Manual of HaploPOP

Nicolas Duforet-Frebourg
Université Joseph Fourier,
Centre National de la Recherche Scientifique,
Laboratoire TIMC-IMAG, Grenoble, France.

February 18, 2015

Contents

1 Introduction

HaploPOP is a software that builds informative haplotypes to characterize population genetic structure based on the Gain of Informativeness for Assignment (GIA) statistic, introduced by Gattepaille and Jakobsson [?]. The method implemented by the software is described in [?].

2 Algorithms

2.1 Aim

We mentionned in the article our will to transform a set of markers in a set of haplotypes being much more informative. The algorithm will be optimization algorithm for the following problem: Consider a set of L Markers $\{S_1, S_2...S_n\}$. Denote that what we would call markers in this manual would refer to any genetic segregating site, such as Single Nucleotide Polymorphisms, Haplotypes, or Microsatellites. Our problem can be formally written as:

$$\begin{cases} argMax_{\Gamma} \sum_{H \in \Gamma} IA(H) \\ \Gamma \in Part(S_1...S_L) \end{cases}$$

For the commodity we show that this score can be express as a function of the Gain of Informativness for Assignment only:

$$\Leftrightarrow \begin{cases} argMax_{\Gamma} \sum_{H \in \Gamma} (\sum_{S \in H} GIA(H_{1..S-1}, S)) \\ \Gamma \in Part(S_1...S_L) \end{cases}$$

We can observe that this problem has no optimal structure. If you find the best combination into a certain number of haplotype, the best solution at next step, with a new haplotype, can not be predictied using the previous best solution. Because of its topology, a lot of method useful in combinatorial problems are not applicable, especially the dynamic programming. We have then to find other solution to compute efficient combinations.

2.2 LEARN

To compute naive combinations when you don't have access to the optimal solution, the greedy algorithm worths to be tried. Though we know that this algorithm, according to certain problem can be performing arbitrarily good or inefficient. In our case, this is doing constantly, and not so surprisingly, good. We introduce here the object we would call the matrix of GIAs, which is initially a $L \times L$ matrix, which is reduced by one dimension at each step:

$$\begin{pmatrix} 0 & GIA(1,2) & GIA(1,3) & ... \\ GIA(2,1) & 0 & GIA(2,3) & ... \\ ... & & & \end{pmatrix}.$$

The greedy algorithm gives good results, however, It produces haplotypes that can group markers very far away from each over. To save some overfit, we force the combination to be more local. In addition, Greedy is a quite expensive algorithm in terms of number of operation. We give an estimation of the cost of such an algorithm as a function of the size of the data. Let's say we are working with N_i chromosomes from haploid individuals. If we consider a number L of markers

$$C_{L,N_i} \approx O(a \times L^3 + b \times L \times N_i + c \times \frac{L^2}{2} \times N_i), \text{ where } a, b, c \in \mathbb{R}_+$$

.

From this Cost, we note that we can save a lot of time, by dividing the Set of Markers into Subsets of smaller sizes. That is what the `Subsets` method does. We target smaller Set of markers, and apply them the greedy algorithm.

$$\{ \underbrace{S_1, ...S_w}_{N_1 Haplotypes}, \underbrace{S_{w+1}, ...S_{2w}}_{N_2 Haplotypes}, ..., \underbrace{S_{F(\frac{L}{w}) \times w+1}, ...S_L}_{N_{F(\frac{L}{w})} Haplotypes} \}$$

The algorithm can be written like this:

```
divide the Set in Subsets
for each Subset do{
    Calculate GIA for every potential combination in GIAMatrix
    while max(GIAMatrix > 0){
        Combine argmax(GIAMatrix) in Data
        Update the Data and the GIAMatrix with the new combination
    }
}
Write the haplotypes of every Subset in the OUTPUTFile
```

As a result, the computation time will be highly reduced, but also the haplotype will be built with a lower "choice" of combination. Then we expect the haplotype marker set resulting to be of a lower quality than the one produced by the greedy. The point of this method is to offer the choice to the user to have a trade off between time and efficiency. We will mention more about this in the note about the performances.

To select the trade off between time and combinations the size of the Subsets can be chosen by two ways:

Fixed Number of marker One can specify a particular number of marker to include in each window. This is the case that has described given above.

Physical distance One can also specify the size as a particular physical distance between markers. It can be a little bit more relevant to consider windows of a chosen constant physical size, rather than a constant number a markers that can be spread on very large distances. By using physical or genetic distances, one take advantage of linkage disequilibrium induced by recombination rate.

2.3 APPLY

HaploPOP is a software made to build haplotypes starting with an initial set of markers. It can be used for its statistical inference of haplotypes, but it is also possible to specify the particular haplotypes one might want to build. This method would write into the `OUTPUTFile` the Combinations of the `INPUTFile` according to the `HAPLOFile` pattern that must be specified here.

This method is particularly useful for study such as case-control study, when sets of individuals are available to learn haplotypes, to then assign individuals to groups. It is also a good feature for split, or Cross validation, when investigating population structure, such as mentionned in [**?**]duforetetal). One just have to use the `.haploID` produced by the previous method, and the `APPLY` method on your validation set.

3 Starters

3.1 windows OS

If you are using windows, you can use the software in two ways. The first one is the command line software. You will have to open a terminal first (`run`, then type `cmd`). Then go in the repertory containing the executable file `HaploPOP.exe`, and just type `HaploPOP.exe` with the parameters of your choice.

If you rather use a more guided execution, you can just run the other executable file `HaploPOPUI.exe`. Parameters will be asked step by step. It is more convenient if your are not familiar with the handling of the parameters.

Note that if you are running the software with windows, and with too large data, an error might occur. It is due to the management of the stack with this OS.

3.2 UNIX OS

Extraction and Compilation The archive of the program is provided with a `Makefile` for `UNIX` OS. Thus Compilation is an easy step done like this:

```
MyMachine $> make
```

If ever, for some reasons, you want to clean the repertory of all executable or binary files, just type:

```
MyMachine $> make clean
```

Say hello Once the program compiled, you are ready to run it. You can do it one first time without parameters, and a Presentation screen will be displayed. Then the software is run as other usual software for LINUX.

3.3 MAC OS

The software has been initially developped for UNIX type of Operating system. It is running fine with MAC OS.

4 Command line

Here is a complete list of the parameters of the program, and their meaning. When a parameter can be unspecified, it is explicitely mentionned. The two command lines to run the software are the following one:

4.1 LEARN

```
 MyMachine $> ./HaploPOP LEARN -i npop File_pop1 File_pop2... File_popn
 -o OUTPUTFile -p LOCUSPOSITIONSFile (-t Threshold)
```

-i npop file$_{pop1}$ file$_{pop2}$ ... file$_{popn}$: The number and the path/names of the files containing the genotypes. Each population should be in a different file.

-o output : The name to specify will be used for the different output files produced, such as output.haploID, output.haploIA...

-s size : The size parameter indicates the window size. It can be an integer, as a number of SNP, or a real if you use physical or genetic distances. In this case the -p option is compulsory.

-p LOCUSPOSITIONFile This file must be specified only if you use the subsets based on loci positions. It contains the physical position of every markers, by increasing order.

-t Threshold An optionnal float, if you want to build only the more informative haplotypes.

4.2 APPLY

```
 MyMachine $> ./HaploPOP APPLY -i npop File_pop1 File_pop2... File_popn
-o OUTPUTFile  -h HAPLOFile
```

-i npop file$_{pop1}$ file$_{pop2}$ ... file$_{popn}$: The number and the path/names of the files containing the genotypes. Each population should be in a different file.

-o output : The name to specify will be used for the different output files produced, such as output.haploID, output.haploIA...

-h HAPLOFile this parameter is the name you want to give to the file containing the haplotype ID of each initial genetic marker.

5 Files

5.1 input files

Data files The data files are your raw data file. The standard format consists in a text file with one individual per row. On each and every row would be written the codes of the alleles of the individual for each marker, so integers representing a certain SNP, halpotype, or microsatellite allele. There must be one file per population.

Here is an example of input file for 10 individuals, with 16 genetic markers. One should note that the integers representing the alleles of one marker (column), does not have to be consecutive.

1	0	1	0	0	0	1	0	1	0	2	1	2	4	1	0
0	0	0	0	2	0	2	2	2	2	0	1	0	2	1	1
0	0	1	0	1	0	1	0	1	2	2	1	0	7	1	0
0	1	0	1	0	1	2	1	2	0	2	1	2	5	1	0
0	1	0	1	2	0	2	1	2	0	2	1	0	1	1	1
0	1	0	1	0	1	2	1	2	0	2	1	2	5	1	0
0	0	1	0	1	0	1	0	1	2	2	1	2	98	1	0
0	0	0	0	1	1	2	0	1	2	0	1	2	56	1	0
1	0	0	0	1	0	2	0	1	2	2	1	0	71	1	0
1	0	1	0	2	0	2	2	2	2	0	1	0	2	1	1

This input file can also be a tped file. In this case, an intelligent extraction of the data will be done if you add the parameter -f, as mentionned above.

Haplotype file The Haplotype file is an input file in only one case, when the method chosen is `APPLY`. In this case, you choose to specify your own haplotypes to be built. The combinations will be read in this particular file. It must contain on one row an integer for each marker indicating the haplotype in which belong the marker.

For example, if you want to combine on the previous data the markers 2, 3, 4, and the markers 1, 12, 13, 14, 15, 16 together, we would use the `APPLY` method with the following haplotype file:

1	2	2	2	3	4	5	6	7	8	9	1	1	1	1	1

Locus Position file This file is useful only if you chose to apply the `SubsetLP` method. Then this file must contain all the informations needed to build the subsets. All the physical positions must be specified on one row and sorted by increasing order.

For instance, you can have the following Position file for the example above:

0.23	0.24	0.26	0.32	0.33	0.34	0.92	0.96	0.97	0.98	0.99	1.02	1.04	1.05	1.06	1.2

Such a position file puts in highlight a gap between markers 6 and 7. Then we could expect the `SubsetLP` method to be more relevant than the simple `Subset` method.

5.2 Output files

.haploID We mentionned this file as an input file, but, depending on the method, it is also an output file. Actually in most of the methods (all of them but `GivenHaplotype`) this file will be written. It is a one row file with as many integers as markers in the initial dataset. Each integer specifies the haplotype built with the marker at this position. The i^{th} integer with value K indicates that the i^{th} marker is in the Haplotype K. This format is very handy for further analysis.

.HaploIA This file indicates the $InformativnessforAssignment$ statistic for each haplotype that has been built.

.initMarkersIA This file indicates the $InformativenessforAssignment$ statistic for each marker initially present in the data set.

.struct_haps The `.struct_haps` file is a *structure* input type of file [**?**], with the haplotypes applied on the input data.

5.3 A toy example for input and output files.

Let's consider a population input file with 4 individuals and 6 genetic markers as follows:

1	0	1	0	0	0
0	0	0	0	2	0
0	0	1	0	1	0
0	1	0	1	0	1

Let's say that after combinations markers 1, 2 and 4 needs to be combined together. The resulting haplotypes corresponds for the 4 individuals to $100, 000, 000$ and 011.These 3 haplotypes are noted alleles $0, 1$ and 2. Assuming there is no other combinations, the resulting **.HaploIA** file is going to be the IA of this Haplotype, and the IA of the SNPs $3, 5$ and 6. The **.struct_haps** file si going to be:

0	1	0	0
1	0	2	0
1	1	1	0
2	0	0	1

The **HaploID** file will contain the resulting haplotype that contains each of the initial marker:

$$\boxed{1 \quad 1 \quad 2 \quad 1 \quad 3 \quad 4}$$

To perform a PCA on the results, one can use the **struct_haps** file and turn it into a presence absence matrix for each haplotype-allele. For example:

$$\boxed{\begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ \underbrace{0}_{} & \underbrace{0}_{} & \underbrace{1}_{} & 0 & 0 & 1 \end{array}}$$

6 Examples

6.1 Learn Haplotypes on two populations with Physical distances

We give an example of application of the software on some simulated data. The data, in the files `dat/pop1.dat` and `dat/pop2.dat` consist in a 2-island model simulated according to classic coalescent, with `ms` [?]. In these data, 200 chromosomes are typed on 1000 segregating sites. The 100 first chromosomes typed are from the first population, 100 others are from the second. In addition, we have access to the physical positions of these segregating sites in the file `LP/LPManual.txt`. These position are spanned between 0 and 1. One should then pick a size according to this scale. We would chose 0.5. The command would be as follow:

```
MyMachine $> ./HaploPOP LEARN -i 2 dat/pop1.dat dat/pop2.dat -o dat/Example2pops
                -s 0.5 -p dat/LP/LPManual.txt
```

6.2 Apply the Haplotypes on the 2-island data

In this last example we show how to learn the haplotypes. We kept the pattern of the haplotypes in dat/2popsExample.haploID, now this file will become an input file. The new data set will be written in Example2pops_haplo. We compute the haplotypes as follows:

```
MyMachine $> ./HaploPOP APPLY -i 2 dat/pop1.dat dat/pop2.dat -o dat/Example2pops
                 -h dat/Example2pops.haploID
```

Note that usually you do not want the haplotypes to be learn and applied on the same sets of individuals, because of overfitting.