

HAPBLOCK – A dynamic programming algorithm to define haplotype blocks

Introduction

This program, HAPBLOCK, was developed for haplotype partitioning under the joint guidance of Ting Chen, Fengzhu Sun, and Michael Waterman within the Center for Computational and Experimental Genomics of the University of Southern California. It was written by Dr. Kui Zhang, a postdoctoral fellow in the center. The primary objective is to minimize the number of representative SNPs required to account for most of the haplotype information in each block. The haplotype block partitions with the minimum number of representative SNPs may not be unique. The secondary objective is to minimize the number of blocks. Both objectives are achieved through a dynamic programming algorithm. Any measure of haplotype information can be used in the algorithm and of course the measure should depend on the specific application. For the detailed algorithm and its applications, refer to Zhang et al. (2002).

The main Menu

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 run it under Windows 95, 98 or NT operating systems. You can also compile it and execute it under UNIX systems. On running the program, an interactive menu will appear on the screen and guide you to set the parameters that are needed to define the haplotype blocks.

Global Parameters

At the beginning of program, you will be asked to give the maximum number of marker loci, the maximum number of haplotypes and the maximum number of blocks that a block can extend. These numbers should be positive. The maximum number of marker loci and the maximum number of haplotypes should be at least equal to the numbers in data file (in the next subsection). The maximum number of blocks that a block can extend should be large enough for long blocks. If the program reports an error like this: “the length of sub-haplotype exceeds the maximum number in dpBlock, please change the maximum number for sub-haplotypes in main”. Then you need to increase this parameter.

Input File

First, you will be prompted to provide an input file. The input file contains the haplotype data you want to analyze. A typical file has the following format:

```
20      5
hap_01      1      3      4      0      2
```

hap_02 4 3 1 1 3

The number of haplotypes and the number of marker loci 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 marker 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 bases 'A', 'C', 'G', 'T' are coded as 0, 1, 2, 3 and 4 in the input file, respectively.

Setting Parameters

You will be asked to give a threshold [a number in (0,1)] to define the block. We follow the block definition of Patil et al. (2001). A subset of consecutive SNPs forms a block if the percentage of repeated unambiguous haplotypes over all the unambiguous haplotypes exceeds this threshold.

Then, the measure for haplotype block information needs to be specified. Two measures are provided. The first measure is the fraction of unambiguous haplotypes uniquely identified by the representative SNPs (Patil et al. 2001). The second measure is the haplotype diversity (Clayton 2001).

Finally, the threshold for defining the representative SNPs in each block. The haplotype information measure defined by the representative SNPs over the haplotype information defined by all the SNPs must be no smaller than this threshold.

Output File

You should provide the name of the output file to store the information of blocks. If the file already exists, it will be replaced by this file. A typical output file is as follows:

```
2579
Block_id      SNPs Needed  StartPos     EndPos       BlockSize
Block_0001    2            0            14           15
Block_0002    0            15           15           1
.....
The total number of SNPs needed: 2556
The total number of blocks: 2579
The number of SNPs in the largest block: 174
```

The total number of blocks is given at the beginning of the file. It is followed by the information of each block, including the block name, the minimum number of representative SNPs, the start and end SNPs and the total number of SNPs in this block. For convenience, the total number of representative SNPs, the total number of blocks and the number of SNPs in the largest block are also given in the last lines of the file.

Other Options

The program also provides two additional options. The first option allows you to output haplotype patterns in defined blocks. The second uses permutation test to assess the significance of the defined blocks. However, you can ignore them to save time.

If you want to output the haplotype patterns to a file, you must first provide a file to store the patterns. A format of this file is as follows:

Block Information: startPos = 0, endPos = 14

Pattern 1: aaccaacaaccaac
CPD0002C28 naccaacaaccnnn
CPD0005C53 aacnaacaaccnnn

Pattern 2: caaaccaacaannn

ambiguous haplotype
CPD0001C23 nnnnnnnnnnnnnnnn
CPD0003C04 naaccaacaannn

The block information (the start and end SNPs) is given first and followed by the patterns of haplotype and its corresponding samples. The ambiguous haplotypes are provided at the end of this block.

If you want to perform permutation test, the program will ask you to provide the number of permutations and a file to store the statistics of interest. This program then gives the following statistics for each permutation: (1) the total number of blocks; (2) the total number of representative 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	blockNum	totalNumSNP	maxBlock	Num01	Num02	Num03
Permut_00001	43	51	5	0	16	27
Permut_00002	45	53	6	0	17	28
Permut_00003	47	55	5	0	16	31
Permut_00004	44	53	5	0	18	26
Permut_00005	41	51	5	0	18	23
Permut_00006	44	50	5	0	18	26
.....						

Bugs

Currently, we assume that each single locus should be a potential block and the dynamic programming algorithm could go forward at least one step. This is true for SNPs data with at least 5 samples using the current block definition. If one locus has too many missing data and is not a potential block, the program will report an error and suggest that to delete the data at this locus and run program again.

Contact Information

If you have any problems, 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)
fsun@hto.usc.edu
<http://www-hto.usc.edu/~fsun>

References:

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

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) Science 294, 1719-1723.

Zhang K, Deng M, Chen T, Waterman MS and Sun F. 2002 A dynamic programming algorithm for haplotype partitioning. Submitted.