function to extract the IBD probabilities we estimated above for the first pair of individuals, either for all IBD classes or only for the most recent IBD class:

```
hbd1 <- predhbd(kintaf_local,zooin=taf3,chrom=1,num=1,startPos = 1,endPos = 140000000)
map1 <- taf3@bp[1:length(hbd1)]/1000000
hbd2 <- predhbd(kintaf_local,zooin=taf3,chrom=1,num=1,startPos=1,endPos=140000000,T=10)
```

And plot the results as follows:

```
plot(hbd1~map1,type='l',ylim=c(0,1),ylab="IBD probability",
     xlab="Position on Chromosome 1")
par(new=T)
plot(hbd2~map1,type='l',ylim=c(0,1),ylab="",xlab="",axes=FALSE,col="brown1")
```



We observe that when all IBD classes are included, the probability of IBD is always higher than 0.25. It takes values around 0.25, 0.50 and 1.00. This IBD probability corresponds to the probability that two alleles randomly sampled in the two individuals (one allele randomly sampled within each individual) are IBD. This probability is 0.25 and 0.50, respectively, if both individuals share one or two pairs of IBD haplotypes. The probability is 1.00 if all four haplotypes are IBD. When the analysis is restricted to recent IBD, we observe that the value remains around 0.25 along the chromosome. This suggests a parent-offspring relationship that always shares one IBD haplotype along the genome, and should not share more than one IBD haplotype if the parents are unrelated. Due to the background IBD (including all IBD classes), the IBD levels are higher than expected for such a relationship.

References

Alemu, S. W., Kadri, N. K., Harland, C., Faux, P., Charlier, C., Caballero, A., and Druet, T. (2021). An evaluation of inbreeding measures using a whole-genome sequenced cattle pedigree. *Heredity*, 126(3):410–423.

Broman, K. W. and Weber, J. L. (1999). Long homozygous chromosomal segments in reference families from the centre d'etude du polymorphisme humain. *The American Journal of Human Genetics*, 65(6):1493–1500.

Druet, T. and Gautier, M. (2017). A model-based approach to characterize individual inbreeding at both global and local genomic scales. *Molecular ecology*, 26(20):5820–5841.

Druet, T. and Gautier, M. (2022). A hidden markov model to estimate homozygous-by-descent probabilities associated with nested layers of ancestors. *Theoretical Population Biology*, 145:38–51.

Forneris, N. S., Bosse, M., Gautier, M., and Druet, T. (2025). Genomic prediction of individual inbreeding levels for the management of genetic diversity in populations with small effective size. *Molecular Ecology Resources*, page e14068.

Gautier, M., Micol, T., Camus, L., Moazami-Goudarzi, K., Naves, M., Guéret, E., Engelen, S., Lemainque, A., Colas, F., Flori, L., et al. (2024). Genomic reconstruction of the successful establishment of a feralized bovine population on the subantarctic island of amsterdam. *Molecular Biology and Evolution*, 41(7):msae121.

Harris, K., Sheehan, S., Kamm, J. A., and Song, Y. S. (2014). Decoding coalescent hidden markov models in linear time. In *Research in Computational Molecular Biology: 18th Annual International Conference, RECOMB 2014, Pittsburgh, PA, USA, April 2-5, 2014, Proceedings 18*, pages 100–114. Springer.

Lavanchy, E. and Goudet, J. (2023). Effect of reduced genomic representation on using runs of homozygosity for inbreeding characterization. *Molecular Ecology Resources*, 23(4):787–802.

Leutenegger, A.-L., Labalme, A., Génin, E., Toutain, A., Steichen, E., Clerget-Darpoux, F., and Edery, P. (2006). Using genomic inbreeding coefficient estimates for homozygosity mapping of rare recessive traits: application to taybi-linder syndrome. *The American journal of human genetics*, 79(1):62–66.

Leutenegger, A.-L., Prum, B., Génin, E., Verny, C., Lemainque, A., Clerget-Darpoux, F., and Thompson, E. A. (2003). Estimation of the inbreeding coefficient through use of genomic data. *The American Journal of Human Genetics*, 73(3):516–523.

Leutenegger, A.-L., Sahbatou, M., Gazal, S., Cann, H., and Génin, E. (2011). Consanguinity around the world: what do the genomic data of the hgdp-ceph diversity panel tell us? *European Journal of Human Genetics*, 19(5):583.

Magi, A., Tattini, L., Palombo, F., Benelli, M., Gialluisi, A., Giusti, B., Abbate, R., Seri, M., Gensini, G. F., Romeo, G., et al. (2014). H 3 m 2: detection of runs of homozygosity from whole-exome sequencing data. *Bioinformatics*, 30(20):2852–2859.

Marchini, J., Howie, B., Myers, S., McVean, G., and Donnelly, P. (2007). A new multipoint method for genome-wide association studies by imputation of genotypes. *Nature genetics*, 39(7):906.

McQuillan, R., Leutenegger, A.-L., Abdel-Rahman, R., Franklin, C. S., Pericic, M., Barac-Lauc, L., Smolej-Narancic, N., Janicijevic, B., Polasek, O., Tenesa, A., et al. (2008). Runs of homozygosity in european populations. *The American Journal of Human Genetics*, 83(3):359–372.

Narasimhan, V., Danecek, P., Scally, A., Xue, Y., Tyler-Smith, C., and Durbin, R. (2016). Bcftools/roh: a hidden markov model approach for detecting autozygosity from next-generation sequencing data. *Bioinformatics*, 32(11):1749–1751.

Pemberton, T. J., Absher, D., Feldman, M. W., Myers, R. M., Rosenberg, N. A., and Li, J. Z. (2012). Genomic patterns of homozygosity in worldwide human populations. *The American Journal of Human Genetics*, 91(2):275–292.

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., Maller, J., Sklar, P., De Bakker, P. I., Daly, M. J., et al. (2007). Plink: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*, 81(3):559–575.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Solé, M., Gori, A.-S., Faux, P., Bertrand, A., Farnir, F., Gautier, M., and Druet, T. (2017). Age-based partitioning of individual genomic inbreeding levels in belgian blue cattle. *Genetics Selection Evolution*, 49(1):92.

Thompson, E. A. (2013). Identity by descent: variation in meiosis, across genomes, and in populations. *Genetics*, 194(2):301–326.

Vieira, F. G., Albrechtsen, A., and Nielsen, R. (2016). Estimating ibd tracts from low coverage ngs data. *Bioinformatics*, 32(14):2096–2102.

Wang, S., Haynes, C., Barany, F., and Ott, J. (2009). Genome-wide autozygosity mapping in human populations. *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society*, 33(2):172–180.

Zucchini, W. and MacDonald, I. (2009). Hidden markov models for time series, volume 110 of monographs on statistics and applied probability.