alleHap: Allele Imputation and Haplotype Reconstruction from Pedigree Databases

Nathan Medina-Rodriguez

Angelo Santana

Package Version: 0.9.2

October 5, 2015

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Departamento de Matemáticas

Abstract

`alleHap` contains tools to simulate alphanumeric alleles, impute genetic missing data and reconstruct non-recombinant haplotypes from pedigree databases in a deterministic way. Allelic simulations can be implemented taking into account many factors (such as number of families, markers, alleles per marker, probability and proportion of missing genotypes, recombination rate, etc). Genotype imputation can be used with simulated datasets or previously loaded databases (in .ped file biallelic format). Haplotype reconstruction can be performed even with missing data, because of `alleHap` firstly imputes each family genotype (considering that each member, due to meiosis, should unequivocally have two alleles, one inherited from each parent. `alleHap` is very robust against inconsistencies within the genotypic data, warning when they occur. Furthermore, the package is handy, intuitive and consumes little time, even when handling large amounts of genetic data. This vignette intends to explain in detail how `alleHap` package works for the desired applications, and it includes illustrated explanations and easily reproducible examples.

Contents

1 Introduction

Genotype imputation and haplotype reconstruction have achieved an important role in Genome Wide Association Studies (GWAS) during recent years. Estimation methods are frequently used to infer either missing genotypes as well as haplotypes from databases containing related and/or unrelated subjects. The majority of these analysis have been developed using several statistical methods [BB11] which are able to impute probabilistically genotypes as well as perform haplotype phasing (also known as haplotype estimation) of the corresponding genomic regions.

Currently algorithms do not carry out genotype imputation or haplotype reconstruction using deterministic techniques on pedigree databases. Despite the fact that computational inference by probabilistic models may cause a number of incorrect inferences, studies composed of large pedigrees are very infrequent. These methods are usually focused in population data and in case of pedigree data, families normally are comprised by trios [BB09], being uncommon those studies consisting of more than two offspring for each line of descent.

On the other hand, certain regions are very stable against recombination but at the same time they may be highly polymorphic. For this reason, in some well studied regions (such as Human Leukocyte Antigen (HLA) loci [MCH$^+$13] in the extended human Major Histocompatibility Complex (MHC) [dBMS$^+$06]) an alphanumeric nomenclature is needed to facilitate later analysis. Under this juncture, the available typing techniques usually are not able to determine the allele phase and therefore the constitution of the appropriate haplotypes is not possible. Although some computational methods have been evaluated for the reconstruction of haplotypes [CMJVC$^+$10], none of them is capable to perform haplotype phasing or genotype imputation of missing data without using reference panels and probabilistic techniques which may lead inaccurate results.

2 Theoretical Description

`alleHap` algorithms are based on a preliminary analysis of all possible combinations that may exist in the genotype of a family, considering that each member should unequivocally have inherited two alleles, one from each parent. The analysis was founded on the differentiation of seven cases, as was described in [BWSD$^+$07]. Each case has been grouped considering the number of unique (or different) alleles per family. So, using the notation N_{par}: *Number of unique alleles in parents* and N_p: *Number of unique alleles in parent p*, the expression: (N_{par}, N_1, N_2) will be able to identify all the non-recombinant configurations in families with one line of descent (i.e. parent-offspring pedigree). The table 1 shows the different configurations in biallelic mode:

Configurations	1	2	3	4	5	6	7
N_{par}	1	2	2	3	2	3	4
(N_1,N_2)	(1,1)	(1,1)	(1,2)	(1,2)	(2,2)	(2,2)	(2,2)
Parents	a/a a/a	a/a b/b	a/a a/b	a/a b/c	a/b a/b	a/b a/c	a/b c/d
Possible Offsring	a/a	a/b	a/a a/b	a/b a/c	a/a b/b a/b	a/a a/b a/c b/c	a/c a/d b/c b/d

Table 1: Biallelic configurations in a parent-offspring pedigree

An identification of the homozygous genotypes for each family is necessary for the proper operation of `alleHap`. An example of some biallelic genotypes (left) and their corresponding Homozygosity (HMZ) matrix is shown in the next table:

	Unphased Data							Homozygosity Info.						
Marker	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Parents	a/a	a/a	a/a	a/a	a/b	a/b	a/b	1	1	1	1	0	0	0
	a/a	b/b	a/b	b/c	a/b	a/c	c/d	1	1	0	0	0	0	0
Offsring	a/a	a/b	a/a	a/b	a/a	a/a	a/c	1	0	1	1	1	0	0
			a/b	a/c	b/b	a/b	a/d			0	1	1	0	0
					a/b	a/c	b/c					0	0	0
						b/c	b/d						0	0

Table 2: Biallelic genotypes and HMZ matrix

In order to perform the phasing of the genotypes, `alleHap` firstly creates an IDentified/Sorted (IDS) matrix per family. An example of the phased genotypes of a family (left) and their corresponding IDS values (right) is the following:

	Phased Data							IDS Information						
Marker	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Parents	a|a	a|a	a|a	a|a	a|b	a|b	a|b	1	1	1	1	1	1	1
	a|a	b|b	a|b	b|c	a|b	a|c	c|d	1	1	1	1	1	1	1
Offspring	a|a	a|b	a|a	a|b	a|a	a|a	a|c	1	1	1	1	1	1	1
			a|b	a|c	b|b	a|b	a|d			1	1	1	1	1
					a/b	a|c	b|c					0	1	1
						b|c	b|d						1	1

Table 3: Phased genotypes and IDS matrix

Sometimes, missing values may occur. These can be located either in parents or children. An example of this is depicted as follows:

	Phased and Missing Data							IDS Information						
Marker	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Parents	a|a	a|a	a|a	a|a	NA	NA	NA	1	1	1	1	0	0	0
	NA	NA	NA	b|c	a|b	a|b	a|b	0	0	0	1	1	1	1
Offspring	a|a	a|b	a|a	NA	a|a	a|a	a|c	1	1	1	0	1	1	1
			a|b	a|c	b|b	a/b	a|d			1	1	1	0	1
					a/b	a|c	b|c					0	1	1
						b|c	b|d						1	1

Table 4: Phased genotypes and IDS matrix with missing data

3 Input Format

`alleHap` only works with PED files, although it can detect and adapt similar formats (with the same structure) in order to later load the data.

3.1 PED files

A PED file is a white-space (space or tab) delimited file where the first six columns are mandatory and the rest of columns are the genotype: *Family ID* (identifier of each family), *Individual ID* (identifier of each member of the family), *Paternal ID* (identifier of the paternal ancestor), *Maternal ID* (identifier of the maternal ancestor), *Sex* (genre of each individual: 1=male, 2=female, other=unknown), *Phenotype* (quantitative trait or affection status of each individual: -9=missing, 0=unaffected, 1=affected) and ***Genotype*** (genotype of each individual in biallelic or coded format).

The IDs are alphanumeric: the combination of family and individual ID should uniquely identify a person. **PED files must have 1 and only 1 phenotype in the sixth column**. The phenotype can be either a quantitative trait or an affection status column. Genotypes (column 7 onwards) should also be white-space delimited; they can be any character (e.g. 1,2,3,4 or A,C,G,T or anything else) except 0 which is, by default, the missing genotype character. All markers should be biallelic and must have two alleles specified [PNTB+07]. For example, a family composed by 3 individuals typed for N SNPs is represented in Table 5:

Fam ID[a]	*Ind ID*	*Pat ID*	*Mat ID*	*Sex*	*Pheno*	*Mkr_1*		*Mkr_2*		*Mkr_3*			*Mkr_N*	
FAM001	1	0	0	1	0	A	A	G	G	A	C	...	C	G
FAM001	2	0	0	2	1	A	A	A	G	C	C	...	A	G
FAM001	3	0	0	1	0	A	A	G	A	A	C	...	C	A

Table 5: Example of a Family in .ped file format

[a] No header row should be given.

3.2 NA values

The missing values or Not Available (NA) values may be placed either in the first 6 colums or in genotype columns. In the genotype colums, when some values are missing either both alleles should be -9, NA, "NA" or "¡NA¿". An example of this would be:

```
   famID indID patID matID sex phenot Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2
1 FAM001     1     0     0   1      0     1     2    NA    NA     1     2
2 FAM001     2     0     0   2      0     3     4     1     2     3     4
3 FAM001     3     1     2   1      0     1     3     1     2     1     3
4 FAM001     4     1     2   2      0    NA    NA     1     1     2     4
5 FAM001     5     1     2   1      0     1     4     1     1     2     4
```

4 Data Simulation

This part of the package simulates biallelic pedigree databases which can be performed taking into account many different factors such as number of families to generate, number of markers (allele pairs), number of different alleles per marker, type of alleles (numeric or character), number of different haplotypes in the population, probability of parent/offspring missing genotypes, proportion of missing genotypes per individual, probability of being affected by disease and recombination rate.

4.1 alleSimulator Function

`alleSimulator` function generates the clinical and genetic information of a group of families according to the previously defined parameters. In order to simulate the data, this function has been developed in several steps:

I. **Internal Functions**: In this step are loaded all the necessary functions to simulate the data. These functions are *labelMrk* (which creates the 'A','C','G','T' character labels), *simHapSelection* (which selects n different haplotypes between the total number of possible haplotypes), *simOffspring* (which generates n offspring by selecting randomly one haplotype from each parent), *simOneFamily* (which simulates one family from a population containing the haplotypes 'popHaplos') and *simRecombHap* (which simulates the recombination of haplotypes).

II. **Alleles per Marker**: The second step is the simulation of a number of alleles per marker (if they are not supplied by user). It is assigned an allele range per marker whether alleles are not character type and, if alleles are character type, they are repeated two times.

III. **Haplotypes in population**: Once the number of alleles per marker and the haplotype size of the population (n) are specified, the population haplotypes are generated. In this proccess n different haplotypes were selected among the total number of possible haplotypes.

IV. **Data Concatenation**: In this step the clinical and the previous simulated data of all families are concatenated.

V. **Data Labelling**: The fifth step is the labelling of the previous concatenated data ("famID", "indID", "patID", "matID", "sex", "phenot", "markers", "recombNr", "ParentalHap", "MaternalHap").

VI. **Data Conversion**: The sixth step is the convertion of the previous generated data into the most suitable type (integer and/or character).

VII. **Missing Data Generation**: The seventh step is the insertion of missing values in the previous generated dataset (only when users require it). The missing values may be generated taking into account four different factors: *missParProb* (probability of parents' missing genotype), *missOffProb* (probability of offspring' missing genotype), *ungenotPars* (proportion of ungenotyped parents) and *ungenotOffs* (proportion of ungenotyped offspring).

VIII. **Function Output**: The last step is the creation of a list containing two different data.frames, for genotype and haplotypes respectively. This may be useful in order to compare simulated haplotypes with later phased haplotypes.

4.2 alleSimulator Examples

Below are listed a couple of examples of how `alleSimulator` works:

alleSimulator Example 1: *Simulation of a family containing parental missing data.*

```
> simulatedFam1 <- alleSimulator(1,2,3,missParProb=0.2,ungenotPars=0.2)

==========================================
===== alleHap package: version 0.9.2 ======
==========================================

Data have been successfully loaded from:
/tmp/RtmpkiEe3V/Rbuild17985792d8e/alleHap/vignettes

===== DATA COUNTING ======
Number of families: 1
Number of individuals: 4
Number of founders: 2
Number of children: 2
Number of males: 2
Number of females: 2
Number of markers: 3
============================
```

```
======== DATA RANGES =========
Family ID: FAM01
Individual IDs: [1,...,4]
Paternal IDs: [0,1]
Maternal IDs: [0,2]
Sex values: [1,2]
Phenotype values: [0]
==============================

===== MISSING DATA =====
Missing founders: 0
Missing children: 0
Missing paternal IDs: 2
Missing maternal IDs: 2
Missing sex: 0
Missing phenotypes: 0
Missing alleles: 4
Number of NA markers: 2
========================

> simulatedFam1[[1]]    # Alleles (genotypes) of the 1st simulated family

  famID indID patID matID sex phenot Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2
1 FAM01     1     0     0   1      0     T     T  <NA>  <NA>  <NA>  <NA>
2 FAM01     2     0     0   2      0     T     T     T     C     C     G
3 FAM01     3     1     2   2      0     T     T     C     T     G     C
4 FAM01     4     1     2   1      0     T     T     C     T     C     C

> simulatedFam1[[2]]    # Haplotypes of the 1st simulated family

  famID indID patID matID sex phenot Paternal_Hap Maternal_Hap
1 FAM01     1     0     0   1      0        T-C-G        T-C-C
2 FAM01     2     0     0   2      0        T-T-C        T-C-G
3 FAM01     3     1     2   2      0        T-C-G        T-T-C
4 FAM01     4     1     2   1      0        T-C-C        T-T-C
```

alleSimulator Example 2: *Simulation of a family containing offspring missing data.*

```
> simulatedFam2 <- alleSimulator(1,3,3,missOffProb=0.2,ungenotOffs=0.2)

==========================================
===== alleHap package: version 0.9.2 ======
==========================================

Data have been successfully loaded from:
/tmp/RtmpkiEe3V/Rbuild17985792d8e/alleHap/vignettes

===== DATA COUNTING ======
Number of families: 1
Number of individuals: 5
```

```
Number of founders: 2
Number of children: 3
Number of males: 3
Number of females: 2
Number of markers: 3
============================

======== DATA RANGES =========
Family ID: FAM01
Individual IDs: [1,...,5]
Paternal IDs: [0,1]
Maternal IDs: [0,2]
Sex values: [1,2]
Phenotype values: [0,1]
==============================

===== MISSING DATA =====
Missing founders: 0
Missing children: 0
Missing paternal IDs: 2
Missing maternal IDs: 2
Missing sex: 0
Missing phenotypes: 0
Missing alleles: 18
Number of NA markers: 3
========================

> simulatedFam2[[1]]    # Alleles (genotypes) of the 2nd simulated family

  famID indID patID matID sex phenot Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2
1 FAM01     1     0     0   1      0     T     T     C     C     C     C
2 FAM01     2     0     0   2      1     T     T     C     C     C     A
3 FAM01     3     1     2   1      0  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
4 FAM01     4     1     2   1      0  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
5 FAM01     5     1     2   2      1  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>

> simulatedFam2[[2]]    # Haplotypes of the 2nd simulated family

  famID indID patID matID sex phenot Paternal_Hap Maternal_Hap
1 FAM01     1     0     0   1      0        T-C-C        T-C-C
2 FAM01     2     0     0   2      1        T-C-C        T-C-A
3 FAM01     3     1     2   1      0        T-C-C        T-C-A
4 FAM01     4     1     2   1      0        T-C-C        T-C-C
5 FAM01     5     1     2   2      1        T-C-C        T-C-C
```

5 Workflow

The workflow of `alleHap` comprise mainly three stages: *Data Loading*, *Data Imputation* and *Data Phasing*. The next subsections will describe each of them.

5.1 Data Loading

The package can be used with either simulated or non-simulated databases, and the data may contain or not genetic missing information. As it has mentioned in the 3 section, the .ped file is the default input format of `alleHap`, and although its loading process is quite simple, it is important to note that a file containing **a large number of markers could slow down the process**. Furthermore, in order to avoid the foregoing, it is highly recommended that users *split the dataset into chromosomes*. Each data chunk should be later loaded by the `alleLoader` function.

`alleHap` has been tested with the Type 1 Diabetes genetics Consortium database [RCE+06]. This database consisted of over 3000 families and 20 markers (16 numeric and 4 character type: "A", "C", "G" or "T"). One example of a dataset with similar genetic information is the following one:

5.1.1 alleLoader Function

The `alleLoader` function tries to load the user dataset into a fully compatible format. In order to perform the above this function has been developed in four steps:

I. **Extention check and data read**: In this step the extension file is checked and if it has a .ped type the dataset is loaded into R as a data.frame. Should this not occur, the message *"The file must have a .ped extension"* is returned and the data will not be loaded.

II. **Data check**: The second step counts the number of families, individuals, parents, children, males, females and markers of the dataset, as well as, it checks the ranges of paternal IDs, maternal IDs, genotypes and phenotype values.

III. **Missing data check**: This step checks the unknown data and adjusts the genotype missing data by replacing the 0 and -9 values with NAs (Not Available values).

IV. **Function output**: In the final step, the dataset is exported as a data.frame and a summary of previous data counting, ranges and missing values is printed into the screen.

5.1.2 alleLoader Examples

Below it is depicted an example of how `alleLoader` should be used:

alleLoader Example 1: *Loading of a dataset in .ped format with alphabetical alleles (A,C,G,T)*

```
> example1 <- file.path(find.package("alleHap"), "examples", "example1.ped")
>
> example1Alls <- alleLoader(example1)  # Loaded alleles of the example 1

==========================================
===== alleHap package: version 0.9.2 ======
==========================================

Data have been successfully loaded from:
/tmp/RtmpkiEe3V/Rinst17983514122/alleHap/examples/example1.ped
```

```
===== DATA COUNTING ======
Number of families: 50
Number of individuals: 227
Number of founders: 100
Number of children: 127
Number of males: 118
Number of females: 109
Number of markers: 12
==========================

======== DATA RANGES =========
Family IDs: [1,...,50]
Individual IDs: [1,...,8]
Paternal IDs: [0,1]
Maternal IDs: [0,2]
Sex values: [1,2]
Phenotype values: [0,1]
==============================

===== MISSING DATA =====
Missing founders: 0
Missing children: 0
Missing paternal IDs: 100
Missing maternal IDs: 100
Missing sex: 0
Missing phenotypes: 0
Missing alleles: 0
Number of NA markers: 0
========================

> example1Alls[1:10,1:20]  # Alleles of the first 10 subjects

   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20
1   1  1  0  0  1  0  T  T  C   T   A   A   G   T   A   G   A   T   G   G
2   1  2  0  0  2  0  A  T  C   G   C   C   A   G   G   G   A   T   C   G
3   1  3  1  2  1  1  A  T  G   T   A   C   G   G   A   G   A   T   G   G
4   1  4  1  2  2  0  A  T  C   G   A   C   G   T   G   G   T   T   G   G
5   1  5  1  2  2  0  A  T  C   G   A   C   G   T   G   G   T   T   G   G
6   2  1  0  0  1  0  A  T  A   G   A   G   A   G   T   T   G   T   A   G
7   2  2  0  0  2  0  G  T  A   C   A   G   A   C   G   G   C   G   C   T
8   2  3  1  2  2  0  G  T  A   C   A   G   A   G   G   T   G   G   C   G
9   2  4  1  2  1  0  A  G  C   G   G   G   A   A   G   T   G   T   A   C
10  2  5  1  2  1  0  T  T  A   A   A   A   C   G   G   T   C   G   G   T
```

alleLoader Example 2: *Loading of a dataset in .ped format with numerical alleles*

```
> example2 <- file.path(find.package("alleHap"), "examples", "example2.ped")
>
> example2Alls <- alleLoader(example2)  # Loaded alleles of the example 2
```

```
==========================================
===== alleHap package: version 0.9.2 ======
==========================================

Data have been successfully loaded from:
/tmp/RtmpkiEe3V/Rinst17983514122/alleHap/examples/example2.ped

===== DATA COUNTING ======
Number of families: 11
Number of individuals: 50
Number of founders: 22
Number of children: 28
Number of males: 25
Number of females: 23
Number of markers: 3
============================

======== DATA RANGES =========
Family IDs: [1036,...,1939]
Individual IDs: [1,...,7]
Paternal IDs: [0,1]
Maternal IDs: [0,2,99]
Sex values: [female,male]
Phenotype values: [1,2]
==============================

===== MISSING DATA =====
Missing founders: 0
Missing children: 0
Missing paternal IDs: 22
Missing maternal IDs: 22
Missing sex: 2
Missing phenotypes: 0
Missing alleles: 42
Number of NA markers: 3
========================

> example2Alls[1:9,]  # Alleles of the first 9 subjects

    V1 V2 V3 V4     V5 V6  V7   V8  V9 V10 V11 V12
1 1036  1  0  0   male  1 101 1601 101 102 501 502
2 1036  2  0  0 female  1 301  401 301 501 201 301
3 1036  3  1  2   male  2 301 1601 102 501 201 502
4 1036  4  1  2   male  2 301 1601 102 501 201 502
5 1239  1  0  0   male  1  NA   NA  NA  NA  NA  NA
6 1239  2  0  0 female  1  NA   NA  NA  NA  NA  NA
7 1239  3  1  2 female  2 301  401 301 501 201 302
8 1239  4  1  2 female  2 301  401 301 501 201 302
9 1239  5  1  2   <NA>  1  NA   NA  NA  NA  NA  NA
```

5.2 Data Imputation

This part of the package imputes the previous simulated/loaded datasets by analyzing all possible combinations of a parent-offspring pedigree in which parental and/or offspring genotypes may be missing; as long as one child was genotyped, in certain cases it is possible an unequivocal imputation of missing genotypes both in parents and children.

5.2.1 alleImputer Function

`alleImputer` sorts the alleles of each family marker (when possible) and then imputes the missing values. In order to perform the data imputation, this function has been developed in six steps:

I. **Internal Functions**: In this step are loaded all the necessary functions to impute the data. The most important ones are: *mkrImputer* (which performs the imputation of a marker), *famImputer* (which imputes all the markers of a family) and *famsImputer* (which imputes all given families).

II. **Data Loading**: The second step tries to load user's data into a fully compatible format by means of the `alleLoader` function.

III. **Imputation**: This is the most important step of the `alleImputer` function. The imputation is performed marker by marker and then, family by family. The marker imputation is implemented by `mkrImputer` internal function which in two stages: children imputation and parent imputation. Given a marker with missing values, these can be imputed only either the genotypes of a parent and/or a child are homozygous. If in a marker, one parent has missing alleles and the other not, and the heterozygous alleles of children are not present in the complete parent, those alleles are imputed to the other parent.

IV. **Data Summary**: Once the imputation is done, a summary of the imputed data are collected.

V. **Data Storing**: In this step, the imputed data are stored in the same path where the PED file was located.

VI. **Function Output**: In this final step, a imputation summary may be printed out, if *dataSummary*=TRUE. Imputed data can be directly returned, whether *invisibleOutput* is deactivated. Incidence messages can be shown, if they are detected. These incidences can be: a) *"Some children have no common alleles with a parent"*, b) *"More alleles than possible in this marker"*, c) *"Some children have alleles not present in parents"*, d) *"Some homozygous children are not compatible in this marker"*, e) *"Three or more unique heterozygous children share the same allele"*, f1) *"Heterozygous parent and more than two unique homozygous children"*, f2) *"Heterozygous parent, four unique alleles and more than one unique homozygous children"*, f3) *"Homozygous parent and more than two unique children"*, g1) *"More than four unique children geneotypes in the family"* or g2) *"Homozygous genotypes and four unique alleles in children"*.

5.2.2 alleImputer Examples

Below are listed a couple of examples showing how `alleImputer` works:

alleImputer Example 1: *Allele imputation of families containing parental missing data.*

```
> ## Simulation of a family contatining parental missing data
> simulatedFam1 <- alleSimulator(1,2,3,missParProb=0.6,dataSummary=FALSE)
>
> simulatedFam1[[1]]  # Alleles of the simulated family
```

```
  famID indID patID matID sex phenot Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2
1 FAM01     1     0     0   1      0  <NA>  <NA>  <NA>  <NA>     A     G
2 FAM01     2     0     0   2      0     A     A  <NA>  <NA>     A     A
3 FAM01     3     1     2   2      0     A     A     T     C     A     G
4 FAM01     4     1     2   2      0     A     A     C     C     G     A
```

```
> ## Allele imputation of the previous family
> imputedFam1 <- alleImputer(simulatedFam1[[1]])

==========================================
===== alleHap package: version 0.9.2 ======
==========================================

Data have been successfully loaded from:
/tmp/RtmpkiEe3V/Rbuild17985792d8e/alleHap/vignettes

===== DATA COUNTING ======
Number of families: 1
Number of individuals: 4
Number of founders: 2
Number of children: 2
Number of males: 1
Number of females: 3
Number of markers: 3
============================

======== DATA RANGES =========
Family ID: FAM01
Individual IDs: [1,...,4]
Paternal IDs: [0,1]
Maternal IDs: [0,2]
Sex values: [1,2]
Phenotype values: [0]
==============================

===== MISSING DATA =====
Missing founders: 0
Missing children: 0
Missing paternal IDs: 2
Missing maternal IDs: 2
Missing sex: 0
Missing phenotypes: 0
Missing alleles: 6
Number of NA markers: 2
========================

Alleles have been successfully imputed!!!

===== IMPUTATION SUMMARY =====
Number of missing alleles: 6
Number of imputed alleles: 6
```

```
Imputation rate: 1
Imputation time: 0.01
==============================

> imputedFam1$imputedMkrs # Imputed alleles (markers)

  famID indID patID matID sex phenot Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2
1 FAM01     1     0     0   1      0  <NA>  <NA>  <NA>  <NA>     A     G
2 FAM01     2     0     0   2      0     A     A  <NA>  <NA>     A     A
3 FAM01     3     1     2   2      0     A     A     T     C     A     G
4 FAM01     4     1     2   2      0     A     A     C     C     G     A
```

alleImputer Example 2: *Allele imputation of families containing offspring missing data.*

```
> ## Simulation of two families containing offspring missing data
> simulatedFam2 <- alleSimulator(2,2,3,missOffProb=0.6,dataSummary=FALSE)
> simulatedFam2[[1]] # Alleles of the simulated families

  famID indID patID matID sex phenot Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2
1 FAM01     1     0     0   1      0     A     A     G     G     T     T
2 FAM01     2     0     0   2      0     A     A     G     G     T     C
3 FAM01     3     1     2   2      1  <NA>  <NA>     G     G     T     C
4 FAM01     4     1     2   2      0     A     A  <NA>  <NA>  <NA>  <NA>
5 FAM02     1     0     0   1      0     G     G     G     G     C     C
6 FAM02     2     0     0   2      0     A     A     A     A     C     C
7 FAM02     3     1     2   1      0  <NA>  <NA>     G     A  <NA>  <NA>
8 FAM02     4     1     2   1      0     G     A     G     A     C     C

> ## Allele imputation of the previous familes
> imputedFam2 <- alleImputer(simulatedFam2[[1]])

==========================================
===== alleHap package: version 0.9.2 ======
==========================================

Data have been successfully loaded from:
/tmp/RtmpkiEe3V/Rbuild17985792d8e/alleHap/vignettes

===== DATA COUNTING ======
Number of families: 2
Number of individuals: 8
Number of founders: 4
Number of children: 4
Number of males: 4
Number of females: 4
Number of markers: 3
============================

======== DATA RANGES =========
Family IDs: [FAM01,...,FAM02]
```

```
Individual IDs: [1,...,4]
Paternal IDs: [0,1]
Maternal IDs: [0,2]
Sex values: [1,2]
Phenotype values: [0,1]
==============================

===== MISSING DATA =====
Missing founders: 0
Missing children: 0
Missing paternal IDs: 4
Missing maternal IDs: 4
Missing sex: 0
Missing phenotypes: 0
Missing alleles: 10
Number of NA markers: 3
========================

Alleles have been successfully imputed!!!

===== IMPUTATION SUMMARY =====
Number of missing alleles: 10
Number of imputed alleles: 1
Imputation rate: 0.1
Imputation time: 0.01
==============================

> imputedFam2$imputedMkrs # Imputed alleles (markers)

  famID indID patID matID sex phenot Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2
1 FAM01     1     0     0   1      0     A     A     G     G     T     T
2 FAM01     2     0     0   2      0     A     A     G     G     T     C
3 FAM01     3     1     2   2      1     A     A     G     G     T     C
4 FAM01     4     1     2   2      0     A     A     G     G     T  <NA>
5 FAM02     1     0     0   1      0     G     G     G     G     C     C
6 FAM02     2     0     0   2      0     A     A     A     A     C     C
7 FAM02     3     1     2   1      0     G     A     G     A     C     C
8 FAM02     4     1     2   1      0     G     A     G     A     C     C
```

5.3 Data Phasing

At this stage, the corresponding haplotypes of the biallelic pedigree databases are generated. To accomplish this, based on the user's knowledge of the intended genomic region to analyse, it is necessary to **slice the data into non-recombinant chunks** in order to perform the haplotype reconstruction to each of them.

5.3.1 hapPhaser Function

`hapPhaser` creates the haplotypes family by family taking into account the previously imputed genotypes, along with the matrix IDS. In order to generate the haplotypes, this function has been developed

in six steps:

I. **Internal Functions**: In this step, numerous functions to reconstruct the haplotypes were implemented, being the most important the *famPhaser* function (which develops the haplotype phasing per family), *famsPhaser* (which reconstructs the haplotypes for multiple families) and *summarizeData* (which generates a summary of the phased data).

II. **Imputation**: This step calls the `alleImputer` function which performs the imputation marker by marker and then, family by family.

III. **Phasing**: This part is the most important of `hapPhaser`, since it tries to solve the haplotypes when possible. The process is the following: once each family genotype has been imputed marker by marker, those markers containing two unique heterozygous alleles (both in parents and offspring) are excluded from the process. Then, an IDentified/Sorted (IDS) matrix is generated per family. Later, the internal function `famPhaser` tries to solve the haplotypes of each family, comparing the information between parents and children in an iterative and reciprocal way. When there are not genetic data in both parents and there are two or more "unique" offspring (not twins or triplets), the internal functions `solveHapsWithMissingParents` and `phaseHapsFromTwoChildren` try to solve the remaining data. Finally, the HoMoZygosity (HMZ) matrix is updated and the excluded markers are again included. *Even if both parental alleles are missing in each marker, it is possible to reconstruct the family haplotypes, identifying the corresponding children's haplotypes, although* ***in certain cases their parental provenance will be unknown****.*

IV. **Data Summary**: Once the data phasing is done, a summary of the phased data is collected.

V. **Data Storing**: In this step, the imputed data are stored in the same path where the PED file was located.

VI. **Function Output**: In this final step, a summary of the phased data may be printed out, if *dataSummary*=TRUE. All the results can be directly returned, whether *invisibleOutput* is deactivated. Incidence messages can also be shown, if they are detected. These may be caused by haplotype recombination on children, genotyping errors or inheritance from non-declared parents. The messages shown in such cases are: *"Irregular inheritance detected"*, *"Multiple compatible parental haplotypes"*, *"Parental information is not compatible with haplotypes found in children"*, *Haplotypes in one child are not compatible with the haplotypes found in the rest of the offspring*, or *"Less than two children detected"*.

The final output the `hapPhaser` is a list comprised by five elements: *imputedMkrs* (which contains the imputed markers), *imputationSummary* (which shows a summary of the imputation process), *phasedMkrs* (which includes the phased alleles for all markers) and *haplotypes* (which stores the phased haplotypes, the number of non-phased alleles, the full phased haplotypes and IDentified/Sorted individuals) and *phasingSummary* (which shows a summary of the phasing process).

5.3.2 hapPhaser Examples

Below are listed some examples showing how `hapPhaser` works:

hapPhaser Example 1: *Haplotype reconstruction of a dataset containing parental missing data.*

```
> ## Simulation of familes containing parental missing data
> simulatedFams1 <- alleSimulator(2,2,6,missParProb=0.2,
+                                 ungenotPars=0.4,dataSummary=FALSE)
```

```
> ## Reconstruction of previous simulated families
> phasedFams1 <- hapPhaser(simulatedFams1[[1]])

==========================================
===== alleHap package: version 0.9.2 ======
==========================================

Data have been successfully loaded from:
/tmp/RtmpkiEe3V/Rbuild17985792d8e/alleHap/vignettes

===== DATA COUNTING ======
Number of families: 2
Number of individuals: 8
Number of founders: 4
Number of children: 4
Number of males: 3
Number of females: 5
Number of markers: 6
============================

======== DATA RANGES =========
Family IDs: [FAM01,...,FAM02]
Individual IDs: [1,...,4]
Paternal IDs: [0,1]
Maternal IDs: [0,2]
Sex values: [1,2]
Phenotype values: [0,1]
==============================

===== MISSING DATA =====
Missing founders: 0
Missing children: 0
Missing paternal IDs: 4
Missing maternal IDs: 4
Missing sex: 0
Missing phenotypes: 0
Missing alleles: 16
Number of NA markers: 6
========================

Alleles have been successfully imputed!!!

===== IMPUTATION SUMMARY =====
Number of missing alleles: 16
Number of imputed alleles: 16
Imputation rate: 1
Imputation time: 0.02
==============================


Haplotypes have been successfully phased!!!
```

```
========== PHASING SUMMARY ============
Proportion of phased alleles: 0.9271
Proportion of non-phased alleles: 0.0729
Proportion of missing haplotypes: 0
Proportion of partial haplotypes: 0.1875
Proportion of full haplotypes: 0.8125
Phasing time: 0.018
=======================================

> simulatedFams1[[1]][,-(1:6)] # Simulated alleles

  Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2 Mk4_1 Mk4_2 Mk5_1 Mk5_2 Mk6_1 Mk6_2
1     T     T     C     C  <NA>  <NA>     T     T     C     C     C     C
2     T     T     T     T     T     C     C     C     C     C     C     C
3     T     T     C     T     C     C     T     C     C     C     C     C
4     T     T     C     T     C     T     T     C     C     C     C     C
5     T     T  <NA>  <NA>     C     C  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
6  <NA>  <NA>  <NA>  <NA>     C     T  <NA>  <NA>     C     C     A     A
7     T     T     C     C     C     T     C     C     C     C     A     A
8     T     T     C     C     C     C     C     C     C     C     A     A

> phasedFams1$phasedMkrs[,-(1:6)] # Imputed/Phased markers

  Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2 Mk4_1 Mk4_2 Mk5_1 Mk5_2 Mk6_1 Mk6_2
1     T     T     C     C     C  <NA>     T     T     C     C     C     C
2     T     T     T     T     C     T     C     C     C     C     C     C
3     T     T     C     T     C     C     T     C     C     C     C     C
4     T     T     C     T     C     T     T     C     C     C     C     C
5     T     T     C  <NA>     C     C     C  <NA>     C  <NA>     A  <NA>
6     T     T     C     C     T     C     C     C     C     C     A     A
7     T     T     C     C     C     T     C     C     C     C     A     A
8     T     T     C     C     C     C     C     C     C     C     A     A

> phasedFams1$haplotypes # Phased haplotypes

    hap1   hap2 nonAlls fullHaps IDSindiv
1 TCCTCC TC?TCC       1        1        0
2 TTCCCC TTTCCC       0        2        0
3 TCCTCC TTCCCC       0        2        1
4 TC?TCC TT?CCC       2        1        0
5 TCCCCA T?C???       4        1        0
6 TCTCCA TCCCCA       0        2        0
7 TCCCCA TCTCCA       0        2        1
8 TCCCCA TCCCCA       0        2        1
```

hapPhaser Example 2: *Haplotype reconstruction of a dataset containing offspring missing data.*

```
> ## Simulation of families containing offspring missing data
> simulatedFams2 <- alleSimulator(2,2,6,missOffProb=0.4,
+                                 ungenotOffs=0.2,dataSummary=FALSE)
```

```
> simulatedFams2[[1]][,-(1:6)] # Simulated alleles

  Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2 Mk4_1 Mk4_2 Mk5_1 Mk5_2 Mk6_1 Mk6_2
1     A     G     A     A     A     G     C     C     C     C     T     C
2     A     G     A     A     A     G     C     C     C     C     T     C
3     A     G  <NA>  <NA>     A     G     C     C     C     C  <NA>  <NA>
4  <NA>  <NA>     A     A  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>     C     C
5     A     A     G     G     G     A     A     A     C     C     T     T
6     A     A     G     G     G     A     A     A     C     C     T     T
7     A     A  <NA>  <NA>     G     G  <NA>  <NA>     C     C     T     T
8     A     A  <NA>  <NA>  <NA>  <NA>     A     A     C     C  <NA>  <NA>

> ## Reconstruction of previous simulated families
> phasedFams2 <- hapPhaser(simulatedFams2[[1]],dataSummary=FALSE)
> phasedFams2$phasedMkrs[,-(1:6)] # Imputed/Phased markers

  Mk1_1 Mk1_2 Mk2_1 Mk2_2 Mk3_1 Mk3_2 Mk4_1 Mk4_2 Mk5_1 Mk5_2 Mk6_1 Mk6_2
1     A     G     A     A     A     G     C     C     C     C     C     T
2     A     G     A     A     A     G     C     C     C     C     C     T
3     A     G     A     A     A     G     C     C     C     C  <NA>  <NA>
4  <NA>  <NA>     A     A  <NA>  <NA>     C     C     C     C     C     C
5     A     A     G     G     G     A     A     A     C     C     T     T
6     A     A     G     G     G     A     A     A     C     C     T     T
7     A     A     G     G     G     G     A     A     C     C     T     T
8     A     A     G     G  <NA>  <NA>     A     A     C     C     T     T

> phasedFams2$haplotypes # Phased haplotypes

    hap1   hap2 nonAlls fullHaps IDSindiv
1 ?A?CCC ?A?CCT       4        1        0
2 ?A?CCC ?A?CCT       4        1        0
3 ?A?CC? ?A?CC?       6        1        0
4 ?A?CCC ?A?CCC       4        1        1
5 AGGACT AGAACT       0        2        0
6 AGGACT AGAACT       0        2        0
7 AGGACT AGGACT       0        2        1
8 AG?ACT AG?ACT       2        1        0
```

References

[BB09] Brian L Browning and Sharon R Browning. A unified approach to genotype imputation and haplotype-phase inference for large data sets of trios and unrelated individuals. *The American Journal of Human Genetics*, 84(2):210–223, 2009.

[BB11] Sharon R Browning and Brian L Browning. Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, 12(10):703–714, 2011.

[BWSD+07] Tanya Y Berger-Wolf, Saad I Sheikh, Bhaskar DasGupta, Mary V Ashley, Isabel C Caballero, Wanpracha Chaovalitwongse, and S Lahari Putrevu. Reconstructing sibling relationships in wild populations. *Bioinformatics*, 23(13):49–56, 2007.

[CMJVC+10] EC Castelli, CT Mendes-Junior, LC Veiga-Castelli, NF Pereira, ML Petzl-Erler, and EA Donadi. Evaluation of computational methods for the reconstruction of hla haplotypes. *Tissue Antigens*, 76(6):459–466, 2010.

[dBMS+06] Paul IW de Bakker, Gil McVean, Pardis C Sabeti, Marcos M Miretti, Todd Green, Jonathan Marchini, Xiayi Ke, Alienke J Monsuur, Pamela Whittaker, Marcos Delgado, et al. A high-resolution hla and snp haplotype map for disease association studies in the extended human mhc. *Nature genetics*, 38(10):1166–1172, 2006.

[MCH+13] S. J. Mack, P. Cano, J. A. Hollenbach, J. He, C. K. Hurley, D. Middleton, M. E. Moraes, S. E. Pereira, J. H. Kempenich, E. F. Reed, M. Setterholm, A. G. Smith, M. G. Tilanus, M. Torres, M. D. Varney, C. E. M. Voorter, G. F. Fischer, K. Fleischhauer, D. Goodridge, W. Klitz, A.-M. Little, M. Maiers, S. G. E. Marsh, C. R. Mller, H. Noreen, E. H. Rozemuller, A. Sanchez-Mazas, D. Senitzer, E. Trachtenberg, and Marcelo Fernandez-Vina. Common and well-documented hla alleles: 2012 update to the cwd catalogue. *Tissue Antigens*, 81(4):194–203, 2013.

[PNTB+07] Shaun Purcell, Benjamin Neale, Kathe Todd-Brown, Lori Thomas, Manuel AR Ferreira, David Bender, Julian Maller, Pamela Sklar, Paul IW De Bakker, Mark J Daly, et al. Plink: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*, 81(3):559–575, 2007.

[RCE+06] Stephen S Rich, Patrick Concannon, Henry Erlich, Cecile Julier, Grant Morahan, Jorn Nerup, Flemming Pociot, and John A Todd. The type 1 diabetes genetics consortium. *Annals of the New York Academy of Sciences*, 1079(1):1–8, 2006.