

HapBlock – A Suite of Dynamic Programming Algorithms for Haplotype Block Partitioning and Tag SNP Selection Based on Haplotype and Genotype Data

Introduction

The suite of programs, HapBlock, is developed for haplotype block partitioning and tag SNP selection under the joint guidance of Ting Chen, Fengzhu Sun, and Michael Waterman within the Center for Computational and Experimental Genomics at the University of Southern California and with collaboration with Zhaohui Qin and Jun Liu in the department of statistics at Harvard University. This suite of programs is implemented and maintained by Dr. Kui Zhang, a former research associate in the center. The partition-ligation-expectation-maximization (PL-EM) algorithm for haplotype inference incorporated in this program is implemented by Dr. Zhaohui Qin, a former research associate in Jun Liu's lab at Harvard University.

The primary objective of this program is to find a set of tag SNPs used in association studies, which is achieved by a suite of dynamic programming algorithms. Our program is quite flexible to incorporate different definitions of haplotype block and different criteria for tag SNP selections for different applications. It also provides an interface that can take the output from other methods for haplotype block partitioning and tag SNP selection as input. Both haplotype data and genotype data from unrelated individuals as well as from general pedigrees can be analyzed by this program. For detailed algorithms and their applications, please refer to Zhang et al. (2002; 2003; 2004a).

Compile the Program

To run HapBlock, you need a compiled copy of the program, at least one input data file and one output file. This program is written by standard C++, so you may compile and run it under Windows 95, 98 or NT operating systems. You can also compile it and execute it under UNIX systems and other operating systems. In this web site, you can only download the executable code for Windows Operation System and Unix System. The source code of the programs may be available upon request.

Execute the Program

The parameters used in the program can be input either by a file, referred as parameter file in the context, or by an interactive menu. In the first way, you may type the following command:

```
HapBlock setpar.dat
```

Here, all necessary parameters are specified in this parameter file - setpar.dat. On running the program by the second way, you need only type the name of executable code:

HapBlock

An interactive menu will appear on the screen and guide you to set the parameters that will be used for haplotype block partitioning and tag SNP selection. We recommend you to run the program using the first way, especially when you need to execute the program many times. Because the parameter file can be easily generated and modified for different settings of parameters and it can be run by the batch command. In this help file, we will illustrate how to generate the parameter file through an example. The use of interactive menu is very similar and quite straightforward once you know how to set a parameter file.

Parameter File

Here is an example of the parameter file:

```
100 //Line 1: the maximum number of samples
1800 //Line 2: the maximum length of haplotypes (alternatively, the number of SNPs)
200 //Line 3: the maximum number of SNPs that a haplotype block can be extended
1 50 snp_pos.dat //Line 4: the algorithms for block partitioning
2 hap.dat block_out.dat //Line 5: the input data, the names of input and output file
1 0.80 0.05 //Line 6: methods for block partitioning
2 0.05 //Line 7: methods for tag SNP selection
1 presnp.dat //Line 8: if there are some pre-specified SNPs included in the analysis
1 1 1 snp_name.dat //Line 9: options for the output file
1 hap_pattern.dat //Line 10: option for the haplotype output
1 100 permute_stat.dat //Line 11: option for the permutation test
```

The parameter file must contain exactly 11 lines. In each line, up to three parameters are specified and followed by a brief explanation of these parameters. The explanation is optional and separated by two back slashes from the parameters. The details for setting those parameters one line by one line are listed as follows.

The first three lines

The three global parameters: the maximum number of samples, the maximum length of haplotype or the maximum number of SNPs, and the maximum number of markers that a haplotype block can be extended, are specified one by one in the first three lines. Each line contains exactly one parameter. They are used for memory allocation in the program. Therefore, all of them should be positive integers. The maximum number of samples is at least equal to the number of haplotypes for the haplotype data or the number of individuals for the genotype data. The maximum number of SNPs is at least equal to the number of SNPs used in the analysis. The maximum number of SNPs that a block can be extended can be equal to the maximum number of SNPs when the maximum number of SNP is not large. It can be set as several hundred if the maximum number of SNPs is very large. As our experience, a number in 200-300 is large enough for most studies.

The dynamic programming algorithms for haplotype block partitioning

The programs depend on two main criteria: the definition of candidate haplotype blocks and a criterion for finding tag SNPs within blocks. Given the definitions of haplotype blocks and tag SNPs, the dynamic algorithm for haplotype block partitioning is determined in the fourth line. Up to three parameters can be specified in this line, depending on the setting of dynamic programming algorithms. The first parameter is 1, 2 or 3, corresponding to different dynamic programming algorithms. Option 1 corresponds to the dynamic programming algorithm of Zhang et al. (2002) to minimize the total number of tag SNPs along the whole genome or a region of interest. Zhang et al. (2003) also proposed two dynamic programming algorithms to partition haplotypes into blocks with limited resources. With limited resources, we want to find a block partition with a given number of tag SNPs to maximize the length of genome covered by the tag SNPs. Specifically, the haplotypes are divided into a set of disjoint blocks, B_1, \dots, B_I , which are interspersed by a set of excluded intervals, E_1, \dots, E_J . Given a length function $L(i, \dots, j)$ for set of consecutive SNPs and a function $f(\cdot)$ for the number of tag SNPs in a block, we want to maximize $\sum_{i=1}^I L(B_i) / (\sum_{i=1}^I L(B_i) + \sum_{j=1}^J L(E_j)) = \sum_{i=1}^I L(B_i) / L(1, \dots, n)$ with the constraint $f(B) = \sum_{i=1}^I f(B_i) \leq m$, where m is the number of SNPs that can be genotyped. The haplotype block partitioning with limited resources can be formulated as two equivalent, dual problems: **Block Partition with a fixed Genome Coverage (FGC)** and **Block Partition with a fixed Number of Tag SNPs (FTSNP)**.

Given a set of haplotypes consisting of n SNPs and a parameter $0 \leq \mu \leq 1$ in **FGC**, we try to find a set of disjoint blocks, B_1, \dots, B_I with $\sum_{i=1}^I L(B_i) \geq \mu L(1, \dots, n)$ such that $\sum_{i=1}^I f(B_i)$ is minimized. A parametric dynamic programming algorithm is proposed to solve **FGC** problem in Zhang et al. (2003). It is worth noting that $f(B)$ is the same as the number obtained by a dynamic programming algorithm (Zhang et al., 2002) when μ is set as 1. Option 2 corresponds to the parametric dynamic programming algorithm to solve the **FGC** problem. We also need to set two other parameters. The second parameter is the fraction of the genome covered by blocks, μ , a positive number less than 1, and the third parameter is the name of a map file that contains the position of each SNP. These positions are used to calculate the length of a set of consecutive SNPs, $L(i, \dots, j)$ ($i \leq j$). For the contents and format of this file, please refer to the section "Input Files".

Given a set of haplotypes consisting of n SNPs and a positive integer m in **FTSNP**, we try to find a set of disjoint blocks, B_1, \dots, B_I with $f(B) \leq m$ such that $L(B)$ is maximized. Zhang et al. (2003) developed a two-dimension dynamic programming algorithm to solve the **FTSNP** problem. Option 3 corresponds to this algorithm. We also need two other parameters. The second parameter is the total number of tag SNPs that can be genotyped, m , a positive integer. m should be less than the total number of tag SNPs obtained by a dynamic programming algorithm when option 1 is used (Zhang et al., 2002). The third parameter is the name of a map file that contains the position of each SNP. These

positions are used to calculate the length of a set of consecutive SNPs, $L(i, \dots, j)$ ($i \leq j$). For the contents and format of this file, please refer to the section “Input Files”.

The calculation of the length for a set of consecutive SNPs, $L(\cdot)$, is based on the position of each SNP. Suppose the positions of a set of SNPs have been obtained from the map file, which will be described in section “Input Files”. Their positions are denoted as l_i ($1 \leq i \leq n$) and a special starting position l_0 is defined as $l_1 - (l_n - l_1)/(n-1)$. As a map, l satisfy $l_i \leq l_j$ for $0 \leq i \leq j \leq n$. We can define $L(\cdot)$ by the following formula:

$$L(i, \dots, j) = l_j - l_{i-1} \quad (1 \leq i \leq j \leq n).$$

Actually, l_i ($1 \leq i \leq n$) could be the position in a physical map, in a genetic map or just the index of this SNP. If l is defined as the index of the SNPs, that is $l_i = i$ ($1 \leq i \leq n$), then $L(i, \dots, j) = j - i + 1$ ($1 \leq i \leq j \leq n$), which is the number of SNPs in this set. For more flexible definitions of l , please refer to Zhang et al. (2003).

The type of input data

Three parameters, the type of input data, the name of input data file, and the name of output file, are set one by one in the fifth line, respectively. The range of the first parameter is from 1 to 5, corresponding to one of five different kinds of data that are supported by the program: 1--haplotype data, 2--genotype data from unrelated individuals, 3--genotype data from unrelated individuals, 4--genotype data from general pedigrees, and 5--all the possible blocks and their corresponding number of tag SNPs,.

Our algorithm can analyze genotype data in two different ways. When genotype data from unrelated individuals are available, the haplotype frequencies are calculated and the most likely pairs for each individual are assigned using the PL-EM algorithm (Qin et al., 2002). In this process, the PL-EM algorithm is performed for each set of consecutive SNPs that can form a potential block other than for the whole set of SNPs. Both option 2 and option 3 set the same type of genotype data. By selecting option 2, the most-likely haplotype pairs identified from the PL-EM algorithm will be used in block partitioning. By selecting option 3, the haplotypes and their frequencies estimated from the PL-EM algorithm will be employed in block partitioning. For details of the algorithm, please refer to Zhang et al. (2004a).

You can specify the first parameter as 4 in the fifth line to take the genotype data from general pedigrees as input. In the program, we assume that there is no recombination within each block and the haplotypes and their frequencies are inferred by logic-rules and PL-EM algorithm (Zhang et al., 2004b). The inference is carried out for each subset of consecutive SNPs that can form a potential block other than for the whole set of SNPs. It is also worth noting that the haplotype inference from pedigrees can be very time consuming, especially in the presence of missing data.

The possible blocks and their corresponding number tag SNPs as input provides us a flexible way to partition haplotypes into blocks using dynamic programming algorithms.

They could be blocks and number of tag SNPs obtained by the previous execution of this program. In general, it is a very time-consuming procedure to figure out all possible blocks and the corresponding number of tag SNPs. This input can save a lot of time in executing the program again. The possible blocks and their number of tag SNPs can also be obtained by an external program, in which the other different methods for haplotype block partitioning and tag SNP selection are utilized but haven't been incorporated in our program. Thus, the option 5 provides an interface between the other methods for haplotype block partitioning and our dynamic programming algorithms. Furthermore, if all the possible blocks and their number tag of SNPs are as input for the program, the parameters in the rest of lines will no longer take effect.

For the contents and format of these data files and the output file, please refer to sections "Input Files" and "Output Files".

The definition of haplotype blocks

The methods for defining haplotype blocks are specified in the sixth line. We are planning to implement many other haplotype block definitions in the future. Currently, the definitions based on common haplotypes (Patil et al., 2001; Zhang et al., 2002), the LD measure D' (Gabriel et al., 2002) and the four-gamete test (Wang et al., 2002) are available, which could be achieved by setting the first parameter in this line to 1, 2 and 3, respectively.

1 - Common Haplotypes. Here, a set of consecutive SNPs with size one or more forms a block if the number of common haplotypes account for at least α percent of all the observed haplotypes. The common haplotypes can be defined by either their frequency or the times represented in the observed haplotypes. There are three parameters in the line. The first parameter is set as 1. The second parameter, α , is a number in (0,1). The third parameter is a positive number denoted as β . The haplotypes represented more than β times are considered as common haplotypes if $\beta \geq 1$. The haplotypes with frequencies at least β are defined as common haplotypes if $0 < \beta < 1$. However, any set of SNPs can form a block when $\beta = 1$. This violates the intuition of block definition and is not allowed in our program. In other word, β can only be in either (0,1) or (1, ∞). The parameters, α and β , must be specified very carefully to ensure that each single SNP can form a block. For example, the program will be terminated if we set α as 96% and β as 5% and there is a SNP with minor allele frequency less than 5%. Since this SNP cannot be a block.

2 - LD measure D' . There are also three parameters when you define blocks based on LD measure D' . The first parameter is set as 2. The second parameter, α , and the third parameter, β , must be a number in (0,1), respectively. By this definition, the absolute D' value exceeding a threshold α is considered as strong LD and we require that the proportion of SNP pairs with strong D' must account for at least β percent of pairs of SNPs.

3 - Four-Gamete Test. When the four-gamete test is applied (Wang et al., 2002), the first parameter is set as 3 and the second parameter, β , is the frequency threshold for common haplotypes. When the PL-EM algorithm is used to infer haplotypes and their frequencies, many rare haplotypes will also appear in the analysis. It is not desirable to perform the four-gamete test for all these haplotypes. Therefore, we provide this option here to enable that the four-gamete test can be performed only on “common” haplotypes. However, the mis-specified value will result unreasonable long blocks. We suggest setting it to be very small or to be equal to 0, especially when you use haplotype data.

The definition of tag SNPs

The definitions of tag SNPs used in the programs are determined in the seventh line. Eight definitions for tag SNPs have been implemented in the current version of the program. They can be chosen by setting the first parameter in this line as one of numbers from 1 to 8. For some tag SNPs selection methods, the parameter settings in this line depend on the parameters in the sixth line.

1 - Fraction of Common Haplotypes Distinguished by tag SNPs. The first definition is based on the fraction of common haplotypes distinguished by the selected SNPs (Patil et al., 2001). In a block, the minimum set of SNPs that can uniquely distinguish a subset of common haplotypes that can account for at least α percentage of all the observed (or inferred) haplotypes are considered as a set of tag SNPs. Once the first parameter in this line is set as 1, the second parameter is the percentage of distinguished common haplotypes and the third parameter is the threshold for common haplotypes in tag SNP selection. It is worth noting that: (1) this tag selection method is only valid when the first haplotype block definition is selected in the sixth line; (2) the proportion of distinguished common haplotypes must be in (0,1) and can not be greater than the coverage of common haplotypes specified in the sixth line; (3) the frequency for common haplotypes for tag SNP selection must be in (0,1) and can not be greater than the threshold for common haplotypes defined in the sixth line.

2 - All Common Haplotypes. The second definition of tag SNP is also based on common haplotypes. Here, the minimum set of SNPs that can distinguish all common haplotypes within a block is defined as a set of tag SNPs. Except that the first parameter is set as 2; you also need to specify the second parameter, the threshold for common haplotypes.

3 - Haplotype Diversity. The third definition of tag SNP is based on haplotype diversity (Clayton 2001). In this method, the proportion of haplotype diversity explained by a subset of SNPs over all SNPs within a block is computed. The minimum set of SNPs that can account for at least α percent of overall haplotype diversity is defined as a set of tag SNPs. To use this option, you should set the first parameter as 3 and set the fraction of haplotype diversity explained by tag SNPs in the second parameter. In addition, you also need to specify the third parameter, the threshold for common haplotypes in tag SNP selection. Although this parameter is not used in searching for tag SNPs, it will be used for calculation of some statistics for a set of tag SNPs in output.

4 - Haplotype Entropy. The fourth definition of tag SNP is based on haplotype entropy (Nothnagel et al., 2003). For this method, the minimum set of SNPs that can account for at least α percent of overall haplotype entropy is defined as a set of tag SNPs. To use this option, you should set the first parameter as 4 and set the fraction of haplotype entropy explained by tag SNPs. In addition, you also need to specify the third parameter, the threshold for common haplotypes in tag SNP selection. Although this parameter is not used in searching for tag SNPs, it will be used for calculation of some statistics for a set of tag SNPs in output.

5 - Haplotype Determination Coefficient. The fifth definition of tag SNP is based on haplotype determination coefficient R_h^2 proposed by Stram et al. (2003). For this method, R_h^2 for each common haplotype is calculated based on a subset of SNPs over all SNPs within a block. Then the minimum R_h^2 among all common haplotypes is taken as the overall “haplotype prediction strength” for this subset of SNPs. Thus, the minimum set of SNPs with the overall haplotype prediction strength exceeding a pre-specified threshold is considered as a set of tag SNPs. To use this method, you must specify the first parameter as 5. The second parameter is the threshold for haplotype prediction strength and must be in (0,1). The third parameter is the threshold for common haplotypes.

6 - LD Measure r^2 . The sixth method for tag SNP selection is based on the LD measure r^2 (Carlson et al., 2004; Zhang and Jin, 2003), which is related to the statistical power in association studies. For any given subset of SNPs within a block, all pair-wise r^2 values between the SNPs in the subset and the other SNPs are calculated. For a given SNP not in the subset, we take the maximum value of r^2 as its prediction power. Then the minimum value over all SNPs not in the subset is taken as the overall prediction power of this subset of tag SNPs. Therefore, the minimum set of SNPs with prediction power exceeding a pre-specified threshold is considered as a set of tag SNPs. To use this method, you must specify the first parameter as 6. The second parameter is the threshold for prediction power and must be in (0, 1). In addition, you also need to specify the third parameter, the threshold for common haplotypes in tag SNP selection. Although this parameter is not used in searching for tag SNPs, it will be used for calculation of some statistics for a set of tag SNPs in output.

7 - All common haplotypes. This method for tag SNP selection is very similar to the second method. The minimum set of SNPs that can distinguish all common haplotypes when there are at most k ($k \geq 0$) missing SNPs for each common haplotype, is defined as a set of tag SNPs. The search procedure is accomplished by iteratively running a linear programming relaxation algorithm and a randomized rounding method (Huang et al., 2004), 2004). To use this method, you should set the first parameter as 7. The second parameter is the threshold for common haplotypes and must be in (0,1). The third parameter, k , is a positive integer.

In the eighth method, the number of tag SNPs is always set to 1, by which we can partition haplotypes into minimum number of blocks.

The Pre-Selected Tag SNPs

In tag SNP selection, there is often the need to include some specific SNPs, such as nonsynonymous SNPs in coding regions and other important functional SNPs from previous studies. This version of program provides an option in the eighth line to allow the inclusion of such specific SNPs. To include the pre-specified SNPs as tag SNPs, you need to set the first parameter as 1 and give a file containing information for pre-selected tag SNPs. Please refer to the section “Input Files” to obtain the detailed description of this file. Otherwise, you need to set the first parameter in the eighth line as 2 to disable this option.

Other Options

The program also provides three additional options to control the format of output file and to perform permutation test. The parameters in the ninth line allow you to output some intermediate results for further use. The parameters in the tenth line allow you to output haplotype patterns in defined blocks. The parameters in the eleventh line allow users to perform permutation test to assess the significance of defined blocks.

There are at most four parameters in the ninth line. Setting the first two parameters to 1 allows you to output all possible blocks and tag SNPs for further use and output all possible sets of tag SNPs. If you would like to list all possible sets of tag SNPs, you can label those tag SNPs by their names (such as rs# or other meaningful names) which are provided from a file or just by their indexes at your choice. To label tag SNPs by their names, you must set the third parameter and give the file name containing the name of SNPs in the ninth line. Otherwise the tag SNPs within each block will be labeled by their indexes.

If you want to output haplotype patterns in a block, the input data must be either haplotype data or genotype data (specified in the fifth line) and the first parameter must set as 1 in the tenth line. The second parameter in the tenth line is the name of the file for output. In addition, you can specify the first number in the eleventh line as 1 to perform permutation test if the input data is either haplotype or genotype and the dynamic program algorithm is set as 1 in the fourth line. The other two parameters, the number of permutations and the file for storing the statistics of interest are set one by one in the ninth line.

For the content and format of these output files, please refer to section “Output Files”.

Input Files

Depending on the parameters, one or more input files are needed for the program.

Map file of SNPs

If the first parameter in the fourth line is set as either 2 or 3, the program needs a map file to calculate the length of a set of consecutive SNPs. A typical map file looks like this:

```
100
1
2
3
.....
100
```

The number of total SNPs is listed in the first line and must be the same as the number of SNPs in the data file. Then the position of each SNP is given from the second line. The SNPs must be listed in an ascending order by their map positions along the chromosome.

Name of SNPs

If the third parameter in the ninth line is set as 1, the program needs a file containing names of SNPs. A typical map file looks like this:

```
100
SNP_001
SNP_002
SNP_003
.....
SNP_100
```

The number of total SNPs is listed in the first line and must be the same as the number of SNPs in the data file. Then the name of each SNP is given from the second line. The SNPs must be listed as a same order with that in the data file.

Haplotype Data

If the first number in the fifth line is set as 1, you may need a file containing haplotypes. A typical data file looks like this:

```
20    5
Hap_001    1    1    2    0    1
Hap_002    2    1    1    2    2
.....
Hap_020    1    0    1    1    0
```

In this file, the number of haplotypes and the number of SNPs in each haplotype are given in the first line of the file. They are followed by the name of the first haplotype, the allele at each SNP locus in the first haplotype. Then the name of the second haplotype is given, and so forth. Our algorithm only deals with SNP markers so far. The missing data and the two alleles at each SNP locus are coded as 0, 1 and 2 in the input file, respectively.

Genotype Data from Unrelated Individuals

If the first number in the fifth line is set as 2 or 3, you may need a file containing genotype of each individual. A typical data file looks like this:

```

20      5
Geno_001  1    0    1    1    2    0    1    0    0    2
Geno_002  2    1    1    2    2    1    0    0    2    1
.....
Geno_020  1    2    1    1    0    1    0    1    1    1

```

In this file, the number of individuals and the number of SNPs are given in the first line of the file. They are followed by the name of the first individual, two alleles at each SNP locus for the first individual. Then the name of the second individual is given, and so forth. Our algorithm can only process SNP markers so far. The missing data, and the two alleles at each SNP locus are coded as 0, 1 and 2 in this file, respectively.

Genotype Data from General Pedigrees

If the first number in the fifth line is set as 4, you may need a file containing genotype from general pedigrees. A typical data file looks like this:

```

10      3
3      1    0    0    1    1    1    1    1    2    0    1
3      2    0    0    2    0    1    2    1    2    1    2
3      3    1    2    1    1    1    1    1    2    1    1
3      4    1    2    1    1    1    1    1    2    1    1
12     11   0    0    1    1    1    1    1    2    0    1
12     22   0    0    2    0    1    2    1    2    1    2
12     23   11   22   1    1    1    1    1    2    0    1
12     44   11   22   1    1    1    1    1    2    0    1
7      5    0    0    1    0    2    2    2    1    2    2
8      3    0    0    2    0    1    1    1    2    1    1

```

The first two positive integers represent the number of individuals in the whole pedigree and the number of marker loci, respectively. Then the following records in the file give the information and the genotype of each individual in the pedigree. For each individual, the family ID, the individual ID, the father ID, the mother ID, the gender and the disease status are given first, followed by the two alleles at each locus successively. The non-negative integer is used to represent such information in all fields. In the gender field, 1 and 2 represents the male and the female, respectively. In the disease field, 1 and 0 represents affected and unaffected status, respectively. Although we do not use disease information to construct haplotypes and infer their frequencies, to be consistent with other general genotype data file used in other linkage or linkage-like software, we still keep this field in the file. At each marker locus, a positive number represents an allele and 0 represents the missing data. However, all alleles at a marker locus must be recoded as consecutive integers starting from 1. No matter how many families in pedigrees and the order of individuals in this file, our program will correctly get the proper pedigree structure and genotype data from the file. The population samples are very helpful to improve the estimates for the haplotype frequencies and can be used in this file. For this situation, each individual has a unique family ID, a non-negative individual ID and is the

only founder member in this family. In the above example, the first four individuals from family 3 (with family ID 3) and the first and the second individuals are the father and the mother of individual 3 and 4, respectively. Individuals The individual 9 and 10 (with family ID 7 and 8 and individual ID 5 and 3, respectively) are unrelated individuals. They are the only founders in family 7 and 8, respectively. This example also illustrates that the family ID and the individual ID are not necessary to be labeled by consecutive integers.

Tag SNP data

If the first parameter in the fifth line is set as 5, you may need a file containing all the possible blocks and their corresponding number of tag SNPs. A typical file looks like this:

```

100
1      1
2      0      1
.....
5      1      1      2      2      3

```

The number of total SNPs used in analysis is listed in the first line. This number is consistent with numbers in the other files. Each SNP occupies a line from the second line. In each line, the first number is a positive integer and represents the total number of blocks can be extend from this SNP to the previous SNPs. Alternatively, this number is the length of the largest block that expands from this SNP to the previous SNPs. As an example, the first number is 5 in the last line. Thus, the subset of SNPs $(i, \dots, 100)$ ($96 \leq i \leq 100$) can form a block, while $(i, \dots, 100)$ ($1 \leq i < 96$) is not a block. This representation is based on two assumptions: (1) any single SNP is a potential block; (2) If a set of consecutive SNPs does not form a block, then any subset of SNPs containing it can not be block.

The other numbers in each line characterize the number of tag SNPs in each extended block. Since the dynamic programming algorithm searches all potential blocks from a SNP backward, these numbers are listed by the search order. For instance, the number in the last line tell us that the number of tag SNPs in block $(100,100)$, $(99,100)$, $(98,100)$, $(97,100)$ and $(96,100)$ are 1, 1, 2, 2 and 3, respectively.

The File for Pre-Selected Tag SNPs

If you set the first parameter as 1 in the eighth line, you may need a file containing the pre-selected SNPs. A typical file will look like follows:

```

100
1
0
1
.....
1

```

The first number is the number of SNPs used in the analysis and it must coincide with the other input files. The other lines indicate if each of SNPs would be included in the set of tag SNPs, in which “1” represents the inclusion of that SNP. In the above example, the first SNP and the third SNPs are always included in the set of tag SNPs.

Output Files

The information of blocks

You need to provide the name of an output file to store the information of blocks. A typical output file can be divided into several parts, depending on the dynamic programming algorithm you select in the fourth line in the parameter file and the parameters in the ninth line. However, there are several common parts. The parameters used in the analysis are listed at the beginning of the file. All the possible blocks and their corresponding number of tag SNPs appear in the second part if the first parameter in the ninth line is set as 1. This part can be directly extracted and used as input for further calculation. For details of this part, please refer to “Tag SNP data” in the section of “Input Files”. The last part includes the information of blocks and looks like as following:

```

13
BlockID      NumTagSNP  StartPos  EndPos  BlockSize  NumHap  TagSNP
Block_0001      2          0         2         3         100
  0          1  1.00000  1.00000  1.00000  1.00000  1.00000  -1
  0          2  1.00000  1.00000  1.00000  1.00000  1.00000  -1
Block_0002      4          3         17        15         100
  3          5   13      16  0.83000  0.98626  0.86449  0.46245  0.04436  -1
  3          5   13      16  0.83000  0.98203  0.84685  0.63645  0.03912  -1
  4          5   13      16  0.83000  0.98494  0.85321  0.60599  0.04436  -1
.....
Block_0013      4        124       137        14         100
 124       125   130   132  0.84000  0.98472  0.85003  0.34240  0.05266  -1
 124       125   130   132  0.84000  0.98467  0.85188  0.34239  0.05266  -1
 124       125   131   132  0.84000  0.98577  0.85307  0.42918  0.05266  -1
 124       125   131   132  0.84000  0.98556  0.85447  0.42908  0.05266  -1

The total number of blocks is:      63
The total number of tag SNPs is:    69
The number of SNPs in the largest block is:    13

```

The total number of blocks is given at the beginning of this part. It is followed by the information of each block, including the block name, the number of tag SNPs, the start and end SNPs, the total number of SNPs and the total number of unambiguous haplotypes in this block. If the second parameter in the ninth line is set as 1, then all possible sets of tag SNPs with some statistics will be a part of output. Each set of tag SNPs occupies a single line and ends by a special string “-1”. A total 5 of quantities, the percentage of uniquely distinguished haplotypes, the proportion of haplotype diversity, the percentage of haplotype entropy, the minimum value of haplotype prediction strength for common haplotypes and the minimax value of pair-wise LD measure r^2 between the

tag SNPs and untagged SNPs for a set of tag SNPs are calculated and displayed, respectively. These quantities can be used for tag SNP selection when multiple sets of tag SNPs exist in a single block.

From the above example, we can see that the second block contains 15 SNPs starting from SNP 4 and ending with SNP 17. There are four tag SNPs in this block and there are three possible sets of tag SNPs, {3, 5, 13, 16}, {3, 10, 13, 16} and (4, 5, 13, 16). The tag SNPs are labeled by their indexes. You may already notice that the index of SNPs always starts from 0.

For your convenience, the total number of blocks, the total number of tag SNPs and the number of SNPs in the largest block are also given at the end of file, respectively. These statistics, along with other statistics of interest, can be easily retrieved.

In the procedure of tag SNP selection, an approximate algorithm is employed for some methods. This algorithm can find all the possible sets of tag SNPs with the absence of missing data. However, our algorithm cannot guarantee to find all sets of tag SNPs when the missing data present. Occasionally, it may fail to find a set of tag SNPs even the number of tag SNPs in a block is larger than 0.

If the parametric dynamic programming algorithm is selected in the fourth line, an additional part appears in this file. It looks like as following:

Results Part -- All deletion parameters and its corresponding block partition:

```

12
deletionPara numBlock numTagSNP coverNumSNP coverRatio numDelete
retainGenome delGenome
0.1500000 1 3 20 0.08547 2 20.000 214.000
0.1666667 4 13 80 0.34188 5 80.000 154.000
0.1739130 5 17 103 0.44017 6 103.000 131.000
.....
.....
0.3750000 13 45 221 0.94444 5 221.000 13.000
0.4285714 14 48 228 0.97436 2 228.000 6.000
0.6000000 15 51 233 0.99573 1 233.000 1.000
1.0000000 15 52 234 1.00000 0 234.000 0.000

```

In summary, the data has 8 columns, representing the deletion parameter at an intersection point, the total number of blocks, the total number of tag SNPs, the total number SNPs contained in blocks, the fraction of genome covered by blocks, the total number of deleted intervals, the total number of SNPs contained in deleted intervals, the total length of genome covered by blocks and the total length of genome covered by deleted intervals. For instance, there are a total of 12 blocks containing 45 tag SNPs when the deletion parameter is set as a value between [0.3750000, 0.4285714). The portion of genome covered by blocks is about 94.4%. Thus, if you want to cover at least 94.4% of genome, you need at least 45 tag SNPs.

If the two-dimension dynamic programming algorithm is selected in the fourth line, an additional part appears in this file. It looks like as following:

```

Results Part -- For discrete dynamic programming algorithm:
53
numBlock  numTagSNP  coverNumSNP  coverRatio  numDelete  retainGenome
delGenome
0      0      0      0.00000      1      0.000      234.00
1      1      5      0.02137      2      5.000      229.00
1      2      11     0.04701      2      11.000     223.00
1      3      20     0.08547      2      20.000     214.00
2      4      25     0.10684      3      25.000     209.00
2      5      31     0.13248      3      31.000     203.00
.....
15     51     233     0.99573      1     233.000     1.00
15     52     234     1.00000      0     234.000     0.00

```

These results are very similar with the results using parametric dynamic programming algorithm. In summary, the data has 7 columns, representing the total number of blocks, the total number of tag SNPs, the total number SNPs contained in blocks, the fraction of genome covered by blocks, the total number of deleted intervals, the total number of SNPs contained in deleted intervals, the total length of genome covered by blocks and the total length of genome covered by deleted intervals. For instance, there are a total of 2 blocks covering about 13.25% of genome when the number of tag SNPs is 5. Thus, only about 13.25% of genome can be covered using 5 tag SNPs.

To fully understand this part of data, please refer to Zhang et al. (2003) to see the details of the algorithms.

The haplotype pattern

If the input data is either haplotype or genotype, you can output haplotypes in each block to a file. The format of this file is as follows when the input is haplotype data:

```

63
Block_0001: StartPos = 0, EndPos = 0, haplotype = 2 74
.....
Block_0007: StartPos = 11, EndPos = 20, haplotype = 12 57

Haplotype Pattern 0: 2111212211
num = 14
Hap_002 : 2111212211
.....
Hap_072 : 2101210211

Haplotype Pattern 1: 1222122112
num = 9
Hap_014 : 1222022012
.....
Hap_070 : 1222122012
.....

```

```

Haplotype Pattern 11: 2122122112
num = 1
Hap_068 : 2122122112

Ambiguous Haplotypes
num = 23
Hap_003 : 2111220111
.....
Hap_076 : 1202120112
.....

Haplotypes and Their Frequencies (from data)
2111212211  0.245614
1222122112  0.157895
.....
2122122112  0.017543

Block_0063: StartPos = 231, EndPos = 233, haplotype = 3 61
.....

```

The total number of blocks is given in the first line. This number should be consistent with the number in other files. For each block, the name of the block, the index of starting and ending SNPs, the total number of haplotype patterns and the total number of unambiguous haplotypes in each block are given one by one. For instance, Block_0007 starts from SNP 11 and expand to SNP 20. There are 12 different kinds of haplotypes and a total of 57 unambiguous haplotypes. The different haplotypes are listed in ascending order by the number of haplotypes that compatible with them. In the above example, the first haplotype pattern is “**2111212211**” and a total of 14 unambiguous haplotypes are compatible with it. All ambiguous haplotypes are provided subsequently after the unambiguous haplotypes. At last, the haplotype and their frequencies estimated from data (when input is haplotype) or by the PL-EM (when input is haplotype) are listed.

There are only some differences between the output file based on the haplotype data and that based on the genotype data. When the genotype data from general pedigree are available, only haplotypes and estimated frequencies are listed. In addition, some individuals may not be used in haplotype inference due to high fraction of missing data and will be listed after the unambiguous haplotypes. The following example clearly demonstrates that only one individual has not been used in the haplotype inference.

```

Unused individuals in Haplotype Inference:
Num = 1
PEDX4551 : 00000000
PEDX4551 : 00000000

```

The results of permutation

If you want to minimize the total number of tag SNPs using either haplotype data or genotype data, you can run permutation test to assess the significant of blocks. This program will give the following statistics for each permutation and store them in a given file: (1) the total number of blocks; (2) the total number of tag SNPs; (3) the number of SNPs in the largest block; (4) the number of blocks that contain more than 10 SNPs; (5)

the number of blocks that contain from 3 to 10 SNPs; (6) the number of block that contains less than 3 SNPs. However, you can easily modify the routine provided in the program to output other statistics of interest. A typical output file is as follows:

Permutation_Id	NumBlock	NumTagSNP	MaxBlock	num01	num02	num03
Permutation_00001	90	115	6	0	47	43
Permutation_00002	91	112	5	0	42	49
.....						
Permutation_00099	89	111	5	0	47	42
Permutation_00100	91	110	6	0	47	44

Limitations and Update

In our implementation, even a single SNP is still considered as a block and the tag SNPs can be itself or none depending on the definition of tag SNPs. Therefore, we assume that each single SNP locus should be a potential block and the dynamic programming algorithm thus could go backwardly at least one step. The validation of this assumption certainly depends on the data itself and the definition of haplotype blocks. Once one SNP locus is found not to form a block, the program will report an error and be terminated. We suggest you to delete the data at this locus and run program again when this happens.

If the input is genotype data, we will infer haplotypes and their frequencies in each block again after the block boundaries have been determined by the dynamic programming algorithm. In general, these haplotypes and their frequencies will be slightly different from the estimation in the dynamic programming algorithm. In some situations, these differences can result a subset of consecutively SNPs fails to form a block. The program will be terminated due to this inconsistency. We recommend you to run program again to get the final results.

Our algorithms are quite flexible that can incorporate different definitions of haplotype blocks and tag SNPs. And we are planning to implement more methods in our program. You are welcome to provide new methods that you want us to implement in this program. We greatly appreciate if you could point out any bugs when you use our program. Please check our web site periodically to get the newest update of this program.

Contact Information

If you have any problem, please contact:

Fengzhu Sun, PhD
Departments of Biology
University of Southern California
1042 W 36th Place, DRB142
Los Angeles, CA90089-1113
(213) 740-2413 (phone)
(213) 740-2424 (fax)

Email: fsun@hto.usc.edu
Webpage: <http://www-hto.usc.edu/~fsun>

Or

Kui Zhang, Ph.D.
Section on Statistical Genetics
Department of Biostatistics
University of Alabama at Birmingham
Birmingham, AL, 35294
(205) 996-4094 (phone)
(205) 975-2540 (fax)
Email: kzhang@ms.soph.uab.edu
Webpage: <http://www.soph.uab.edu/ssg/default.aspx?id=108>

References:

Carlson CS, Eberle MA, Rieder MJ, Yi Q, Kruglyak L, Nickerson DA (2004) Selecting a maximally informative set of single-nucleotide polymorphisms for association analysis using linkage disequilibrium. *Am J Hum Genet* 74: 106-120.

Clayton D (2001) <http://www.nature.com/ng/journal/v29/n2/extref/ng1001-233-S10.pdf>.

Gabriel SB, Schaffner SF, Nguyen H, Moore JM, Roy J, Blumenstiel B, Higgins J, DeFelice M, Lochner A, Faggart M, Liu-Cordero SN, Rotimi C, Adeyemo A, Cooper R, Ward R, Lander ES, Daly MJ, Altshuler D (2002) The structure of haplotype blocks in the human genome. *Science* 296: 2225-2229.

Huang YT, Zhang K, Chen T, Chao KM (2004) Approximation Algorithms for the Selection of Robust Tag SNPs. *Lecture Notes in Computer Science/Lecture Notes in Bioinformatics* (For Proceedings of WABI 2004 (4th Workshop on Algorithms in Bioinformatics, Bergen, Norway, September 14 - 17, 2004)). In press.

Nothnagel M, Furst R, Rohde K (2003) Entropy as a measure for linkage disequilibrium over multilocus haplotype blocks. *Hum Hered* 54: 186-198.

Patil N, Berno A J, Hinds D A, Barrett W A, Doshi J M, Hacker CR, Kautzer CR, Lee D H, Marjoribanks C, McDonough DP, et al. (2001) Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science* 294, 1719-1723.

Qin Z, Niu T, Liu J. 2002. Partitioning-Ligation-Expectation-Maximization Algorithm for haplotype inference with single-nucleotide Polymorphisms. *Am J Hum Genet* 71: 1242-1247.

Stram DO, Haiman CA, Hirschhorn JN, Altshuler D, Kolonel LN, Henderson BE, Pike MC (2003) Choosing haplotype-tagging SNPs based on unphased genotype data using

preliminary sample of unrelated subjects with an example from the multiethnic cohort study. *Hum Hered* 55: 27-36.

Wang N, Akey JM, Zhang K, Chakraborty K and Jin L (2002) Distribution of recombination crossovers and the origin of haplotype blocks: The interplay of population history, recombination, and mutation. *Am J Hum Genet* 71: 1227–1234.

Zhang K, Deng M, Chen T, Waterman MS and Sun F (2002) A dynamic programming algorithm for haplotype partitioning. *Proc Natl Acad Sci USA* 99: 7335-7339.

Zhang K, Jin L (2003) HaploBlockFinder: haplotype block analysis. *Bioinformatics*, 19: 1300-1301.

Zhang K, Qin Z, Chen T, Waterman MS, Liu JS, Sun F (2004a) Haplotype Block Partitioning and Tag SNP selection using genotype data and their applications to association studies. *Genome Res*, in press.

Zhang K, Sun F, Waterman MS, Chen T (2003) Dynamic programming algorithms for haplotype block partitioning: applications to human chromosome 21 haplotype data. *Am J Hum Genet* 73: 63-73.

Zhang K, Sun F, Zhao H (2004b) HAPLORE: A program for haplotype reconstruction in general pedigrees without recombination. *Bioinformatics*, in press.