HapMiner Version 1.1 User Manual

Jing Li
Department of Electrical Engineering and Computer Science
Case Western Reserve University
jingli@eecs.cwru.edu.

January 22, 2006

References:

(i) Li, J. and Jiang, T. 2005. Haplotype-based linkage disequilibrium mapping via direct data mining. *Bioinformatics* **21**:4384-4393.

(ii) Li, J. et al. Haplotype-based Quantitative Trait Mapping Using a Clustering Algorithm. Submitted.

Contents

1 Introduction **5**

1.1 What's new 5

1.2 Overview 5

1.3 Algorithms 5

1.4 Platforms 9

2 Running the software **11**

2.1 Obtaining and installing the software 11

2.2 File list 11

2.3 Running the software 11

2.4 Input files and data preparation 12

2.5 Output files 14

2.6 Conventions and restrictions 15

Chapter 1

Introduction

1.1 What's new

We have implemented the QTL mapping method [2] in version 1.1 of HapMiner. In addition, we have extended the types of weight functions. The two weight functions now can be assigned to different functions. Pairwise linkage disequilibrium can also be used as the weight function. There are also some minor changes. The permutation test will be automatically performed unless the number of permutation is 0. There is no need to provide the number of SNPs and the number of individuals anymore. We also correct one bug in the previous version regarding the statistics of the last locus.

1.2 Overview

HapMiner is a computer program for association mapping based on directly mining of the haplotypes via a density-based clustering algorithm. HapMiner is well suited for gene fine mapping and efficient for whole-genome screens. This document provides the following information about the software: brief description about the methods, how to obtain, install, and run the software, how to set parameters and how to read the results files. Details about the methods can be referred to [1, 2].

1.3 Algorithms

We briefly describe the idea of the algorithms in HapMiner for mapping qualitative trait. The framework is the same for quantitative trait. HapMiner works as follows. A whole-genome screen for haplotype association is performed by sliding a window with certain length. Within each window, clusters are identified based on some similarity measure via a density-based clustering algorithm. The Pearson χ^2 statistic or Z-score based on a contingency table derived from the numbers of case haplotypes and control haplotypes in a cluster

can be used as an indicator of the degree of association between the cluster and disease. Both measures can also be used as association/independence test statistics, properly adjusted (*e.g.*, using Bonferroni correction) for multiple tests. A statistical significance threshold can be chosen independent of the sample size and all findings that exceed the threshold will be reported. In the current algorithm, we only use the Z-score as an indicator of the degree of association. The effectiveness of the method depends on the similarity measure of haplotype fragments used in the clustering algorithm. We will first describe the new haplotype similarity measure.

A general haplotype (dis)similarity measure. The similarity of two haplotype segments is defined with respect to a particular marker locus. Suppose that we focus on a marker at locus 0, with loci $1, 2, \ldots, r$ on one side and $-1, -2, \ldots, -l$ on the other side. Assume that the genetic/physical distance from any locus to locus 0 is known and denoted as x_k, where $-l \leq k \leq r$. A haplotype h spanning this region is just an $(l + 1 + r)$-dimensional vector and the k^{th} dimension of h, denoted as $h(k)$, is the allele at locus k. For a pair of haplotypes h_i, h_j , we define the similarity score of h_i, h_j with respect to locus 0 as:

$$s_{i,j} = \sum_{k=-l}^{r} w_1(x_k) I(h_i(k), h_j(k)) + \sum_{k=1}^{r'} w_2(x_k) + \sum_{k=-1}^{-l'} w_2(x_k), \tag{1.3.1}$$

where I is the identity function, $-l'$ and r' are two boundary loci such that the two haplotypes h_i, h_j are identical between these two loci and different at both locus $-l' - 1$ and locus $r' + 1$. The weights w_1 and w_2 are two decreasing functions so that the measure on each locus is weighted according to the distance from locus 0. The choices of the weights w_1 and w_2 will be discussed shortly.

The first summation in Equation 1.3.1 is a weighted measure of the number of alleles in common between haplotypes h_i and h_j in the region, which can be thought of as Hamming similarity. The remaining summations form a weighted measure of the longest continuous interval of matching alleles around locus 0, which has some resemblance to the notion of a longest common substring (*p.p.* 125 in [3]). This definition is quite flexible and generalizes several similarity measures used in the literature [4]. For instance, by setting $w_1 = 1$ and $w_2 = 0$, the measure becomes the counting measure described in [4]. The length measure in the same article can be achieved by setting $w_1 = 0$ and $w_2 = 1$. This definition of haplotype similarity is more powerful than the above two specialized measures and can be used for different types of markers by choosing appropriate weighting functions. It has the strengths of both specialized measures. That is, it is robust against recent marker mutations and genotyping/haplotyping errors, and it also apprehends partial sharing from a common ancestral haplotype due to historical recombination events. Notice that $s_{i,i} = s_{j,j}$, a distance metric between haplotypes h_i and h_j at marker locus 0 can be defined as:

$$d_{i,j} = \frac{s_{i,i} - s_{i,j}}{s_{i,i}} = \frac{s_{j,j} - s_{i,j}}{s_{j,j}}. \tag{1.3.2}$$

The distance is normalized to the interval $[0, 1]$ so it will not increase with the length of haplotypes.

The requirement for both weighting functions w_1 and w_2 is that they must be decreasing functions. It can be exponentially, quadratically, or linearly decreasing. It can also be a discrete function with its values

defined only at marker positions. The user has the freedom of choosing the weighting function depending on the marker density of the input data. The selection of w_1 and w_2 in our simulation is basically a linear functions since we are using dense markers. Missing alleles can be handled directly in the calculation of similarity measure by either taking all the missing alleles as a new distinct allele, or imputing them first according to allele frequencies in the sample. As an extension to the definition, when considering multiple DS loci simultaneously, the overall distance is defined as an average of pairwise distances at each locus.

A density-based clustering algorithm. Clustering is a powerful tool for mining massive data. In the haplotype association mapping setup, we are interested in identifying haplotype clusters that are strongly associated with the disease under study. The goal is not to partition all the haplotypes into certain clusters. Neither do we try to build a cladogram because of the difficulty of reconstructing the evolutionary relationship for all the haplotypes. Instead, we believe that haplotypes from affected individuals are expected to be more similar at the disease gene location than those from controls which are assumed to be random samples. We do not expect control haplotypes to form any clusters except by chance. A difficulty lies in the fact that, due to the existence of allele heterogeneity and phenocopies, some haplotypes from affected individuals do not necessarily form a cluster. This is also a main reason why a gene mapping method using case-control data would likely fail in reality if it assumes, explicitly or implicitly, that all or at least most affected individuals do have the same disease mutations. We take the problem of finding strongly disease associated haplotype clusters as the problem of finding clusters from data with noise background. We use the concept of "density-based clusters" and adopt an algorithm called DBSCAN [5] with minor modifications. In order to keep the paper self-contained, we briefly introduce the DBSCAN algorithm in the context of haplotype mapping.

There are two input parameters for DBSCAN. One is the radius of the interested neighborhood ϵ and the other is a density threshold *MinPts*. A haplotype is called a *core* haplotype if there are more than *MinPts* haplotypes in its ϵ neighborhood. The haplotypes in the ϵ neighborhood are *directly reachable* from the core haplotype and a haplotype is *reachable* from a core haplotype if there is a chain of core haplotypes between these two haplotypes where each is directly reachable from the preceding one. Two haplotypes are *density-connected* if there is a core haplotype such that both haplotypes are reachable from it. A *density-based cluster* of haplotypes is a set of density-connected haplotypes with maximal density-reachability. All the above definitions are with respect to the two parameters ϵ and *MinPts*. DBSCAN examines every haplotype and starts to construct a cluster once a core haplotype is found. It then iteratively collects directly reachable haplotypes from a core haplotype, merging clusters when necessary. The process terminates when all haplotypes have been examined. The clusters are then output and the haplotypes that do not belong to any cluster are regarded as noise. More details about the algorithm can be found in [5].

Score of the degree of association. We measure the degree of association between a haplotype cluster and the disease of interest using Z-scores. Suppose that we are given m case haplotypes and n control haplotypes. Let m' and n' denote the number of case and control haplotypes in a cluster, respectively. A

2×2 contingency table can be constructed and the Z-score of the cluster is defined as:

$$Z = \frac{m'/m - n'/n}{\sqrt{\frac{m'+n'}{m+n}(1 - \frac{m'+n'}{m+n})(1/m + 1/n)}}. \tag{1.3.3}$$

It is the weighted difference of relative frequencies of the case and control haplotypes in a cluster and follows approximately a normal distribution if we assume haplotypes randomly occur in the cluster. A large Z-score means strong association between the cluster (actually, the haplotypes within the cluster) and the disease. The cluster with the highest score is taken as the prediction for each marker. The score is regarded as the point estimation of each marker locus and a consensus haplotype pattern or a haplotype profile based on the cluster can be used as diseased associated pattern centered at the locus. The allele heterogeneity can be naturally modelled by taking multiple clusters at each position if their scores are significant. To assess the significance of the predicted disease clusters, a permutation test can be easily performed by shuffling the disease labels.

The degree of association with quantitative traits. Analogous to the Z-score used in disease gene mapping [1], we measure the degree of association with the trait under study using a Q-score, which is actually a t-statistic when we assume that the haplotypes in the cluster and the remaining haplotypes are sampled from two different populations. A large Q-score means strong association between the cluster (actually, the haplotypes within the cluster) and the trait. More specifically, let m denote the number of haplotypes in the cluster and let n denote the number of remaining haplotypes. Let $\hat{\mu_c}$ and $\hat{\sigma}_c^2$ denote the sample mean and variance of the m haplotypes within the cluster and let $\hat{\mu_r}$ and $\hat{\sigma}_r^2$ denote the sample mean and variance of the n remaining haplotypes, respectively. The Q-score of the cluster is defined as:

$$Q = \frac{(\hat{\mu_c} - \hat{\mu_r})\sqrt{m+n-2}}{\sqrt{(\hat{\sigma}_c^2 + \hat{\sigma}_r^2)(1/m + 1/n)}}. \tag{1.3.4}$$

It is the scaled difference of population means from two samples, evaluated using sample means and variances, and follows approximately a t-distribution if we assume the trait values within the cluster and outside the cluster are independent, and both are normally distributed within groups with identical variances. This homoscedasticity assumption might not hold because similar haplotypes are more closely related, and thus their trait values should be more similar to each other, than haplotypes not forming any clusters. Therefore, it is expected that the variance of the trait values inside a cluster will be smaller. In this case, Welch's T test for two independent samples with different variances can be adopted. Notice that the proposed algorithm is non-parametric in nature and its significate level is obtained via a permutation test. The t distribution and the assumption of equal variances are not used directly in the algorithm.

The overall time complexity of our algorithm is $O(MN^2)$, where M is the total number of marker loci (or the number of all possible pairs of loci when studying a 2-gene disease) and N is the sample size which is around hundreds in most real datasets. So, the algorithm is efficient for whole-genome screens.

1.4 Platforms

HapMiner V1.1 has executable code for Windows and Linux, and other platforms may be available in the future.

Chapter 2

Running the software

2.1 Obtaining and installing the software

The HapMiner software and this document can be obtained at http://www.eecs.case.edu/~jxl175/HapMiner.html. Once downloading the software, the user could create a folder and uncompress the files under the newly created folder. The user could use WinZip to unzip the file on Windows and use the command "`tar -xzvf HapMiner.tar.z`" to unzip the file on Linux.

2.2 File list

In addition to the executable file HapMiner.exe (HapMiner on Linux) and the user's guide HapMinerReadMeV1.1.pdf, there are three more example files: "dgdata", "qtldata", and "parameter.txt". The files "dgdata" and "qtldata" contain simulated data of a disease gene and a quantitative trait, respectively. The file "parameter.txt" contains the parameters necessary to run HapMiner. The format of the files will be discussed in details in Section 2.4.

2.3 Running the software

One can launch HapMiner on a Command Prompt (DOS) window on Windows by typing the following command:

`HapMiner.exe -option datafile parameterfile [distfile]`.

Similarly, one can launch HapMiner by typing the following command on Linux terminal:

`HapMiner -option datafile parameterfile [distfile]`.

The above assumes that HapMiner is in the current directory. Otherwise, we need include the correct path information in front of the executable file.

There are 2 options in the current version, one for qualitative trait and one for quantitative trait. Different options will invoke different algorithms as follows.

```
-s:  qualitative trait (disease gene) mapping
-q:  quantitative trait (QTL) mapping
```

2.4 Input files and data preparation

The first row of the input data file is the title row. Starting from the second row, each row represents an individual. The title row specifies the meaning of each column. The first column is the individual ID, followed by the affected status, 'a' indicates affected (case) haplotypes and 'c' indicates normal (control) haplotypes, or the value of the quantitative trait. The remaining columns are marker alleles, represented by non-negative integers, one allele each column. Zero stands for missing value.

The file "parameter.txt" contains important parameters used in HapMiner and the meanings of the parameters will be explained in details shortly. The file is organized in the following way. Each parameter is specified by two consecutive rows. The first row is a tag and has the format "Pi", where i is a number indicating which parameter. The second row specify the value of each parameter. Currently, the file contains 6 parameters and an example file (parameter.txt) is provided.

```
P1
1 0.2
P2
0.25
P3
7
P4
1 -10 1
P5
1000
P6
1.0
```

The first parameter specifies the parameter ϵ used in the clustering algorithm. The first row in the two-row set is the tag "P1". There are tow ways to set the value of ϵ. The first integer (either 1 or 2) in the second row tells HapMiner which way to set ϵ and the second parameter is the value. The first way to set ϵ is to specify the value of ϵ directly. Since the pairwise distances are within the range [0,1], one can specify the neighborhood radius ϵ to be 0.2 for example. The other way to set ϵ is to choose a percentile according to the

distribution of all pairwise distances. For example the second row can be "2 0.01", which tells HapMiner to choose the (lower) 1 percentile from the distribution of all pairwise distances as the value of ϵ.

The second has a tag row "P2" (and each parameter has a such tag row of "Pi" and we will not mention it any more for the remainings). It specifies how the parameter *MinPts* used in the clustering algorithm will be chosen. Again, here we use the percentile concept to select the value of *MinPts* instead of directly assigning its value. We first calculate the number of neighbors (respect to ϵ, which is chosen first) for every haplotype and based on the distribution of the number of neighbors, we choose the (top) 25 percentile as the value of *MinPts*.

The way to choose the values of these two parameters reflects some prior knowledge about the data, for example, about the rate of phenocopy, allele heterogeneity, *etc.* The users may use the default values if lack of such information and may try different values to see the changes of the results.

The third parameter sets the length of haplotype segment (the width of the sliding window). This parameter can be chosen based on the marker interval distances. The default value is set to be 7 markers. When choosing the segment length, it is useful to think about the weight functions simultaneously. The fourth parameter specifies the way to choose the two weight functions used in formulae 1.3.1. In HapMiner V1.1, there are six ways to set the weight functions, indicated by the value of the first integer (denoted as t for type). The total number of remaining parameters in this line depends on the value of the first integer. In the first case ($t = 1$), both weight functions take the same exponential function of the form be^{ax} and in the second case ($t = 2$), both weight functions take the same linear function of the form $ax + b$, where the x is the genetic distance in Morgan of a marker to the central marker. In both cases, the second and the third numbers are the value of a and b, respectively. The value a must be a negative number since both weight functions must be decreasing functions. If the return value of the linear function is smaller than 0, the value of the weight function is set to be 0. The marker interval distances can be read through the marker interval distance file. By default, HapMiner assumes that the marker interval distance is 1 cM if no distance file is supplied. HapMiner can also use pairwise LD as the weight functions, which can be specified by setting $t = 3$. In this case, the second parameter in this line must be specified as 1, 2, or 3, representing three cases: 1) $w1 = LD, w2 = 0$; 2) $w1 = 0, w2 = LD$; 3) $w1 = w2 = LD$. Users may also specify different functions for $w1$ and $w2$ by using $t = 4$ (exponential) or $t = 5$ (linear). In this two cases, there are four additional values in this line, representing $a1, b1$ and $a2, b2$ for $w1$ and $w2$. In other words, type 4 ($t = 4$) will be the same as type 1 ($t = 1$) if $a1 = a2$ and $b1 = b2$. The last type ($t = 6$) can be used to select constant $w1$ and $w2$. There are two additional binary integers in this line, corresponding to the values of $w1$ and $w2$ respectively. For example, given an input like "6 1 0", HapMiner will use $w1 = 1$ and $w0 = 0$ (*i.e.*, Hamming distance, or counting measure [4]). While the case "6 0 1" represents the length of longest continuous interval of matching alleles around a locus (similar as the length measure [4], but in HapMiner the measure is with respect to a specific locus).

The fifth parameter is to control the times for a permutation test. In version 1.1, users do not need to invoke the program using different options for permutation test. They only need to change the value of

permutation times. It is set to be 1000 by default. If users do no need to perform a permutation test, its value should be 0.

The sixth parameter is to control the output file. It is the threshold of the Z-score (or Q-score for QTL) and only clusters with Z-score larger than the threshold will be output in the "output" file described in section 2.5.

The file about the marker interval distance is very simple and only contains one line. That is, the marker interval distances in the same order of the marker alleles are separated by a space or a tab.

The input file can be prepared by using any file editors such as Notepad on Windows or Emacs or vi on Linux. The data files can also be exported from a database system. There can be sets of data in the input data file (for simulations), but they will use the same set of parameters. At the end of the input file, there is an empty line. The format of the all the input file should be strictly followed.

2.5 Output files

HapMiner has two output files. If the input data file has the name "datafile", one output file will be named as "datafile-results" and the other file will be named as "output". The "datafile-results" contains the score information for each marker position. There is one line for each marker and each line has six values for a qualitative trait, representing the genetic distance from the first position, marker index and Z-score, $-\log(p)$, Z-score based on single marker and its $-\log(p)$ value, at the current position. The last line of the file contains the positions with the largest Z-score (*i.e.* the prediction) by HapMiner and by the single SNP association (χ^2), together with their scores and p-values. The score profile figures in [1] are drawn based on the information in this file. The file "output" contains all the clusters for every marker position such that their Z-scores are larger than a predefined threshold (the parameter 6 in the parameter file). More specifically, for each marker position, all the clusters with Z-score large than the threshold will be output one by one. For each cluster, HapMiner outputs the ID of the cluster, followed by the haplotypes within the cluster. For each haplotype, it outputs the individual ID, the affected status, whether it is a core haplotype and the haplotype pattern centered at the current marker position where the segment length is as predefined. The last line of a cluster reports the ChiSquare and the Z-score based on the number of case and control haplotypes in the cluster, followed by the total number of haplotypes, total number of core haplotypes in the cluster, and the significate level ($-\log(p)$).

For QTL mapping, the structure of the two files ("datafile-results" and "output") are the same as that for disease gene mapping. But the items in each line for QTL mapping are different from that for disease mapping. Each line in the "datafile-results" contains the score information with nine values for each marker position. These values are the genetic distance from the first position, marker index and Q-score, $-\log(p)$, T statistic based on single marker, Welch-T statistic and its degree of freedom, the Mann-Whitney U (MWU) statistic and the Z-score associated with MWU [2]. The last line of the file contains the prediction results

of three methods, *i.e.*, HapMiner, T-test based on single SNPs, and MWU. For each method, we output the predicted position, the statistic, its $-\log(p)$-value and its permutation p-value.

2.6 Conventions and restrictions

There are some conventions and restrictions in the current version of HapMiner:

- The current version can only deal with SNP data. We will test the algorithms on multi-allelic data and implement it in the next release.

Bibliography

[1] Li, J. and Jiang, T. Haplotype-based linkage disequilibrium mapping via direct data mining. *Bioinformatics* **21**:4384-4393, 2005.

[2] Li, J. et al. Haplotype-based Quantitative Trait Mapping Using a Clustering Algorithm. *Submitted*.

[3] Gusfield, D. Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, Cambridge, UK, 1997.

[4] Tzeng, J.Y., Devlin, B., Wasserman, L. & Roeder, K. On the identification of disease mutations by the analysis of haplotype similarity and goodness of fit. *Am. J. Hum. Genet.*, **72**:891-902, 2003.

[5] Easter, M., Kriegel, H.P., Sander, J. & Xu, X. A density-based algorithm for discoving clusters in large spatial databases with noise. In *Proc KDD'96*, 226-231, 1996.