Haplin data formats

Håkon K. Gjessing

2011-12-09

Introduction

Haplin can handle data in three different ways:

1. **Haplin's own text file format.** This format places a nuclear family on a single line in the data file, with genotypes for maternal, paternal, and fetal genes in separate columns. When running `haplin`, the argument `filename` is used to specify the data file. The arguments `data` and `pedIndex` are not used. The input file format is relatively flexible and allows multi-allelic markers. However, for larger number of markers, say, more than a thousand, this approach is less efficient than converting to the GenABEL format (see below).

2. **Convert from ped-format to Haplin.** The supplied conversion function `pedToHaplin` converts a "standard ped format" file directly to Haplin's own data format. The specific ped format required by Haplin is described in detail below. Analyses can then proceed as with the Haplin data format. The `pedToHaplin` function is somewhat flexible in terms of what separators can be used. It allows multi-allelic markers, and the individual id needs only be unique within each family id (although it's always a good idea to use a unique individual id). The help file of `pedToHaplin` provides more details. While fairly large files can, in principle, be converted and run in the Haplin format, or cut into separate pieces before being fed to Haplin, it is much more memory efficient to use the GenABEL format described below. *Important:* The `pedToHaplin` function has not been tested very extensively, so it's always recommended to check the converted file before use.

3. **Convert from ped-format to GenABEL format.** Haplin can convert from a text file in the ped format to the internal data format used by the GenABEL package. This does not require previous knowledge of the GenABEL package, as a few commands will suffice to do the appropriate data conversion. Once a GenABEL data object has been created and loaded into R, haplin is run by using the `data` argument to specify the GenABEL object. The additional `pedIndex` argument in `haplin` is used to specify the name of a file

containing the family information. This information is extracted from the ped file during conversion. The argument `filename` is not used. For larger data files, in particular GWAS data, it is recommended to use the GenABEL data format. A drawback, however, is that it cannot handle multi-allelic markers, only snps, and since it is only intended for case-control data, Haplin needs an extra step of data processing to recover the family information and place it in the pedIndex file. When installing Haplin, the GenABEL [1] package is automatically installed as well, so no extra steps are needed to install it.

In the following, we first describe the Haplin data format in detail. If you have a modest number of markers in your file, the standard format is quite sufficient, and `pedToHaplin` can be used to convert from the ped format, if necessary. If so, the part about the GenABEL conversion is not needed, and the information you need is contained in Section 1 in this document.

On the other hand, if you are dealing with a large data file that needs the GenABEL data setup, we suggest you skip directly to Section 2, where details of the conversion to GenABEL data are described. In that case, a few more (simple) steps of data conversion are needed, as explained below.

1 The Haplin data format

Haplin requires data to be in an ASCII file in a specific format:

- The data can contain a number of leading columns with covariates (such as a case/control variable), followed by columns containing the genetic data.
- Each line represents a case-parent triad (or, in the case-control design, a single individual).
- Columns should be separated by white space.
- Within each column the two alleles for that individual in that locus are separated by a semi-colon, such as 1;2, C;T, A;A etc.
- Missing data are coded as NA.
- There should be no row or column names in the file.

(For the convenience of the user, other separators and missing data indicator can be specified using the arguments sep, allele.sep and na.strings, see below)

In addition, the file structures slightly differ from design to design:

1.1 The case-parent triad design

This design is specified in `haplin` using the `design = "triad"` argument, and involves the "pure" case-parent triad design.

Each line represents one triad. There are three columns for each locus, one for the mother (M), one for the father (F) and one for the child (C). The columns are

placed in the following sequence (where the numbers indicate marker):

M1 F1 C1 M2 F2 C2 ...etc.

Important: Make sure the sequence is correct, this is the only way for HAPLIN to figure out which is which.
To illustrate, for 2 loci with 4 and 2 alleles, respectively, one would have a setup of the following type:

```
    marker 1      marker 2
|--------------||-------------|
 4;4   4;4   4;4   T;T   C;T   C;T     <- Triad no. 1
 2;4   2;4   2;4   T;T   T;T   T;T     <- Triad no. 2
 2;4   2;4   2;4   T;T   T;T   T;T       etc.
 2;4   2;2   2;4   T;T   T;T   T;T
 2;3   4;4   3;4   T;T   C;T   T;T
 4;4   2;4   4;4   T;T   T;T   T;T
|---||---||---||---||---||---|
  M     F     C     M     F     C
```

Assuming there could also be missing data, the first four lines of data might look like

```
4;4 4;4 4;4 T;T C;T C;T
2;4 2;4 2;4 T;T T;T  NA
2;4  NA 2;4 T;T T;T T;T
2;4 2;4 2;4 T;T T;T T;T
```

(Note the NAs that indicate missing genotype at the first marker of the father in the third triad, and at the second marker of the child in the second triad.)

1.2 The combined case-parent triad and control-parent triad design (hybrid design)

This design is specified in `haplin` using the `design = "cc.triad"` argument, and involves case-parent triad data combined with control-parent triad data.

The data format is identical to the triad format, but there must be an extra column to the left of the genetic data, specifying the case/control status of the triad.
Important: The case/control column must be numeric with two different values. The largest one will always be used to denote a case triad!
So, for example, the first four lines could look like

```
0 4;4 4;4 4;4 T;T C;T C;T
1 2;4 2;4 2;4 T;T T;T  NA
1 2;4  NA 2;4 T;T T;T T;T
0 2;4 2;4 2;4 T;T T;T T;T
```

which would indicate that lines 2 and 3 are case triads, lines 1 and 4 are controls. Note that, if for instance only the control child has been genotyped, not the parents, one can use a file like

```
0   NA   NA 4;4   NA   NA C;T
1 2;4 2;4 2;4 T;T T;T   NA
1 2;4   NA 2;4   NA   NA T;T
0   NA   NA 2;4   NA   NA T;T
```

i.e. the control parents have been set to missing (NA).

1.3 The "pure" case-control design

This design is specified in `haplin` using the `design = "cc"` argument, and involves case-control data; no parents are genotyped.

The format should be as for the `cc.triad` (hybrid) data above, but columns relating to parents should be completely removed, as in

```
0 4;4 C;T
1 2;4   NA
1 2;4 T;T
0 2;4 T;T
```

1.4 "Intermediate" designs

As noted above, Haplin can use reduced versions of the above designs. Two important special cases are:

- Case-parent triads with only control children, no control parents. Specified using `design = "cc.triad"` and setting the columns for control parents to missing (NA).
- Case-parent dyads combined with control-parent dyads, no fathers. Specified using `design = "cc.triad"` and setting the column for the fathers to missing (NA).

1.5 Note (for all designs)

- For all the designs, the file can contain an arbitrary number of columns to the left of the genetic data. The number of columns should be specified using the argument `n.vars`. The default, `n.vars = 0`, applies to the triad design, if no covariate columns are present.
- The design should be specified with the `design` argument, which takes the values `"triad"` (default), `"cc.triad"` and `"cc"`.

1.6 Using other separators

To improve the flexibility of Haplin for the user, there are two arguments to haplin, `sep` and `allele.sep`, which can be used to set the separators between columns and within columns, respectively. For instance, space can be used for both, and the file could then look like

```
      marker 1           marker 2
|-----------------||-----------------|
 4   4   4   4   4   4   T   T   C   T   C   T
 2   4   2   4   2   4   T   T   T   T   T   T
 2   4   2   4   2   4   T   T   T   T   T   T
 2   4   2   2   2   4   T   T   T   T   T   T
 2   3   4   4   3   4   T   T   T   T   T   T
 4   4   2   4   4   4   T   T   T   T   T   T
|----||----||----||----||----||----|
  M      F      C      M      F      C
```

Or, for instance, with `allele.sep = ""` (empty) and `sep = " "` (space) it would be

```
44 44 44 TT CT CT
24 24 24 TT TT TT
24 24 24 TT TT TT
24 22 24 TT TT TT
23 44 34 TT TT TT
44 24 44 TT TT TT
```

1.7 Marker selection

To run HAPLIN on various selections of the markers, you don't have to create a separate file for each selection. The markers argument in `haplin` can be set to, for instance, `c(2,3)`, which will use only the second and third markers in the data file, as in the command `haplin("filename", markers = c(2,3))`. Note that if the argument `use.missing` is set to `FALSE`, HAPLIN will exclude all triads with any form of missing data (all non-complete triads). However, it will only look at markers chosen by the `markers` argument, so that triads with missing data on unused markers will not be removed as long as they are complete on the selected markers.

2 Convert from ped format to GenABEL format

This approach is particularly useful for large and very large data files with only snp data, for instance GWAS data. We start by describing the input data, then the steps needed to convert the data to the GenABEL format. The GenABEL [1] package is designed to handle GWAS data in a memory-efficient way. However, the GenABEL data handling does not keep track of family information, so Haplin needs an extra data conversion step to extract the necessary triad information from the ped file.

There is no need to know much about the GenABEL package since the conversion steps are relatively straightforward, and once the conversion is done, the resulting data object can be fed directly to Haplin. Still, we include a few useful GenABEL suggestions at the end, for the benefit of the user.

2.1 Quick summary

We start with a quick summary of the steps needed:

1. Extract family and phenotype information:

```
prepPed(pedfile = "data/mygwas.ped", outdir = "data",
    create.map = T)
```

2. Convert to raw file format:

```
convert.snp.ped(pedfile = "data/mygwas.ped", mapfile
   = "data/mygwas.map", outfile = "data/mygwas.raw")
```

3. Load into R:

```
mygwas.data <- load.gwaa.data(phenofile = "data/
   mygwas.ph", genofile = "data/mygwas.raw")
```

4. Run `haplinSlide`:

```
haplinSlide(data = mygwas.data, pedIndex = "data/
   mygwas.pedIndex", markers = ... etc.)
```

Each step is explained in detail in the following sections.

2.2 The input data format

Assume the data are contained in a ped file named `mygwas.ped`. The structure of the file should be something like this:

```
1104   1104-1   1104-2   1104-3   1   1   A   B   B   B
1104   1104-2        0        0   1   1   B   B   A   B
1104   1104-3        0        0   2   1   A   B   A   B
1105   1105-1   1105-2   1105-3   2   2   B   B   A   A
1105   1105-2        0        0   1   2   B   B   A   A
1105   1105-3        0        0   2   2   0   0   A   A
```

The column values are:

1. Family id
2. Individual id
3. Father's id

4. Mother's id

5. Sex (1 = male, 2 = female)
   or alternatively, (1= male, 0 = female)

6. Case-control status (0 = controls, 1 = cases)

Column 7 and onwards contain the genotype data, with alleles in separate columns, *or* joined, as AB BB, etc. A "0" is used to denote missing data. In the example file, alleles follow the generic A/B Illumina coding, but any of the standard coding patterns, such as 1/2, C/T etc. could be used.

Id variables do not need to have any specific format, but make sure the individual id variable labels are unique! In the columns for the father's and mother's id's, a "0 " is used when that parent is not in the file.

For a pure case-control design, where no parents have been genotyped, the third and fourth columns should be all "0".

Note that Haplin always uses the 1/2 coding for male/female, whereas GenABEL uses 1/0. The `prepPed` function described below will thus always create a `mygwas.ph` file with the 1/0 coding to prepare for loading with `load.gwaa.data`, if necessary converting from 1/2 in `mygwas.ped` to 1/0 in `mygwas.ph`. When `haplin` is subsequently run on the loaded object it will automatically convert back to 1/2. If you use the `sel.sex` argument in `haplin` to restrict the analysis to males or females, make sure you use the 1/2 coding.

The case-control status variable can have any numeric coding, as long as (at most) two values are used. The variable is passed unchanged to Haplin, which always assumes the largest one are the cases, the smallest the controls. So, for instance, the coding (1 = controls, 2 = cases) would also be valid. For consistency, however, it might be a good idea to stick to the 0/1 coding suggested above.

Missing values in the sex and case-control columns are not accepted.

2.3 Extract family and phenotype information

To extract family information and phenotype information, use a command like

```
prepPed(pedfile = "data/mygwas.ped", outdir = "data",
   create.map = T)
```

`pedfile` specifies the location and name of the ped file, `outdir` specifies the directory where you want your data files to be stored, in this example the same directory as the `mygwas.ped`. The function `predPed` creates three files:

1. The pedIndex file `data/mygwas.pedIndex`. This file contains the family information extracted from the ped file, in a special format. It file will be used by Haplin later on, when converting from GenABEL format to Haplin format. It should not be modified by the user.

2. A phenotype file `data/mygwas.ph`. This file is in a format suitable for loading into GenABEL, and it contains three columns, named "id", "sex", and "cc". Do not change the id variable, as this is matched to the id variable in the `mygwas.raw` data when loading into R. The file will look something like this:

```
     "id"  "sex"   "cc"
"1104-1"    "1"    "1"
"1104-2"    "1"    "1"
"1104-3"    "0"    "1"
"1105-1"    "0"    "2"
"1105-2"    "1"    "2"
"1105-3"    "0"    "2"
```

Note that the "sex" variable has been converted to 1/0, as discussed above. If relevant, you may add extra columns directly to this file. The extra columns could contain, for instance, grouped exposure variables that you might want to stratify your analysis on. Just make sure the extra columns are added to the right of the first three columns, and do not change the values of the first three columns.

3. Optionally, when create.map = T, prepPed will also create an "dummy" map file `data/mygwas.map`. If the `mygwas.ped` file already has an accompanying `mygwas.map` file, it is better to just set create.map = F and use the supplied `mygwas.map` file. Otherwise, prepPed will create the new file having the same structure as a map file, but where the map information is artificial. The only purpose of this map file is to get GenABEL to load the data even if the original map file is unavailable. The map file should look something like this:

```
chr        name     pos
  1   rs9629043  554636
  1  rs12565286  711153
  1  rs12138618  740098
```

The artificial map file created by `prepPed`, if requested, will simply have snp names and positions that are values running from 1 upwards, and the chromosome set to 0. It will not make genetic sense; it is only a dummy map file to be fed to `convert.snp.ped` to allow conversion (see below).

All files produced by prepPed are given the same name as the input ped file, only the file extensions will be changed. **Warning:** Files in the `data` directory with the same names as the pedIndex, phenotype, or map file will be overwritten without question.

2.4 Convert to GenABEL raw format

Having prepared the family, phenotype, and map information files as above, the ped file can be converted to a raw GenABEL file. The function convert.snp.ped from the GenABEL package will do this, and the basic syntax is:

```
convert.snp.ped(pedfile = "data/mygwas.ped", mapfile = "
   data/mygwas.map", outfile = "data/mygwas.raw")
```

The out argument specifies the name of the raw output file. You may specify any name, but for consistency it might be a good idea to place it in the same directory as the other files, and to use the same name, except with a .raw extension. Converting a full GWAS file may take a little while. For other conversion options, see the help file on convert.snp.ped in the GenABEL package.

2.5 Loading data into R

To use the .raw data in Haplin, you must first load them into R. This is done by creating a GenABEL data object (of class `gwaa.data`) in R, which serves as a link to the data on disk. The function to do this is load.gwaa.data:

```
mygwas.data <- load.gwaa.data(phenofile = "data/mygwas.ph
   ", genofile = "data/mygwas.raw")
```

Note that the created "link" is stored in R with the name `mygwas.data` (or whatever name you choose). Whenever starting a new R session, you may reload this with the load.gwaa.data function, or you may leave the `mygwas.data` in the R workspace and save it when exiting R.

2.6 Using the GenABEL data object in Haplin

To run `haplin` using the loaded GenABEL object instead of a standard Haplin file, use all the standard `haplin` syntax, but rather than using the `filename` argument, the data and pedIndex arguments should be used:

```
haplin(data = mygwas.data, pedIndex = "data/mygwas.
   pedIndex", markers = ... etc.)
```

Haplin will then extract and convert the relevant data from the `mygwas.data`, using the pedIndex file to keep track of families.

When extracting data from a GenABEL object like this, the `haplin` arguments `sep`, `allele.sep`, `na.strings` don't have to be set; they are simply ignored. In addition, `haplin` extracts the arguments `n.vars`, `ccvar`, and `sex` from the `mygwas.data` object automatically. Other arguments, like `design` and `markers`, may be set as needed.

2.7 Additional comments

While the above is sufficient to start running Haplin, a few other suggestions and GenABEL commands may be useful.

Adding extra covariates to the data

Other covariates/exposure variables can be added to the `mygwas.ph` file, to the right of the first three columns, as described above.

Extracting information from the data object

The snp names of the object can be extracted with `snpnames(mygwas.data)`, the id's with `idnames(mygwas.data)`, the gender of each individual with `male(mygwas.data)`, the chromosome with `chromosome(mygwas.data)`, the snp names of the autosomal snps with `autosomal(mygwas.data)`, and phenotype data with `phdata(mygwas.data)`. The latter extracts a data frame with a column for the id variable, and columns for sex, case-control status, and possibly other covariates. The sex variable should be the same as the one extracted with `male`.

Warning: With the exception of subsetting (see below), you should probably not do modifications to the `mygwas.data` object, including the phenotype data, unless you really understand what you are doing.

Checking the data conversion

You may perhaps want to check that the `mygwas.data` object converts correctly before running Haplin. To check, say, the three first snps, use

```
temp <- gwaaToHaplin(mygwas.data[, 1:3], pedIndex = "data
   /mygwas.pedIndex")
```

The `temp` object should be a character matrix where each line of data corresponds to a family, in a format similar to that described for the original Haplin data format. Make sure you don't try to convert more than a moderate number of snps at a time. However, the `gwaaToHaplin` function is mostly for internal use and should not be needed.

Run haplin on subsets of snps

To run an analysis on a subset of the snps you can, as before, use the `markers` argument in `haplin` and `haplinSlide`. For instance,

```
haplin(data = mygwas.data, pedIndex = "data/mygwas.
   pedIndex", markers = 20:22, ...etc.)
```

will run `haplin` only on snp 20 to 22 (and reconstruct length 3 haplotypes for these snps). Using

```
haplinSlide(data = mygwas.data, pedIndex = "data/mygwas.
   pedIndex", markers = 100:1000, winlength = 2, ...etc.)
```

will slide windows of length 2 from snp 100 up to snp 1000, reconstructing length 2 haplotypes along the way. To select snps by name, one can for instance use

```
snpuse <- c("rs9629043", "rs12565286", "rs12138618")
snpselect <- which(snpnames(mygwas.data) %in% snpuse)
haplin(data = mygwas.data, pedIndex = "data/mygwas.
   pedIndex", markers = snpselect, ...etc.)
```

Note that we use `which` to make sure the `snpselect` variable is integer, not logical, since the `markers` argument only accepts numbers. The `snpuse` list of snps in this

example can of course be made as long as you like when running `haplinSlide`, but it should be kept in mind that when running `haplinSlide` with `winlength` larger than 1, the snps should be kept in correct physical ordering for the haplotype reconstruction to make sense. To keep track of snpnames, it might be useful to add the names to the output from `haplinSlide` after running:

```
result <- haplinSlide(data = mygwas.data, pedIndex = "
   data/mygwas.pedIndex", markers = 100:1000, winlength =
    2, ...etc.)
names(result) <- snpnames(mygwas.data)[100:1000]
```

As an alternative to using the `markers` argument, the data object can be subsetted directly. Create a new data object with, for instance,

```
mygwas.data1 <- mygwas.data[, 100:1000]
```

and run `haplin` or `haplinSlide` on `mygwas.data1`. Note: If you use the `markers` argument when running `haplin` or `haplinSlide` on the new `mygwas.data1` object, the `markers` now refer to the snps in the reduced `mygwas.data1` data, not the original.

A useful selection could be to run `haplinSlide` on a selection of chromosomes. To select chromosomes 1 and 2, say,

```
snpselect <- which(chromosome(mygwas.data) %in% c(1,2))
```

and then use the `markers` argument or subsetting, as above.

Using

```
mygwas.data1 <- mygwas.data[, autosomal(mygwas.data)]
```

can similarly select all autosomal snps/chromosomes for analysis. Since Haplin (the current version) does not detect the X-chromosome automatically, this is useful in order to split analyses over the autosomal snps and the X-chromosome snps.

Run haplin on subsets of families

Using subsetting of the `mygwas.data` object is also useful when restricting the analysis to selected individuals/families in the data file. For instance, if the phenotype data contains a variable named `group` (that you added to the `mygwas.ph` file), with values 1 and 2, you can subset the data with

```
indselect <- (phdata(gwas.data)$group == 1)
mygwas.data1 <- mygwas.data[, indselect]
```

Alternatively, if you have, for instance, an R vector `tobeselected` containing the ids of the individuals to be selected, you can subset with

```
indselect <- (idnames(gwas.data) %in% tobeselected)
mygwas.data1 <- mygwas.data[, indselect]
```

Important: Both subsetting approaches select on an individual basis, not on a family basis. It is a good idea to make sure that the `group` variable is correctly defined so that complete nuclear families always belong to the same group.

Run haplin on only case-parent triads or control-parent triads

If you have hybrid data, with both case triads and control triads, you can still run an analysis on cases alone by selecting on the `cc` variable. Do this as with the `group` variable in the example above. Since the `cc` variable refers to the status of the child, you should make sure the parents are given the same `cc` code as the child so that entire families are selected.

Obviously, if your original data consist of only case-parent triads you only use a single code for the cc variable, and use the `design = "triad"` argument.

Important

Do not change the id's of `mygwas.data`, and do not make changes to the pedIndex file, or the .ph file (except perhaps adding columns). The id's stored in the pedIndex file used by Haplin must correspond to those in `mygwas.data`. However, subsetting `mygwas.data`, i.e. removing individuals or snps can be done without changing the pedIndex file.

References

[1] GenABEL developers (2011). GenABEL: genome-wide SNP association analysis. R package version 1.6-7. http://CRAN.R-project.org/package=GenABEL