hsphase – an R package for identification of recombination events, pedigree and haplotype reconstruction and imputation using SNP data from half-sib families

Mohammad H. Ferdosi and Cedric Gondro

November 26, 2018

Contents

1 Overview

hsphase comprises a suite of functions to reconstruct haplotypes using SNP data from half-sib families (a data structure widely used in livestock genomics). The package can be used to identify the paternal strand of origin (*blocks*) which the half-sibs inherited from their sire (i.e. which chromosomal regions in an individual come from either the paternal or maternal strand of the sire). These *blocks* define the recombination events that occurred in the sire to form the offspring's haplotype. *Blocks* can then used to count recombination events and detect hot and cold spot regions. The package also includes a function to phase the half-sibs using an algorithm based on opposing homozygotes and another function to impute and phase ungenotyped haplotypes of the sires. If the pedigree is not available or the pedigree is not reliable, the pedigree can be inferred from the raw genotypes. Diagnostic images can be generated to evaluate the quality of results.

Note 1: Auxiliary functions *hss*, *cs* and *para* (Sections 4.2 and 5) can be used to analyse large datasets (they are generally used in this order). *cs* and *hss* parse genotype, pedigree and map files into a format that can be used downstream by *para*.

Note 2: These functions are meant for autosomes only. The package will not phase sex chromosomes.

2 Data Input Format

hsphase uses a simple numeric matrix of animals (rows) and SNP (columns). SNP should be coded as 0, 1 and 2 for respectively AA, AB and BB. Use 9 for missing data. The matrix should have *rownames* which are the sample IDs and *colnames* which are the SNP names. The matrix must only contain individuals from one half-sib family and one chromosome and the SNP should be ordered in ascending order by base pair position. There should be at least four samples in the dataset. A large genotype file can be split into halfsib groups and chromosomes by utilising *hss* and *cs* respectively. The *cs* output can be analysed with *para*.

This matrix can be used directly with the *bmh, ssp, phf, aio* (Section 3), *rplot* (Section 6.2) and *pm* (Section 7.1) functions. Auxiliary functions to read a large genotype dataset into R, parse into family groups and chromosomes are *readGenotype* (Section 2.1), *hss* and *cs* (Section 4.2).

Note: We have found that we get better results using just the raw data with missing values than data which has previously had genotypes imputed.

Toy example of a half-sib genotype matrix:

```
genotype <- matrix(c(
  0,0,0,0,1,2,2,2,0,0,2,0,0,0,
  2,2,2,2,1,0,0,0,2,2,2,2,2,2,
  2,2,2,2,1,2,2,2,0,0,2,2,2,2,
  2,2,2,2,0,0,0,0,2,2,2,2,2,2,
  0,0,0,0,0,2,2,2,2,2,2,0,0,0
```

```
), ncol = 14, byrow = TRUE)

AnimalID <- paste("ID-", 1:5, sep="")
SNPID <- paste("SNP-", letters[1:14], sep="")

rownames(genotype) <- AnimalID
colnames(genotype) <- SNPID
genotype[1:5, 1:5]
```

2.1 Reading the Genotype File

The *readGenotype* function reads and performs a *sanity* check on a genotype file. The input is the name of the genotype file.

```
readGenotype(genotypePath, separatorGenotype = " ", check = TRUE)
```

If the *check* option is set to *TRUE*, the genotype file will be checked for possible errors.

Note 1: The SNP and the animals' IDs should not contain a double quotes.

Note 2: This function uses the *scan* function in R to improve read speeds for large genotype files.

3 Main Functions: Block Partitioning, Sire Imputation and Phasing of a Half-Sib Family

3.1 Block Partitioning (bmh)

The *bmh* function creates the *block* structure for the half-sibs. The result is a matrix (one row for each half-sib and one column for each SNP in the same order as the input matrix) that shows which of the sire's haplotype each half-sib inherited. Paternal strands are arbitrarily coded as 1 and 2; i.e. individuals that have the same numbers (e.g. 1, 1) for a given SNP, inherited the same haplotype from the sire. Alternatively, if two half-sibs have e.g. a 1 and 2, they inherited different haplotypes from the sire. 0 is used when the paternal strand of origin could not be determined.

```
recombinationBlocks <- bmh(genotype)
```

3.2 Sire Imputation and Phasing (ssp)

The *ssp* function infers (imputes) and phases the sire's genotype. It requires the half-sib's original genotype matrix and the block structure generated with *bmh*. The function returns a matrix with two rows, one for each haplotype of the sire (columns are SNP in the order of the genotype matrix). Alleles are coded as 0 (A) and 1 (B). Alleles that could not be imputed are coded as 9.

Note: Across chromosomes there is no relationship between the sire's first and

second haplotype (i.e. the paternal/maternal haplotypes of the sire can be swapped between chromosomes).

```
sireHaplotype <- ssp(bmh(genotype), genotype)
```

3.3 Half-Sib Family Phasing (phf)

The *phf* function phases the half-sib data. It uses as input the half-sib's genotype matrix, block structure (generated with *bmh*) and the matrix of imputed sire (generated with *ssp*). The output is a matrix that contains the phased haplotype each half-sib inherited from the sire. The resulting matrix has the same number of rows as the input genotype matrix. It uses 0, 1 and 9 for A, B and missing. The maternal haplotype (haplotype offspring inherited from the dam) can be created by subtracting the genotype matrix from this matrix. The function *aio* (below) can be used to generate both haplotypes in a single matrix.

```
familyPaternalHaplotypes <- phf(genotype, bmh(genotype),
                      ssp(recombinationBlocks, genotype))
```

3.4 All-in-one Phasing (aio)

The *aio* function uses the previous functions to generate the half-sib's haplotypes (from both the sire and dam). The output is a matrix containing 0 and 1 for alleles A and B (with 9 for missing/unknown phase). The IDs of each animal (*rownames*) are duplicated with p and m suffixes which stand for paternal (inherited from the sire) and maternal (from the dam) haplotypes.

```
familyPhased <- aio(genotype)
```

4 Auxiliary Functions

The objective of these functions is to assist users to convert the data to the format used by *hsphase*. Essentially, these functions will simply split a matrix of genotypes into a list of chromosomes and half-sib families.

4.1 Input Files

Three input files are required to use the *hss* and *cs* functions:

4.1.1 Genotype File

A flat file with genotypes (ID $\times$ SNP), where the SNP names are in the first row and the ID of each individual in the first column (ID should not have any header). The delimiter can be specified via the *readGenotype* function (Section 2.1). The IDs and SNP names should be unique. SNP are denoted by 0, 1, 2 and 9 for AA, AB, BB and missing respectively.

	SNP1	SNP2	SNP3	SNP4
IND1	0	2	1	1
IND2	1	9	0	2
IND3	2	0	1	0

4.1.2 Pedigree File

The pedigree file should contain at least two columns, one for the half-sibs and one for the sires, in this order. Other columns are ignored. Unknown sires can be specified by *0* (but results will simply be a string of *9s*) and offspring IDs should be unique. This file must *not* have a header.

4.1.3 Map File

The map file must contain a column for each of: SNP name, chromosome and their position in base pairs. The first line must contain a header with *"Name Chr Position"*. Additional columns are ignored. Space is the default separator for all files.

4.2 Functions

4.2.1 Half-sib Family Separator (hss)

The *hss* function generates a list of matrices (ID × SNP), one for each of the half-sib groups that have at least 4 offspring per sire. The input are the genotype and pedigree files (path to the file, matrix or data.frame). The names of the list are the names of the sires in the pedigree file.

```
halfsib <- hss(pedigree, genotype)
```

4.2.2 Chromosome Separator (cs)

After splitting the data into family groups, the *cs* function can be used to separate the *hss* data into the different chromosomes based on a map file. The output is also a list of matrices (ID × SNP, one for each family and chromosome). The *names* in the R output list are the half-sib groups name (sire ID) and their chromosome numbers separated by an underscore (_). Subsets can be found using e.g. *grep* (Section 10). Data in the correct format can also be built manually; it's simply a split list of numeric matrices (ID × SNP with 0, 1, 2 and 9), with SNP ordered by BP position. One matrix for each chromosome and sire.

```
halfsib <- cs(halfsib, mapPath, mapSeparator)
```

5 Parallel Data Analysis (para)

The *para* function uses the list of matrices (the output of *cs*) and runs one of the options below, on each element of the list, in parallel. This function requires the *snowfall* package. To run use e.g.:

```
blocks <- para(halfsib, cpus = 20, option = "bmh", type = "SOCK")
sireImpute <- para(halfsib, cpus = 20, option = "ssp", type = "SOCK")
phasedSibs <- para(halfsib, cpus = 20, option = "aio", type = "SOCK")
```

cpus sets the number of CPUs to use, *option* sets the type of analysis and can be any of the above described *bmh*, *ssp*, *aio* or *pm* (7.1) and also *rec*. The *rec* option returns a list with the sum of recombinations between SNP for each chromosome in each half-sib group. The parameter *type* sets the type of cluster for parallel analysis (for more information refer to the snowfall documentation). The output is a list of the same length as the input data and its contents depends on the *option* selected (Section 3). *rec* is just a wrapper function which is equivalent to:

```
result <- pm(bmh(genotype))
result <- apply(result, 2, sum)
```

The *result* can be plotted to visualise the recombinations.

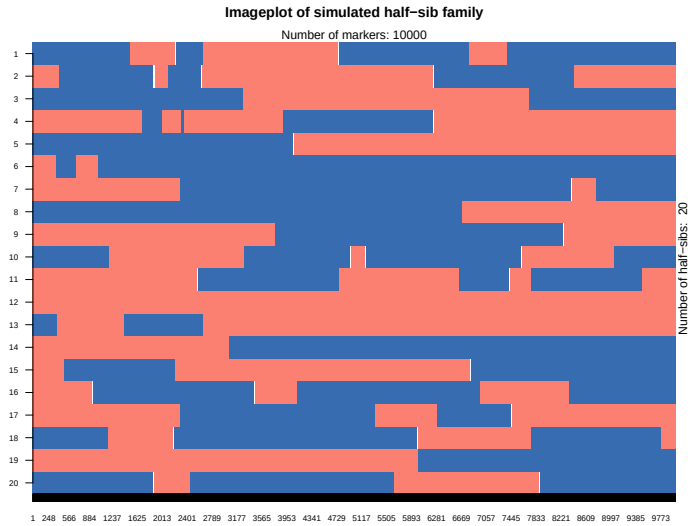
6 Visualisation

6.1 Blocking Structure Plot (imageplot)

The *imageplot* function creates a plot of the blocking structure created by either *bmh* or *hbp* (Sections 3.1 and 7.2). White indicates regions of unknown origin, red and blue correspond the two sire strands.

Note: across chromosomes the colours are not comparable – i.e. blue in chromosomes 1 and 2 may not relate to the same strand of origin in the sire (paternal/maternal).

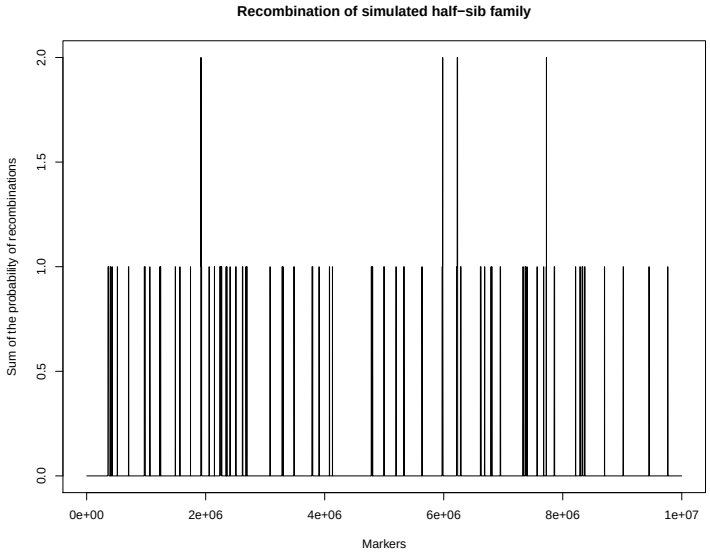
```
imageplot(bmh(genotype))
imageplot(blocks[[1]]) # from para results
```



6.2 Recombination Plot (rplot)

This function creates a plot which shows the sum of all recombination events across the half-sib family. It uses the half-sib genotypes and needs a vector of SNP positions for each SNP on the chromosome.

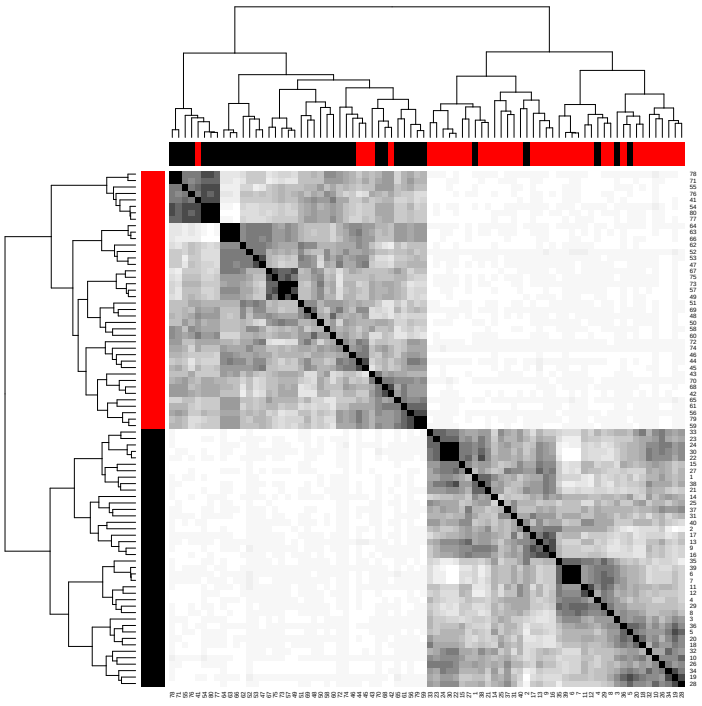
```
rplot(genotype, distance)
```



6.3 Heatmap of Half-sib Families (hh)

The *hh* function generates a heatmap of the half-sib families using the matrix of opposing homozygotes (8).

```
hh(ohg(genotype), inferredPedigree, realPedigree)
```

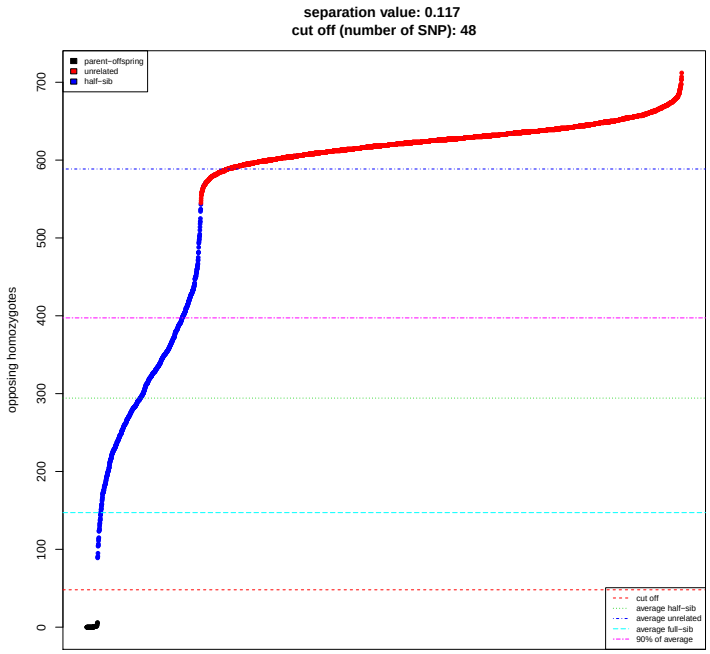


The heatmap assists identification of pedigree errors and can help to check if the pedigree reconstruction results seem correct. The *inferredPedigree* (Section 8.2) and *realPedigree* are used by the *hh* function to create, respectively, the *RowSideColors* and *ColSideColors*. In practice either can be freely interchanged to colour code the side bars of the heatmap. A maximum of 21 colours are used by the heatmap. If there are more half-sib families, colours will be repeated.

6.4 Plot of Opposing Homozygotes (ohplot)

The *ohplot* function plot the vectorized and sorted opposing homozygote matrix.

```
ohplot(ohg(genotype), genotype, pedigree, check = TRUE)
```



7 Diagnostic Tools

7.1 Probability Matrix (pm)

The *pm* function utilises the block structure matrix (from *bmh*) to find recombination sites. It returns a matrix of 0s and 1s. 1 indicates that a recombination occurred between two consecutive SNP or 0 for no recombination. If the exact position of the recombination cannot be determined the region of uncertainty is filled with 1s.

```
genotype <- matrix(c(
0,2,0,1,0,
2,0,1,2,2,
2,2,1,0,2,
2,2,1,1,1,
0,0,2,1,0), ncol = 5 ,byrow = T)

(result <- bmh(genotype))
pm(result)
```

This function can be useful to identify mapping errors. For example, if (very) large recombination rates are observed at a particular SNP across families.

7.2 Haplotype Block of Phased Data (hbp)

The *hbp* function creates a block structure of the half-sib family based on phased data from the sire and its half-sib family. It requires a haplotype matrix from the sire (*ssp*) and one from the half-sib family (*aio*). This function can be used as a diagnostic tool to evaluate the result of other phasing algorithms (provided there are parents – offspring in the data).

```
sire <- matrix(c(
  0,0,0,0,0,1,                  # Haplotype one of the sire
  0,1,1,1,1,0                   # Haplotype two of the sire
  ), byrow = T, ncol = 6)

haplotypeHalfsib <- matrix(c(
  1,0,1,1,1,1,                  # Individual one, haplotype one
  0,1,0,0,0,0,                  # Individual one, haplotype two
  0,1,1,0,1,1,                  # Individual two, haplotype one
  1,0,0,1,0,0                   # Individual two, haplotype two
  ), byrow = T, ncol = 6)       # 0s and 1s are allele A and B

hbp(haplotypeHalfsib, sire)
```

Note: The results can be plotted with *imageplot*.

7.3 Identification of Recombination Events (recombinations)

The *recombinations* function counts the number of recombinations for each individual in a half-sib family.

```
genotype <- matrix(c(
  2,1,0,0,
  2,0,2,2,
  0,0,2,2,
  0,2,0,0
  ), byrow = TRUE, ncol = 4)

recombinations(bmh(genotype))
```

Note: This function can be used to detect pedigree errors and is robust if there are at least 10 half-sibs in the family. Individuals that show a disproportionate number of recombinations do not belong to that family group.

8 Pedigree Reconstruction

8.1 Matrix of Opposing Homozygotes (ohg)

The *ohg* function creates a matrix of the number of opposing homozygotes between all pairs of individuals. The result is square matrix where the rownames and colnames are the IDs of individuals.

```
ohg(genotype, cpus = 2)
```

Note: This function utilises OpenMP in GNU/Linux and the number of *cpus* is only valid in GNU/Linux.

8.2 Pedigree Reconstruction of Half-sib Families (rpoh)

The *rpoh* function reconstructs half-sib families; i.e. splits the individuals into half-sib groups.

Four methods *simple*, *recombinations*, *calus* and *manual* can be utilised to reconstruct the pedigree. For more details please refer to – article.

```
pedigree1 <- rpoh(oh = oh, snpnooh = 732, method = "simple")
pedigree2 <- rpoh(genotypeMatrix = genotypeChr1, oh = ohg(genotype),
maxRec = 10 , method = "recombinations")
pedigree3 <- rpoh(genotypeMatrix = genotype, oh = oh, method = "calus")
pedigree4 <- rpoh(oh = oh, maxsnpnooh = 31662, method = "manual")
```

Note 1: The functions *ohg* and *rpoh* with *recombinations* method can be slow with very large datasets. The genotype matrix with only one chromosome is usually sufficient to separate the individuals into half-sib groups and can speed up the process.

Note 2: The first argument of *rpoh* function (genotypeMatrix) for *recombinations* method must use only genotypic data from a single chromosome.

Note 3: The *snpnooh* is the number of SNPs (divided by 1000) used to create opposing homozygote matrix.

Note 4: The *maxsnpnooh* is the maximum number of allowing opposing homozygote in a family.

8.3 Fix Pedigree Errors (pedigreeNaming)

The *pedigreeNaming* function tries to link the inferred half-sib family groups to the sire IDs in the original pedigree and fix errors. This works well if the original pedigree is relatively correct but individuals will be misassigned if most of the individuals originally allocated to a sire are not its offspring.

8.4 Parentage Assignment (pogc)

The *pogc* function utilises the opposing homozygote matrix and return pedigree of parent-offspring assignments.

```
pedigree <- pogc(oh, genotypeError)
```

The genotypeError argument is the maximum number of mismatches allowed.

9 Imputation

The *impute* function impute the low density SNP marker to high density marker utilising the high density haplotype of sire.

```
impute(halfsib_genotype_ld, sire_hd)
```

The *halfsib_genotype_ld* is the genotype of half–sibs with low density markers. The *sire_hd* is the haplotype of sire that can be either phased sire haplotype or the haplotype of sire assembled from sequence data.

Note: The *halfsib_genotype_ld* and *sire_hd* must have *colnames* which are the SNP names.

10 Quick Guide

10.1 Half-sib Family Analysis

The *aio, ssp and bmh* functions phase a half-sib family, impute the sire and create the block structure respectively. The half-sibs must be a numeric matrix with animal IDs as rownames and SNP IDs as colnames. If the genotype file contains only one half-sib family, it can be read with the *readGenotype* function (Section 2.1).

```
genotype <- readGenotype("path to the genotype file", separator) # Section 2.1
recombinationBlocks <- bmh(genotype)                             # Section 3.1
sireHaplotype <- ssp(bmh(genotype), genotype)                    # Section 3.2
familyPhased <- aio(genotype)                                    # Section 3.4
```

10.2 Multi-family Analysis

The input genotype file must have the same structure as the half-sib genotype file discussed above (Section 2).

```
# read in the genotype file (Section 2.1)
genotype <-
readGenotype("path and name of the genotype file", separator)

# hss generates a list of half-
sibs based on a pedigree file (Section 4.2.1)
halfsib <- hss(pedigree, genotype)

# splits the output from hss into the various chromosomes (Sec-
tion 4.2.2)
halfsib <- cs(halfsib, mapPath)

# Block Partitioning (Section 5 and 3.1)
recombinationBlocksList <-
para(halfsib, cpus = 20, option = "bmh", type =
```

```
"SOCK")

# Sire Imputation (Section 5 and 3.2)
sireHaplotypeList <-
para(halfsib, cpus = 20, option = "ssp", type = "SOCK")

# Half-Sib Family Phasing  (Section 5 and 3.4)
familyPhasedList <-
para(halfsib, cpus = 20, option = "aio", type = "SOCK")
```

Results can be concatenated by using a simple function such as:

```
chromosomeMatch <- function(listHalfsibs, numberChr)
{
    chr <- list()
    for(i in 1:numberChr)
    {
        chr[[i]] <- listHalfsibs[grep(paste("_", i, "$", sep = ""), names(listHalfsibs))]
        chr[[i]] <- do.call(rbind, chr[[i]])
    }

    phasedGenotype <- do.call(cbind, chr)
    phasedGenotype
}
```

The *numberChr* is the number of chromosomes.

Note: A comprehensive demo and example dataset is available from Cedric Gondro's home page or running *demo(hsphase)*.

11 How to Cite the hsphase

To cite the package please type

```
citation("hsphase")
```