PediHaplotyper Manual

Roeland Voorrips, Wageningen UR – Plant Breeding, 2015

Introduction

PediHaplotyper is software for assigning haploblock alleles to individuals in a pedigree, based on observed marker alleles that have already been phased by other software. A haploblock is defined as a group of very closely linked markers among which there are no recombinations over the pedigree (except perhaps in the founder or final generations).

There are several advantages to working with haploblock alleles, rather than directly with phased marker alleles, especially given the currently available high-density SNP data. For the human investigator it is much easier to follow inheritance patterns based on a few, multi-allelelic haploblocks than on a large number of bi-allelic SNPs. A better understanding helps to spot possible inconsistencies in inheritance patterns, which may point to errors in samples, in pedigree relationships or in marker data or marker phasing. It may also help to discover or validate unknown or suspected parentage of individuals. For QTL mapping, the compact form of the haploblock alleles and their higher information content reduces memory requirements and increases computation speed, and may also increase the power of QTL detection.

PediHaplotyper is a package written in R (https://cran.r-project.org/) and will run on any system where R is available, including Windows, Linux and Mac OS X. In the following sections we will consider the required input data, the commands needed to run PediHaplotyper and the output files, and we will present a small example.

Citation

Voorrips RE, Bink MCAM, Kruisselbrink JW, Koehorst–van Putten HJJ, van de Weg WE. PediHaplotyper: Software for consistent assignment of marker haplotypes in pedigrees (submitted to Molecular Breeding)

Availability

Contents

Input data

The data that PediHaplotyper needs consist of a pedigree, a marker map, and phased marker data. These data can be supplied in tab-separated text files, the generic data format. Because PediHaplotyper was developed in a situation where the phasing of the marker data was done using FlexQTL™ (Bink et al., 2014) (https://www.wageningenur.nl/en/show/FlexQTL.htm) and the results produced by PediHaplotyper were again analysed using FlexQTL, we provide also specific import and export formats suitable for use with FlexQTL. If the input data are supplied in FlexQTL format, a FlexQTL parameter file (by default with name flexqtl.par) must be present.

Marker map

The generic map format consists of a tab-separated text file with at least four columns with captions *Marker*, *Chromosome*, *Position* and *Haploblock*. Other columns may be present as well but will be ignored. Markers, chromosomes and haploblocks may have names or numbers. The map must be sorted: the order of the chromosomes is not relevant, but within each chromosome the markers must be ordered by increasing position. The positions must be numbers; the units (e.g. Morgan, centiMorgans, bases or megabases) are not relevant. Haploblocks must consist of a contiguous series of markers, all of which must be on the same chromosome.

A different map file format is also allowed. In this case each chromosome starts with a header line consisting of the word "group" followed by the chromosome name. Below this header are lines, each consisting of a marker name, a position and a haploblock name, separated by whitespace (tabs and/or spaces). Empty lines are allowed and ignored, and lines starting with a ';' are also ignored. This format is similar to that used by other mapping software including FlexQTL, JoinMap®, MapQTL® and MapChart, except that on each marker line, after the position, a haploblock name is required. Again the map must be sorted in ascending order per chromosome, and the same restrictions on haploblocks apply.

In both map formats it is possible to exclude markers from the analysis by adding a ';' as first character on the line. In the second format, if all markers of a chromosome are excluded in this way the header line of the chromosome must also be preceded by a ';'.

PediHaplotyper checks automatically which of the two map formats is used and generates an error when the format is not recognized.
In the case of FlexQTL input the map may also be read from the phased genotypes file, see below.

Pedigree file

The generic pedigree file format consists of a tab-separated text file with at least three columns with captions *Name*, *Parent1* and *Parent2*, in that order as the first 3 columns in the file. Additional columns, e.g. with phenotypic trait data are allowed; these are read but not used by PediHaplotyper and will be exported along with the sorted pedigree (see Output data below).

The pedigree may be given in any order; it is checked for consistency (e.g. no individual may be its own ancestor, all individuals must have distinct names, and all parents must also occur as individuals). A founder is specified as an individual with two empty cells as parents. If necessary the pedigree will be sorted by PediHaplotyper such that parents occur before all their progeny.

In the case of FlexQTL input the pedigree is read from the FlexQTL input data file, or a similar file written by FlexQTL with the pedigree sorted. This is a text file where the cells are separated by whitespace, without column captions. The first three columns contain the pedigree as described above, except that in the case of founder individuals the missing parents are indicated by a '0' (zero) or '–' (dash). This file may also contain 'nuisance' and 'trait' columns, which are specified in the FlexQTL parameter file; these are read and written in the output as the phenotypic trait data described above. After this the unphased marker data appear; these are ignored by PediHaplotyper. The default filename of this file is flexqtl.sort; if that is not present the input file specified in the FlexQTL parameter file (by default flexqtl.par) is used. For further information on this file format see the FlexQTL documentation.

Phased genotypes file

The generic phased genotypes file format consists of a tab-separated text file with the individuals in the first column, followed by a pair of columns for each marker containing the alleles inherited from Parent1 and Parent2 in that order. The caption of the first column with individuals is ignored; the caption of the first column for each marker is taken as the marker name, and the caption of the second column for each marker is ignored. The marker alleles can be any name and/or number, and any number of different allele names per marker is allowed (to accommodate e.g. bi-allelic SNPs as well as multi-allelic SSRs). Missing data are represented as empty cells.

All markers in the map, and all individuals in the pedigree must also occur in the phased haplotypes file, but they do not need to appear in the same order. There may be extra markers and/or individuals in the phased haplotypes file but these will be ignored.

In the case of FlexQTL input the phased haplotypes are read from a file with default name mhaplotypes.csv; this is a text file in comma-separated format. We will not discuss this format in detail here but only mention that the phased marker data for each individual are read from the lines with 'hap1' and 'hap2' after the individual's name. It is possible to read the marker map from this file, from the lines starting with 'mchr', 'mpos' and 'ID'. Since this map does not specify the haploblocks, all markers with the same position after rounding down to integer numbers are taken to belong to the same haploblock.

Some other software may produce phased genotypes files with the data arranged in a different way, e.g. with two rows per individual and one column per marker, and/or with the markers in the rows and the individuals in the columns. For easy conversion of files or data frames to the generic format read by PediHaplotyper three functions are provided: transpose (exchange the rows and columns), and twocols2tworows and tworows2twocols which manipulate the data as their names suggest.

Old haploblock alleles file

This file is optional. If the marker scores of some individuals have changed, or if individuals have been added or deleted or the pedigree has changed in other ways and PediHaplotyper is run again, it is convenient to have the same allele numbers assigned to the same haploblock alleles as in the earlier run. The Old haploblock alleles file provides the information needed to do this. It is a tab-separated text file with for each haploblock a header line which ends with the names of the markers in that haploblock, followed by one line for each allele of that haploblock with some statistics and the marker alleles in that haploblock allele (some or all of which may be missing values, represented by '-'). This file is one of the files produced by PediHaplotyper (the hballelesfile, see under Output). Note that this file cannot be read using the R function read.table, as the lines for different haploblocks will be different in length.

Operation

Before the functions of PediHaplotyper can be used, the package must first be installed. Download the package file (PediHaplotyper_1.0.tar.gz) from its website (https://www.wageningenur.nl/en/show/PediHaplotyper.htm) and open an R window (or RStudio). Type the

```
install.packages("D:/PediHaplotyper_1.0.tar.gz", repos = NULL, type = "source")
```

(assuming that the file name is PediHaplotyper_1.0.tar.gz and is located in D:\ ). In RStudio the installation can also be performed using the menu: in the Packages menu select Install, Install from Package Archive File and select the file.
The installation needs to be done only once; after that the package will remain available in the computer.

In order to use PediHaplotyper, first copy all input files into one directory. Then open an R window (or RStudio) and type the commands

```
setwd("d:/data")
library(PediHaplotyper)
```

Here and in all other cases where a path, filename or other literal text string is specified, it must be enclosed in straight "quotes" and not the pretty "inverted quotes" that Word likes to insert. The first command sets the working directory of R to d:\Data; any other directory can of course also be used. Note that even under Windows the backslash "\" must be replaced by a forward slash "/" ! The second command makes the functions in PediHaplotyper available in the current R session.

PediHaplotyper provides two functions that perform the haploblock allele assignment, one for use with generic tab-separated text files and one for use with FlexQTL data files. These two functions are (relatively) user-friendly wrappers that perform the complete process from reading the input to producing the output, calling a series of functions to obtain the result. Experienced R users that need more flexibility can have a look at the (relatively well-documented) source code and follow their own approach; the source code is available within the tar.gz file.

Operation with generic input files

The function call to perform the haploblock allele assignment is:

```
haplotyping_session(sessionID="A", mapfile="my_map.dat", pedigreefile="my_pedigree.dat",
phasedgenofile="my_phased_geno.dat")
```

Apart from the mapfile, pedigree file and phased genotypes file also a 'session ID' is specified. The session ID is a short string that is prefixed to all output file names and serves to distinguish the files from different runs (e.g. with different parameters of input files) from each other.

Operation with FlexQTL input files

If the input data were generated by FlexQTL we use a different function to perform the haploblock allele assignment: fq_haplotyping_session instead of haplotyping_session (note the prefix fq_). By default the input files are assumed to have the standard names given by FlexQTL and they don't need to be specified. The function call can therefore be rather simple:

```
fq_haplotyping_session(sessionID="A", mapfile="my_map.dat")
```

As above, the sessionID is a string that is prefixed to all output file names. The assumed input file names are fqparfile="flexqtl.par" (the FlexQTL parameter file), pedigreefile="flexqtl.sort", phasedgenofile="mhaplotypes.csv". If flexqtl.sort is not found, PediHaplotyper looks in the FlexQTL parameter file to find the name of the original FlexQTL data file.

If the input files do not have the standard FlexQTL names they need to be specified:

```
fq_haplotyping_session(sessionID="A", mapfile="my_map.dat", fqparfile="my_flexqtl.par",
pedigreefile="my_flexqtl.sort", phasedgenofile="my_mhaplotypes.csv")
```

It is possible (but not recommended) to read the map from FlexQTL's mhaplotypes.csv file. In that case the haploblocks are defined as consisting of all markers that have the same position after rounding down. In this case the map file is not specified:

```
fq_haplotyping_session(sessionID="A")
```

By default, if the input files are in FlexQTL format, the output files with the phased marker genotypes and phased haploblock genotypes will also be in FlexQTL format, and the output will include matching map and FlexQTL parameter files. If the output must be in generic tab-separated file format a parameter FQout=FALSE must be added, e.g.:

```
fq_haplotyping_session(sessionID="A", mapfile="my_map.dat", FQout=FALSE)
```

Using the haploblock allele numbers from an earlier session

In order to use the haploblock allele numbers from an earlier run (so that it is easier to compare the results of both runs) also parameter oldhballeles must be supplied, e.g.:

```
haplotyping_session(sessionID="B", mapfile="my_map.dat", pedigreefile="my_pedigree.dat",
phasedgenofile="my_phased_geno.dat", oldhballeles="A_hballeles.dat")
```

where the earlier run had sessionID "A" and the new run has sessionID "B". This works for the generic (haplotyping_session) and FlexQTL input (fq_haplotyping_session) versions.

Changing the haploblock allele name format

In the output files the haploblock alleles have numbers as names, i.e. each allele has a different number. It is possible to add the number of missing marker scores in the haploblock allele to the allele name: in that case, names are formed like 3(2) and 5(0), meaning allele nr 3 (which has 2 missing marker scores) or allele nr 5 (with no missing marker scores). In order to do so, a parameter mv.count=TRUE must be added to function call:

```
haplotyping_session(sessionID="A", mapfile="my_map.dat", pedigreefile="my_pedigree.dat",
phasedgenofile="my_phased_geno.dat", mv.count=TRUE)
```

The same parameter can also be added to the FlextQTL input version of the function call.

Suppressing output files

It is possible to suppress specific output files by setting their filenames to "" in the function call. For example, to suppress the mrkpolymorphism.dat file (discussed in the Output section) the function call can be extended to

```
haplotyping_session(sessionID="A", mapfile="my_map.dat", pedigreefile="my_pedigree.dat",
phasedgenofile="my_phased_geno.dat", mrkpolymorphismfile="")
```

Again, this works for the generic (haplotyping_session) and FlexQTL input (fq_haplotyping_session) versions. In the Output section all possible output files are discussed, and parameter settings to suppress each file are also given.

Output

The names of all output files start with the session ID given in the haplotyping_session function call. Many output files are specific for marker data or haploblock data; this is indicated by 'mrk' or 'hb' in the file name. Similarly some filenames contain "orig" or "final"; these files have information on the data as they were before and after the haploblock allele assignment, respectively.

We discuss the various output files using a small example. All input and output files of the example are provided in the file PediHaplotyper_Example.zip. This zip file contains a subdirectory input (containing only the input files) and a subdirectory output (with the input and output files as present after running the example, and some Pedimap project files with visualisations of the results). First we give some instructions on how to reproduce the Example output files.

Example

Input and output files are provided both in generic format and in FlexQTL format. To run the examples, first create a new folder; we assume this folder has the name D:\Example. Next copy the input files to this folder:

Three files in generic format:

- pedigree.dat,
- map.dat,
- phasedgenotypes_2rows.dat,

Four files in FlexQTL format:

- flexqtl.par,
- flexqtl.sort,
- flexQTL.map,
- mhaplotypes.csv.

Next start R (or RStudio), set the working directory to the folder we just created, and make the PediHaplotyper functions available:

```
setwd("d:/Example")
library(PediHaplotyper)
```

Generic file input

In this example we assume that the marker phasing has been done with software that produces a phased data file with two rows per individual and one column per marker, while PediHaplotyper requires one row per individual and two columns per marker. The first thing to do is therefore to convert the phased genotypes file to the correct format. PediHaplotyper offers some functions to carry out these transformations and we use the appropriate one her:

```
tworows2twocols(source="phasedgenotypes_2rows.dat", target="phasedgenotypes.dat", sep="\t")
```

Next we run the haplotyping_session function:

```
haplotyping_session(sessionID="A", mapfile="map.dat", pedigreefile="pedigree.dat",
phasedgenofile="phasedgenotypes.dat")
```

FlexQTL input

Most input files have already their default names as given by FlexQTL except the map file, which has been manually extended to include a haploblock for each marker. The call to fq_haplotyping_function can therefore be very simple:

```
fq_haplotyping_session(sessionID="B", mapfile="flexQTL.map")
```

Output file description

All output files from the generic input have a name prefixed with "A_", the output files from the FlexQTL input have prefix "B_"; in the description below the prefix is omitted.

Files produced for both generic and FlexQTL output

messages.txt

File messages.txt may contain some messages about discrepancies between the pedigree and the phased genotypes file, and how these are handled. Also it gives the convergence result for each haplotype. These messages also appear on the screen during the session.

```
haplotyping session started on Fri Oct 16 14:17:36 2015


hb 1 = HB1: 10 initial alleles
hb 2 = HB2: 12 initial alleles

hb 1 = HB1: convergence = yes in 3 cycles
hb 2 = HB2: convergence = yes in 2 cycles
```

mrkpolymorphism.dat

File mrkpolymorphism.dat gives some data on each marker. Among this information is whether the marker is used or discarded by PediHaplotyper (shown as a 1 or 0 in column *usemarker*): a marker is used if at least two alleles occur each at least min.allele.freq times (by default 3 times). So the default setting is a little bit more stringent than just requiring that a marker is not monomorphic.
Output of this file can be suppressed by adding mrkpolymorphismfile="" to the (fq_)haplotyping_session function call.

marker	chrom	pos	haploblock	usemarker	count_NA	count_1	count_2	allele_1	allele_2
M01	1	32	HB1	1	6	46	14	A	B
M02	1	32.01	HB1	1	2	51	13	A	B
...									
M09	1	44.04	HB2	1	4	11	51	A	B
M10	1	44.05	HB2	0	0	66	0	B	
M11	1	47.04	HB2	1	2	20	44	A	B
...									

In this example marker M10 is not used (usemarker=0) because it is monomorphic. Apart from the chromosome, position and the *usemarker* columns this file shows the frequency of all alleles in the input data. Columns count_NA, count_1 and count_2 are the number of missing scores, scores of the first allele and scores of the second allele, and allele_1 and allele_2 are the names of those alleles. If there were markers with more alleles the number of columns for count_x and allele_x would be increased.

orig_hballeles_byHS.dat and final_hballeles_byHS

These are two files that show the haploblock alleles per Half-Sib (HS) family at the start (orig_hballeles_byHS.dat) and end (final_hballeles_byHS.dat) of the haplotyping process. Output of these files can be suppressed by adding HSorighballelesfile="" and/or HSfinalhballelesfile="" to the (fq_)haplotyping_session function call.

HSfam	famsize	parent	haploblock	parhap1	parhap2	hballeleID	inparent	freq	markeralleles					
1	11	Golden_Delicious	HB1	2	3	2	1	4	1	1	1	1	2	1
1	11	Golden_Delicious	HB1	2	3	3	1	4	2	1	2	2	2	1
1	11	Golden_Delicious	HB1	2	3	6	0	1	-	1	-	2	2	1
1	11	Golden_Delicious	HB1	2	3	7	0	1	-	1	2	2	2	1
1	11	Golden_Delicious	HB1	2	3	9	0	1	2	1	-	-	2	1

...

The HS families are sorted by decreasing size, so the largest HS family is the first one. This family has 11 members and its common parent is Golden Delicious, which can be the mother of some members and the father of others (and in principle it could be both mother and father if some members of the HS family, in the case of a selfing). Here are shown only the lines for haploblock HB1. The two parental alleles are 2 and 3, both alleles have no missing marker data. In this HS family 5 different alleles occur for this haploblock (column *hballeleID*), only two of which occur in the common parent (column *inparent*). The frequency of each allele is shown: the three alleles that do not occur in the parent each have a frequency of 1. Finally the marker alleles of each haploblock are shown, where 1 and 2 refer to the allele numbering in the mrkpolymorphism file. Different haploblocks may have different numbers of markers, so the lengths of the lines in this file look different for different haploblocks (although actually the final columns are always present but empty in lines shorter than the maximum). The alleles 6, 7 and 9 all have missing marker values; the first two only match with parental allele 3 but 9 matches with both parental alleles.

hballeles.dat

File **hballeles.dat** lists all alleles per haploblock that are present in the pedigree at the end of the process or at some intermediate stages. It is needed to translate the haploblock allele numbers to sequences of marker alleles. It can also be used as input to a new run to ensure that all haplotypes will keep their earlier ID. This requires the parameter 'oldhballeles=' in the function call (see the Operation section).
Output of this file can be suppressed by adding hballelesfile="" to the (fq_)haplotyping_session function call.

hbnr	haploblock	markercount	hballeleID	hballelenr	nacount	freq	M01	M02	M03	M04	M05
1	HB1	5	1	1	5	2	-	-	-	-	-
1	HB1	5	2	2	0	9	A	A	A	A	B
1	HB1	5	3	3	0	17	B	A	B	B	B
1	HB1	5	4	4	0	24	A	A	A	B	B
1	HB1	5	5	5	0	13	A	B	A	B	A
1	HB1	5	6	6	2	0	-	A	-	B	B
1	HB1	5	7	7	1	0	-	A	B	B	B
1	HB1	5	8	8	0	1	A	A	A	A	A
1	HB1	5	9	9	2	0	B	A	-	-	B
1	HB1	5	10	10	2	0	A	A	-	-	B

...

Allele 1 consisting of only missing marker scores is always present. The first 3 columns give information about the haploblock and are identical for all alleles. The haploblock allele ID consists by default of the allele number only (as here) but with parameter mv.count-TRUE the number of missing marker scores in the allele can be added to its ID (see Operation section). This haploblock has 5 alleles with no missing marker data, while the alleles with missing marker data all match with at least one of those 5 (that is not always the case).

hbstatistics.dat
This file shows how the final and original haploblock scores compare to each other. For each haploblock and each individual a code is listed that indicates the transformation; these codes are documented in Table 1.

Table 1. Result codes in the statistics and Pedimap files

0	Score unchanged
1	Missing score unchanged
2	Missing score imputed (initially unscored, allele assigned by PediHaplotyper)
3	Original score changed (original score replaced by another score)
4	Original score removed (original score replaced by a missing value)
7	Marker not analyzed (almost monomorphic; does not apply to haploblocks)
8	No convergence: cyclical
9	No convergence: 50 iterations reached

Codes 0 to 4 apply to (markers in) haploblocks where the process terminated correctly. Code 7 (marker not analysed) is shown for markers that were not taken into account because their level of polymorphism was too low, as reported in file mrkpolymorphism.dat (see below). Codes 8 and 9 are used for (markers in) haploblocks for which no solution was found (Table 1).
The layout of the haploblock statistics file is similar to that if the phased genotypes file: one row for each individual, and two columns per haploblock: first the allele inherited from parent 1, then that from parent 2. At the top and left of this matrix are 8 rows and columns labelled c0 .. c9 giving the totals for each code in that row or column and top-left is a diagonal giving the totals per code over the whole data set. The top row has the names of the haploblocks, suffixed by _2nd for the second column of each haploblock. At the very top, below the haploblock name, its chromosome and position are listed.
Output of this file can be suppressed by adding hbstatisticsfile="" to the (fq_)haplotyping_session function call.

name	c0	c1	c2	c3	c4	c7	c8	c9	HB1	HB1_2nd	HB2	HB2_2nd
chrom									1		1	
pos									32.038		45.814	
c0	119								29	29	31	30
c1		4							1	1	1	1
c2			0						0	0	0	0
c3				8					3	3	1	1
c4					1				0	0	0	1
c7						0			0	0	0	0
c8							0		0	0	0	0
c9								0	0	0	0	0
Golden_Delicious	4	0	0	0	0	0	0	0	0	0	0	0
Delicious	4	0	0	0	0	0	0	0	0	0	0	0
Cameo	4	0	0	0	0	0	0	0	0	0	0	0
Cox	4	0	0	0	0	0	0	0	0	0	0	0
F2_26829-2-2	4	0	0	0	0	0	0	0	0	0	0	0
...												

mrkstatistics.dat
This file is identical in layout and meaning to the previous hbstatistics.dat, except that now the statistics are given for the individual markers rather than for the haploblocks, and that there is an extra row *hb* below row *pos*. In this file (not shown) all entries for marker M10 are 7, meaning "marker not analyzed" (Table 1); this corresponds with the listing of usemarker=0 for this marker in file mrkpolymorphism.dat (see above).

Output of this file can be suppressed by adding mrkstatisticsfile="" to the (fq_)haplotyping_session function call.

pedigree.dat
This file is identical to the pedigree input file, except that the file is sorted such that parents always appear before all their progeny, which is not required for the input file.

hbmap.dat

marker	chrom	pos
HB1	1	32.038
HB2	1	45.814

This is the genetic map of the haploblocks. The position of each haploblock is the average of the positions of all markers in the haploblock. The format of the file is different from that of the input marker map file because here there is no column for the haploblocks; actually the haploblock names are now in the marker column (because in follow-up analyses such as QTL mapping, these haploblocks will be considered to be multi-allelic markers).

mrkmap.dat
This file is identical to the input marker map file, except that the order of the columns is always marker – chrom – pos – hb, which may be different in the input map file.

Pedimap files (four files with extension .ped)
There are two files in Pedimap format ((Voorrips et al., 2012, https://www.wageningenur.nl/en/show/Pedimap.htm) that allow to compare visually the haploblock alleles at the start (**hborig.ped**) and at the end (**hbfinal.ped**). There are also two Pedimap file with the marker alleles at the start (**mrkorig.ped**) and end (**mrkfinal.ped**). In the two 'final' files the alleles have a color code according to Table 1.
Output of these files can be suppressed by adding any or all of origpedimaphbfile="", finalpedimaphbfile="", origpedimapmrkfile="" and/or finalpedimapmrkfile="" to the (fq_)haplotyping_session function call.

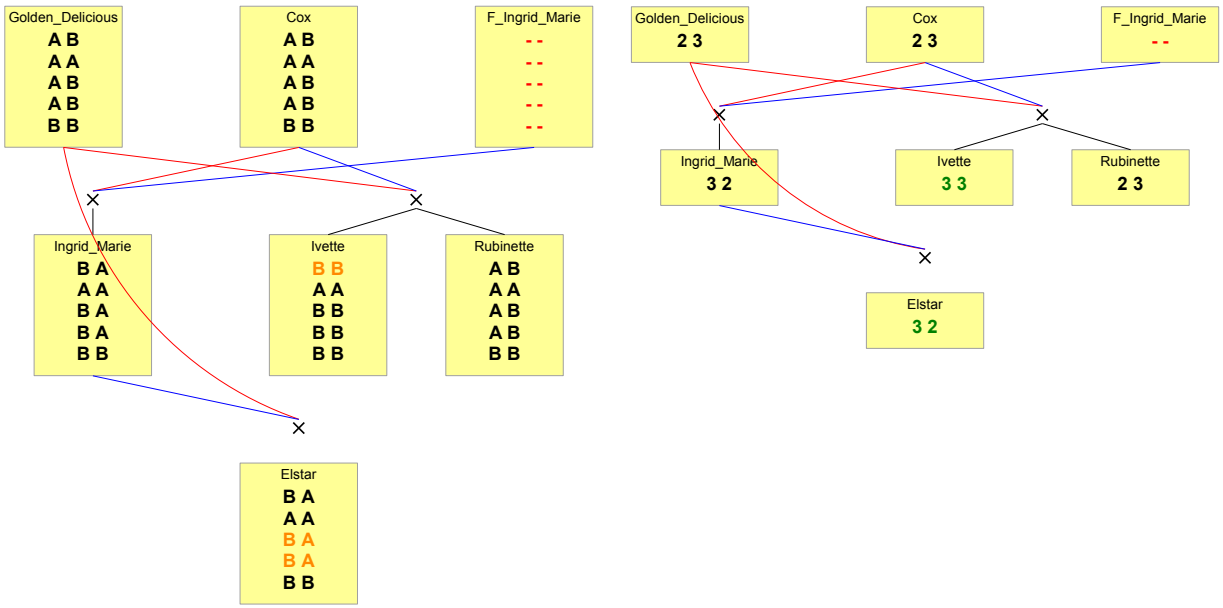

Figure 1. Two Pedimap pictures showing part of the Example pedigree. The left panel shows the marker alleles composing haploblock HB1 (from mrkfinal.ped) , the right panel the haploblock alleles for HB1 (from hbfinal.ped), both at the end of the alleles assignment process. The alleles are color-coded according to Table 1: black=0, score not changed; red=1, missing score unchanged; orange=2, missing score imputed; green=3, score changed to another allele. Imputation of missing marker alleles in Ivette and Elstar (left

panel) leads to a change in the haploblock alleles (right panel): from an allele with one missing marker score to one with no missing marker score.

Files produced only for generic output

finalhb_geno.dat and orighb_geno.dat

These two files contain the phased haploblock genotypes at the start and end of the process.
Output of these files can be suppressed by adding any or all of origdatahbfile="" and/or finaldatahbfile="" to the haplotyping_session function call.

name	HB1	HB1_2nd	HB2	HB2_2nd
Golden_Delicious	2	3	2	3
Delicious	4	5	4	5
Cameo	2	5	2	5
Cox	2	3	3	6
F2_26829-2-2	4	4	7	3
PRI14-126	3	4	3	7
...				

The format of these files is the same as that of the phased genotypes input file: one row per individual, 2 columns per haploblock; to the caption of the second column of each haploblock "_2nd" is appended. The haploblock allele names are as discussed for the hballeles.dat file. Note that for haploblock HB1, which consists of 5 markers, just one digit indicating the number of missing marker scores is used, while for HB2, with 12 markers, 2 digits are needed.

finalmrk_geno.dat and origmrk_geno.dat

These files are similar to the preceding two files, but contain the phased marker genotypes at the start and end of the process, rather than the phased haploblock genotypes.
Output of these files can be suppressed by adding any or all of origdatamrkfile="" and/or finaldatamrkfile="" to the haplotyping_session function call

name	M01	M01_2nd	M02	M02_2nd	M03	M03_2nd	...
Golden_Delicious	A	B	A	A	A	B	
Delicious	A	A	A	B	A	A	
Cameo	A	A	A	B	A	A	
Cox	A	B	A	A	A	B	
F2_26829-2-2	A	A	A	A	A	A	
PRI14-126	B	A	A	A	B	A	
PRI612-1	A	A	B	A	A	A	
...							

The format is the same as that of the phased genotypes input file: one row per individual, two columns per marker. The contents of the origmrk_geno.dat file are identical to those of the input file, but the order of the markers and individuals are according to the map and pedigree, which is not necessary in the input file. In our example the input file was already ordered so the files are identical except perhaps for the captions of the first column (the individuals) and the second column of each marker; these captions are ignored by PediHaplotyper.

Files produced only for FlexQTL output

Four sets of three files are produced: for the haploblocks and for the markers, at the start and at the end of the process. Each set of files is a valid set of input files for FlexQTL. We show the files for the end of the process; those for the start have an identical format.
Output of these sets of files can be suppressed by adding any or all of origflexqtlhbfiles="",

finalflexqtlhbfiles="", origflexqtlmrkfiles="" and/or finalflexqtlmrkfiles="" to the fq_haplotyping_session function call.

finalhb_flexqtl.map

```
group     1
HB1       32.038
HB2       45.814
```

This is the haploblock in FlexQTL format. This map is identical for the start and end of the process. If there were more chromosomes the second would again start with a line "group".

A_finalmrk_flexqtl.map

```
group     1
M01       32
M02       32.01
M03       32.03
M04       32.07
...
```

This is the marker map in FlexQTL format. It is identical to the input file flexqtl.map, except that it does not have the haploblock after each marker; this is needed for the map file to be accepted by FlexQTL.

finalhb_flexqtl.dat

This file combines the pedigree (including the 'phenotypic' data columns) and the phased haploblock genotypes.

;population	name	parent1	parent2	Nuisance_0	Row#	Resistance	HB1	HB1_2nd	HB2	HB2_2nd
1	Golden_Delicious	-	-	1	8	0	2	3	2	3
1	Delicious	-	-	1	3	0	4	5	4	5
1	Cameo	Golden_Delicious	Delicious	1	1	0	2	5	2	5
1	Cox	-	-	2	2	0	2	3	3	6
1	F2_26829-2-2	-	-	1	4	1	4	4	7	3
1	PRI14-126	Golden_Delicious	F2_26829-2-2	1	18	1	3	4	3	7

...

The first line is ignored by FlexQTL but is added by PediHaplotyper for documentation. The file orighb_flexqtl.dat shows the same information for the start of the process. The finalmrk_flexqtl.dat and origmrk_flexqtl.dat show the same information for the markers rather than the haploblocks; they contain pairs of columns for each marker.

finalhb_flexqtl.par

The FlexQTL parameter (*.par) files are a copy of the input parameter file, with changes for the data file and map file lines. For the haploblock files (finalhb_flexqtl.par and orighb_flexqtl.par) also the nmrkC line is adapted to the number of haploblocks rather than the number of markers.

References

Bink, M.C.A.M., Jansen, J., Madduri, M., Voorrips, R.E., Durel, C.-E., Kouassi, A.B., Laurens, F., Mathis, F., Gessler, C., Gobbin, D., et al. (2014). Bayesian QTL analyses using pedigreed families of an outcrossing species, with application to fruit firmness in apple. Theor. Appl. Genet. *127*, 1073–1090.

Voorrips, R.E., Bink, M.C.A.M., and van de Weg, W.E. (2012). Pedimap: software for the visualization of genetic and phenotypic data in pedigrees. J. Hered. *103*, 903–907.

Links

Pedimap: https://www.wageningenur.nl/en/show/Pedimap.htm

FlexQTL: https://www.wageningenur.nl/en/show/FlexQTL.htm

R: https://cran.r-project.org/

Appendix 1: How to obtain a phased marker genotypes file using FlexQTL

To produce the mhaplotypes.csv file, you will need to run FlexQTL with specific values for the following variables in the FlexQTL parameter file, i.e.,

```
skipSampleMarkers  0
REDprint           0
```

The variable `skipSampleMarkers` needs to be equal to zero to perform proper calculation of the haplotype probabilities (which is not critical in the QTL mapping analysis).

In addition, for dense marker maps you may want to set the following value (this will improve mixing of the unknown variables):

```
markerblock        5
```

The above implies that you will need to run a separate FlexQTL simulation to obtain the mhaplotypes.csv file. However, you will only have to do this once for your dataset as the marker data (and haplotypes) will be the same across the various traits and various QTL models that you may wish to study.