Haplo Stats:
Statistical Methods for Haplotypes When Linkage Phase is Ambiguous
version 1.1.1

Jason P. Sinnwell and Daniel J. Schaid
Mayo Clinic, Rochester MN

April 23, 2004

Contents

1 Brief Description **4**

2 Operating System and Installation **4**

3 New Features **5**

4 Getting Started **5**
- 4.1 Example Data . . . 5
- 4.2 Creating a Genotype Matrix . . . 6
- 4.3 Random Numbers and Setting Seed . . . 7

5 Preview Missing Data: summaryGeno **7**

6 Haplotype Frequency Estimation: haplo.em **9**
- 6.1 Algorithm . . . 9
- 6.2 Example Usage . . . 10
- 6.3 Control Parameters for haplo.em . . . 13
- 6.4 Haplotype Frequencies by Group Subsets . . . 14

7 Haplotype Score Tests: haplo.score **16**
- 7.1 Quantitative Trait Analysis . . . 16
- 7.2 Ordinal Trait Analysis . . . 18
- 7.3 Binary Trait Analysis . . . 19
- 7.4 Plots and Haplotype Labels . . . 20
- 7.5 Skipping Rare Haplotypes . . . 20
- 7.6 Haplotype Scores, Adjusted for Covariates . . . 22
- 7.7 Permutation p-values . . . 23
- 7.8 Combine Score and Group Results: haplo.score.merge . . . 25
- 7.9 Score Tests on Sub-Haplotypes: haplo.score.slide . . . 26
- 7.10 Plot Results from haplo.score.slide . . . 27

8 Regression Models: haplo.glm **27**
- 8.1 Setting Up the data.frame . . . 29
- 8.2 Regression for a Quantitative Trait . . . 29
- 8.3 Fitting Haplotype x Covariate Interactions . . . 31
- 8.4 Regression for a Binomial Trait . . . 32
- 8.5 Control Parameters and Genetic Models . . . 34

9 License and Warranty **36**

10 Acknowledgements **37**

A Counting Haplotype Pairs When Marker Phenotypes Have Missing Alleles **38**

1 Brief Description

Haplo Stats is a suite of S-PLUS/R routines for the analysis of indirectly measured haplotypes. The statistical methods assume that all subjects are unrelated and that haplotypes are ambiguous (due to unknown linkage phase of the genetic markers). The genetic markers are assumed to be codominant (i.e., one-to-one correspondence between their genotypes and their phenotypes), and so we refer to the measurements of genetic markers as genotypes. The primary functions in Haplo Stats are:

- haplo.em: for the estimation of haplotype frequencies, and posterior probabilities of haplotype pairs for a subject, conditional on the observed marker data
- haplo.glm: glm regression models for the regression of a trait on haplotypes, possibly including covariates and interactions
- haplo.score: score statistics to test associations between haplotypes and a variety of traits, including binary, ordinal, quantitative, and Poisson.

For those users who have used the previously distributed haplo.score package, it is important to note that the haplo.score function has changed dramatically from the previous distribution, including the parameters passed to this function. Please follow the examples provided in this document to see how to use this function.

2 Operating System and Installation

Haplo Stats version 1.1.1 library is written for both S-PLUS (version 6.0) and R for Unix. It has been placed on the Comprehensive R Archive Network (CRAN) for R version 1.9.0. Like other packages available on CRAN, Haplo Stats is made available for Unix and Windows systems, with additional portability to other systems as well. Installation procedures for S-PLUS and R systems will vary; the Unix installations are explained in the README.haplo.stats text file, located at the top level of the haplo.stats directory. The procedures for running analyses are the same for S-PLUS and R, following instructions in this document.

3 New Features

1. Accounting for missing genotypes: The original release of the haplo.score package removed subjects who were missing any marker genotypes. The current Haplo Stats functions allow for missing marker genotypes.

2. Improved EM algorithm for estimating haplotype frequencies: See section 6.

3. Haplotype frequencies by subsets: Another new feature provides estimated haplotype frequencies for subsets defined by levels of a qualitative "group" variable (see the new function haplo.group). This information can be combined with output from haplo.score by the new function haplo.score.merge. These new functions are useful for case-control studies in order to align estimates of haplotype frequencies for cases and controls with the corresponding score statistics.

4. Regression models: The function haplo.glm is a major new addition, which provides a way to regress a trait on haplotypes, covariates, and possibly their interactions.

4 Getting Started

After installing the Haplo Stats package, the routines and an example data set are available by starting an S-PLUS or R session and attaching the appropriate directory. The easiest way to get started is by following an example. An experienced user may want to skip the example and simply view the details in the help files. As illustrated in the following example session, a user enters the indented text following the prompt ">", and the output results follow.

4.1 Example Data

First load the Haplo Stats library and the example data set (hla.demo). If the Haplo Stats library is installed for global use, load the library as done below. If installed as a local library, specify its location in lib.loc parameter as shown in comments(##).

```
##if local library, use:
        ##library(haplo.stats, lib.loc='/install/path/'

> library(haplo.stats)
> setupData(hla.demo)

[1] "hla.demo"

> attach(hla.demo)
> names(hla.demo)

 [1] "resp"     "resp.cat" "male"     "age"      "DPB.a1"   "DPB.a2"
 [7] "DPA.a1"   "DPA.a2"   "DMA.a1"   "DMA.a2"   "DMB.a1"   "DMB.a2"
[13] "TAP1.a1"  "TAP1.a2"  "TAP2.a1"  "TAP2.a2"  "DQB.a1"   "DQB.a2"
[19] "DQA.a1"   "DQA.a2"   "DRB.a1"   "DRB.a2"   "B.a1"     "B.a2"
[25] "A.a1"     "A.a2"
```

The column names of hla.demo are shown above. They are defined as follows:

- **resp:** quantitative antibody response to measles vaccination
- **resp.cat:** a factor with levels "low", "normal", "high", for categorical antibody response
- **male:** gender code with 1="male" , 0="female"
- **age:** age (in months) at immunization

The remaining columns are genotypes for 11 HLA loci, with a prefix name (e.g., "DQB") and a suffix for each of two alleles (".a1" and ".a2"). The variables in hla.demo can be accessed by typing hla.demo$ before their names, such as hla.demo$resp. Alternatively, it is easier for these examples to attach hla.demo, (as shown above with attach()) so the variables can be accessed by simply typing their names.

4.2 Creating a Genotype Matrix

Many of the functions require a matrix of genotypes, denoted here as geno. This matrix is arranged such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a

chromosome. If there are K loci, then the number of columns of geno is $2K$. Rows represent the alleles for each subject. For example, if there are three loci, in the order A-B-C, then the 6 columns of geno would be arranged as A.a1, A.a2, B.a1, B.a2, C.a1, C.a2. For illustration, three of the loci in hla.demo will be used to demonstrate some of the functions. Create a separate data frame for 3 of the loci, and call this geno. Then create a vector of labels for the loci.

```
> geno <- hla.demo[, c(17, 18, 21:24)]
> label <- c("DQB", "DRB", "B")
```

4.3 Random Numbers and Setting Seed

Simulations are used in several of the functions (e.g., to determine random starting values for haplo.em, and to compute permutation p-values in haplo.score). In order to reproduce results in this user guide, you must set the .Random.seed before any function which uses random numbers. We illustrate this below, and we invisibly reset the seed in making this document. In practice, however, the user would not ordinarily reset the seed.

```
> seed <- c(17, 53, 1, 40, 37, 0, 62, 56, 5, 52, 12, 1)
> set.seed(seed)
```

The above mechanism for controlling .Random.seed makes results reproducible in the respective S-PLUS and R platforms. However, the random number generators for S-PLUS and R use the seeds differently, so results will not completely agree across platforms. Because the results in this document were generated by R on a Unix platform, results from S-PLUS that depend on random numbers will not exactly match the results in this document. Nonetheless, results can be forced to agree across platforms by omitting the randomness within haplo.em (and its results used in haplo.score and haplo.glm) by setting the control parameter n.try=1 within haplo.em.control (see section 6.3).

5 Preview Missing Data: summaryGeno

Before computing haplotype statistics, the user may want to look for missing genotype data to determine the completeness of the data. If many genotypes

are missing, the functions may take a long time to compute results, and the user may want to remove some of the subjects with a lot of missing data. This can be accomplished with the summaryGeno function, which checks for missing allele information and counts the number of potential haplotype pairs that are consistent with the observed data (see the Appendix for a description of this counting algorithm).

The codes for missing values of alleles are defined by the parameter miss.val, which may be a vector to define multiple missing value codes. Because it has been common practice to use a zero to code for missing alleles, the default values for miss.val are 0 and NA. Below, name the result from summaryGeno as geno.desc. Since it is a data frame, individual rows may be printed. Here we show the results for subjects 1-10, 80-85, and 135-140.

```
> geno.desc <- summaryGeno(geno, miss.val = c(0, NA))
> print(geno.desc[c(1:10, 80:85, 135:140), ])
```

	loc miss-0	loc miss-1	loc miss-2	num_enum_rows
1	3	0	0	4
2	3	0	0	4
3	3	0	0	4
4	3	0	0	2
5	3	0	0	4
6	3	0	0	2
7	3	0	0	4
8	3	0	0	2
9	3	0	0	2
10	3	0	0	1
80	3	0	0	4
81	2	0	1	1800
82	3	0	0	2
83	3	0	0	1
84	3	0	0	2
85	3	0	0	4
135	3	0	0	4
136	3	0	0	2
137	1	0	2	129600
138	3	0	0	4
139	3	0	0	4
140	3	0	0	4

The columns with 'loc miss-' illustrate the number of loci missing either 0, 1, or 2 alleles, and the last column, num_enum_rows, illustrates the number of haplotype pairs that are consistent with the observed data. In the example above, subjects indexed by rows 81 and 137 have missing alleles. Subject #81 has one locus missing two alleles, while subject #137 has two loci missing two alleles. As indicated by num_enum_rows, subject #81 has 1,800 potential haplotype pairs, while subject #137 has nearly 130,000.

Because of the missing data, the number of possible haplotype pairs is quite large, which increases computation time of haplo.em in section 6.2. With geno rows #81 and #137 included, haplo.em requires about 300 seconds of CPU time, while without those two rows it takes just over 1 second. It is a good idea to preview the data for missing values using the summaryGeno function. If there are just a few subjects with missing alleles, it may be worthwhile to exclude them.

6 Haplotype Frequency Estimation: haplo.em

6.1 Algorithm

For genetic markers measured on unrelated subjects, with linkage phase unknown, haplo.em computes maximum likelihood estimates of haplotype probabilities. Because there may be more than one pair of haplotypes that are consistent with the observed marker phenotypes, posterior probabilities of haplotype pairs for each subject are also computed. Unlike the usual EM which attempts to enumerate all possible haplotype pairs before iterating over the EM steps, our *progressive insertion* algorithm progressively inserts batches of loci into haplotypes of growing lengths, runs the EM steps, trims off pairs of haplotypes per subject when the posterior probability of the pair is below a specified threshold, and then continues these insertion, EM, and trimming steps until all loci are inserted into the haplotype. The user can choose the batch size. If the batch size is chosen to be all loci, and the threshold for trimming is set to 0, then this reduces to the usual EM algorithm. The basis of this progressive insertion algorithm is from the "snphap" software by David Clayton[2]. Although some of the features and control parameters of haplo.em are modeled after snphap, there are substantial differences, such as extension to allow for more than two alleles per locus, and some other nuances on how the algorithm is implemented.

6.2 Example Usage

Use haplo.em on geno for the 3 loci defined above, then view the results stored in save.em. In this example we show just a quick glance of the output by using the option nlines=10, which prints only the first 10 haplotypes of the full results. (The nlines parameter has been employed in some of the print methods in the Haplo Stats package to shorten the lengthy results for this user guide. In practice, it is best to exclude this parameter so that the default will print all results.)

```
> save.em <- haplo.em(geno = geno, locus.label = label, miss.val = c(0,
+     NA))
> print(save.em, nlines = 10)

================================================================================

                                    Haplotypes

================================================================================

   DQB DRB  B hap.freq
1   21   1  8  0.00232
2   21   2  7  0.00227
3   21   2 18  0.00227
4   21   3  8  0.10408
5   21   3 18  0.00229
6   21   3 35  0.00570
7   21   3 44  0.00378
8   21   3 45  0.00227
9   21   3 49  0.00227
10  21   3 57  0.00227
================================================================================

                                     Details

================================================================================

lnlike =  -1847.675
lr stat for no LD =  632.8897 , df =  125 , p-val =  0
```

Explanation of Results

The haplotypes and their estimated frequencies are listed, as well as a few details. The lr stat for no LD is the likelihood ratio statistic contrasting the lnlike for the estimated haplotype frequencies versus the lnlike assuming that alleles from all loci are in linkage equilibrium. Trimming by the progressive insertion algorithm can invalidate the lr stat and the degrees of freedom (df) — see the help file for haplo.em for more details.

Summary Method

The summary on save.em shows the list of haplotypes per subject, and their posterior probabilities:

```
> summary(save.em, nlines = 7)

================================================================================

            Subjects: Haplotype Codes and Posterior Probabilities

================================================================================

  subj.id hap1code hap2code posterior
1       1       78       58   1.00000
2       2       13      143   0.12532
3       2      138       17   0.87468
4       3      168       25   1.00000
5       4       13       39   0.28621
6       4       17       38   0.71379
7       5       55       94   1.00000
================================================================================

                    Number of haplotype pairs: max vs used

================================================================================


x        1   2   3   72  135
```

```
1        18   0   0   0   0
2        50   4   0   0   0
4       116  29   1   0   0
1800      0   0   0   1   0
129600    0   0   0   0   1
```

Explanation of Results

The first part of summary lists the subject id (row number of input geno matrix), the codes for the haplotypes of each pair, and the posterior probabilities of the haplotype pairs. The second part gives a table of the maximum number of pairs of haplotypes per subject, versus the number of pairs used in the final posterior probabilities. The haplotype codes remove the clutter of illustrating all the alleles of the haplotypes, but may not be as informative as the actual haplotypes themselves. To see the actual haplotypes, use the show.haplo=TRUE option:

```
> summary(save.em, show.haplo = TRUE, nlines = 7)

================================================================================

            Subjects: Haplotype Codes and Posterior Probabilities

================================================================================

     subj.id hap1.DQB hap1.DRB hap1.B hap2.DQB hap2.DRB hap2.B posterior
78         1       32        4     62       31       11     61   1.00000
13         2       21        7      7       62        2     44   0.12532
138        2       62        2      7       21        7     44   0.87468
168        3       63       13     62       31        1     27   1.00000
13.1       4       21        7      7       31        7     44   0.28621
17         4       21        7     44       31        7      7   0.71379
55         5       31       11     51       42        8     55   1.00000
================================================================================

                    Number of haplotype pairs: max vs used

================================================================================
```

```
x          1   2   3   72  135
  1       18   0   0   0   0
  2       50   4   0   0   0
  4      116  29   1   0   0
  1800     0   0   0   1   0
  129600   0   0   0   0   1
```

6.3 Control Parameters for haplo.em

An additional argument can be passed to haplo.em, called "control". This is a list of parameters that control the EM algorithm based on progressive insertion of loci. The default values are set up by a function called haplo.em.control (see the help(haplo.em.control) for a complete description). Although the user can accept the default values, there are times when they may need to be adjusted. For example, for small sample sizes and many possible haplotypes, finding the global maximum of the log-likelihood can be difficult. The algorithm uses multiple attempts to maximize the log-likelihood, starting each attempt with random starting values. If the results from haplo.em, haplo.score, or haplo.glm change when rerunning the analyses, this may be due to different maximizations of the log-likelihood. To avoid this, the user can increase the number of attempts (n.try) to maximize the log-likelihood, increase the batch size (insert.batch.size), or decrease the trimming threshold for posterior probabilities (min.posterior). If the EM algorithm fails to converge, try increasing the maximum number of iterations (max.iter). These parameters are defined below:

- **insert.batch.size:** Number of loci to be inserted in a single batch.
- **min.posterior:** Minimum posterior probability of haplotype pair, conditional on observed marker genotypes. Posteriors below this minimum value will have their pair of haplotypes "trimmed" off the list of possible pairs.
- **max.iter:** Maximum number of iterations allowed for the EM algorithm before it stops and prints an error.

- **n.try:** Number of times to try to maximize the lnlike by the EM algorithm. The first try will use, as initial starting values for the posteriors, either equal values or uniform random variables, as determined by random.start. All subsequent tries will use uniform random values as initial starting values for the posterior probabilities.

The example below illustrates how to set the number of tries to 20, and maximum number of iterations to 1,000. The function would take about twice as long to finish, so we comment the code here just to demonstrate the syntax.

```
##save.em <- haplo.em(geno=geno, locus.label=label, miss.val=c(0, NA),
 ##control = haplo.em.control(n.try = 20, max.iter = 1000) )
```

6.4 Haplotype Frequencies by Group Subsets

To compute the haplotype frequencies for each level of a grouping variable, use the function haplo.group. The following example illustrates the use of a binomial response based on resp.cat, y.bin, that splits the subjects into two groups.

```
> y.ord <- as.numeric(resp.cat)
> y.bin <- ifelse(y.ord == 1, 1, 0)
> group.bin <- haplo.group(y.bin, geno, locus.label = label, miss.val = 0)
> print(group.bin, nlines = 15)

-------------------------------------------------------------------------------

                      Counts per Grouping Variable Value

-------------------------------------------------------------------------------

group
  0   1
157  63


-------------------------------------------------------------------------------
```

Haplotype Frequencies By Group

	DQB	DRB	B	Total	y.bin.0	y.bin.1
1	21	1	8	0.00232	0.00335	NA
2	21	10	8	0.00181	0.00318	NA
3	21	13	8	0.00274	NA	NA
4	21	2	18	0.00227	0.00318	NA
5	21	2	7	0.00227	0.00318	NA
6	21	3	18	0.00229	0.00637	NA
7	21	3	35	0.00570	0.00639	NA
8	21	3	44	0.00378	0.00333	0.01587
9	21	3	45	0.00227	NA	NA
10	21	3	49	0.00227	NA	NA
11	21	3	57	0.00227	NA	NA
12	21	3	70	0.00227	NA	NA
13	21	3	8	0.10408	0.06974	0.19048
14	21	4	62	0.00455	0.00637	NA
15	21	7	13	0.01072	NA	0.02381

Explanation of Results

The group.bin object can be very large, depending on the number of possible haplotypes, so only a portion of the output is illustrated above. The first section gives a short summary of how many subjects appear in each of the groups. The second section is a table with the following columns:

- The first column gives row numbers.
- The next columns (3 in this example) illustrate the alleles of the haplotypes.
- Total are the estimated haplotype frequencies for the entire data set.
- The last columns are the estimated haplotype frequencies for the subjects in the levels of the group variable (y.bin=0 and y.bin=1 in this example). Note that some haplotype frequencies have an "NA", which occurs when the haplotypes do not occur in the subgroups.

7 Haplotype Score Tests: haplo.score

The function haplo.score is used to compute score statistics to test associations between haplotypes and a wide variety of traits, including binary, ordinal, quantitative, and Poisson. This function provides several different global and haplotype-specific tests for association, allows for adjustment for non-genetic covariates, and optionally allows computation of permutation p-values (which may be needed for sparse data). Details on the background and theory of the score statistics can be found in Schaid et al.[6].

7.1 Quantitative Trait Analysis

First, analyze the quantitative trait called resp. A quantitative trait is identified in haplo.score by the parameter trait.type="gaussian" (a reminder that a gaussian distribution is assumed for the distribution of the error terms). The other arguments, all set to default values, are defined in the help file, viewed by typing help(haplo.score). Note that rare haplotypes can result in unstable variance estimates, and hence unreliable test statistics for the rare haplotypes. For hints on handling rare haplotypes, see section 7.5. Execute the function then view the results using the print method (again, output shortened by nlines).

```
> score.gaus <- haplo.score(resp, geno, trait.type = "gaussian",
+     skip.haplo = 0.005, locus.label = label, simulate = FALSE)
> print(score.gaus, nlines = 10)

------------------------------------------------------------------------------

                          Global Score Statistics

------------------------------------------------------------------------------

global-stat = 46.49606, df = 38, p-val = 0.16222



------------------------------------------------------------------------------
```

```
                    Haplotype-specific Scores
-------------------------------------------------------------------------
      DQB DRB B  Hap-Freq Hap-Score p-val
 [1,] 21  3   8  0.10408  -2.39631  0.01656
 [2,] 21  7   13 0.01072  -2.29908  0.0215
 [3,] 31  4   44 0.02849  -2.24273  0.02491
 [4,] 63  13  60 0.00575  -1.75669  0.07897
 [5,] 62  2   35 0.00751  -1.2084   0.22689
 [6,] 51  1   44 0.01731  -0.99357  0.32043
 [7,] 63  13  44 0.01606  -0.84453  0.39837
 [8,] 33  7   57 0.00682  -0.58522  0.5584
 [9,] 31  11  44 0.01059  -0.54951  0.58266
[10,] 63  2   7  0.01333  -0.50736  0.6119
```

Explanation of Results

The section Global Score Statistics prints results for testing an overall association between haplotypes and the response. The global-stat has an asymptotic χ^2 distribution, with degrees of freedom (df) and p-value as indicated. Haplotype-specific scores are given in a table format. The column descriptions are as follows:

- The first column gives row numbers.
- The next columns (3 in this example) illustrate the alleles of the haplotypes.
- Hap-Freq is the estimated frequency of the haplotype in the pool of all subjects.
- Hap-Score is the score for the haplotype, the results are sorted by this value.
- p-val is the asymptotic chi-square (1 df) p-value.

7.2 Ordinal Trait Analysis

To create an ordinal trait, convert resp.cat (a factor with levels "low", "normal", "high") to numeric values, y.ord (with levels 1, 2, 3). For haplo.score, use y.ord as the response variable, and set the parameter trait.type = "ordinal".

```
> y.ord <- as.numeric(resp.cat)
> score.ord <- haplo.score(y.ord, geno, trait.type = "ordinal",
+     offset = NA, x.adj = NA, skip.haplo = 0.005, locus.label = label,
+     miss.val = 0, simulate = FALSE)
> print(score.ord, nlines = 10)
```

```
-------------------------------------------------------------------------------

                          Global Score Statistics

-------------------------------------------------------------------------------

global-stat = 62.89774, df = 38, p-val = 0.00674



-------------------------------------------------------------------------------

                          Haplotype-specific Scores

-------------------------------------------------------------------------------

      DQB DRB B  Hap-Freq Hap-Score p-val
 [1,] 21  7   13 0.01072  -3.67306  0.00024
 [2,] 21  3   8  0.10408  -2.79247  0.00523
 [3,] 31  4   44 0.02849  -2.61319  0.00897
 [4,] 63  13  60 0.00575  -2.35846  0.01835
 [5,] 33  7   57 0.00682  -0.93375  0.35043
 [6,] 33  9   60 0.00682  -0.93375  0.35043
 [7,] 31  11  44 0.01059  -0.75601  0.44964
 [8,] 62  2   35 0.00751  -0.7069   0.47963
 [9,] 63  13  44 0.01606  -0.69172  0.48911
[10,] 51  1   44 0.01731  -0.62185  0.53404
```

Warning for Ordinal Traits

When analyzing an ordinal trait with adjustment for covariates (using the x.adj option), the software requires the libraries Design and Hmisc, distributed by Frank Harrell, Ph.D.[4]. If the user does not have these libraries installed, then it will not be possible to use the x.adj option. However, the unadjusted scores for an ordinal trait (using the default option x.adj=NA) do not require these libraries. Check the list of your local libraries in the list shown from entering library() in your prompt.

7.3 Binary Trait Analysis

Because "low" responders are of primary interest, create a binary trait that has values of 1 when response is "low", and 0 otherwise. Then in haplo.score specify the parameter trait.type="binomial".

```
> y.bin <- ifelse(y.ord == 1, 1, 0)
> score.bin <- haplo.score(y.bin, geno, trait.type = "binomial",
+     offset = NA, x.adj = NA, skip.haplo = 0.005, locus.label = label,
+     miss.val = 0, simulate = FALSE)
> print(score.bin, nlines = 10)

-------------------------------------------------------------------------------

                            Global Score Statistics

-------------------------------------------------------------------------------

global-stat = 61.82492, df = 38, p-val = 0.00861



-------------------------------------------------------------------------------

                            Haplotype-specific Scores

-------------------------------------------------------------------------------
```

```
      DQB DRB B  Hap-Freq Hap-Score p-val
 [1,] 62  2   7  0.05098  -2.19387  0.02824
 [2,] 51  1   35 0.03018  -1.58421  0.11315
 [3,] 63  13  7  0.01655  -1.56008  0.11874
 [4,] 21  7   7  0.01246  -1.47495  0.14023
 [5,] 64  13  35 0.00897  -1.27347  0.20285
 [6,] 63  13  62 0.00866  -1.14173  0.25356
 [7,] 32  8   7  0.00682  -1.10475  0.26927
 [8,] 64  13  63 0.00682  -1.10475  0.26927
 [9,] 64  13  60 0.00654  -1.09005  0.27569
[10,] 51  1   51 0.00727  -1.08131  0.27956
```

7.4 Plots and Haplotype Labels

A convenient way to view results from haplo.score is a plot of the haplotype frequencies (Hap-Freq) versus the haplotype score statistics (Hap-Score), as shown in Figure 1.

Some points on the plot may be of interest, perhaps due to their score statistic, or their haplotype frequency. To identify individual points in the plot, use locator.haplo(score.gaus) for this example. Then with the left mouse button, click on all the points of interest. After points are chosen, click on the middle mouse button, and the points are labeled with their haplotype labels.

7.5 Skipping Rare Haplotypes

For the quantitative trait analyses, the option skip.haplo=.005 was used to pool all haplotypes with frequencies < 0.005 into a common group. As a guideline, you may wish to set skip.haplo to get expected haplotype counts greater than 5. If N is the number of subjects and f the haplotype frequency, then the expected haplotype count is $E = 2 \times N \times f$. So you can choose skip.haplo $= \frac{E}{2 \times N}$ if $E = 5$ is chosen. Here we try a different cut-off than before, skip.haplo=.01, which corresponds to expected haplotype counts of $2 \times 220 \times .01 = 4.4$. In the output, notice the global statistic and its p-value change (due to decreased df), but the haplotype-specific scores do not change.

```
> plot(score.gaus)
```

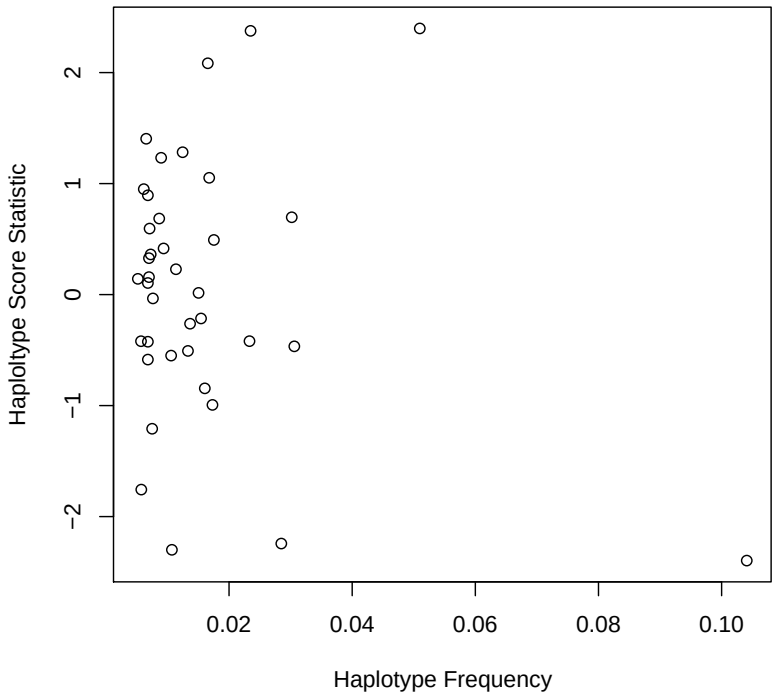

Figure 1: Haplotype Statistics: Score vs. Frequency; Quantitative Response

```
> score.gaus.01 <- haplo.score(resp, geno, trait.type = "gaussian",
+     offset = NA, x.adj = NA, skip.haplo = 0.01, locus.label = label,
+     miss.val = 0, simulate = FALSE)
> print(score.gaus.01, nlines = 10)
--------------------------------------------------------------------------------

                            Global Score Statistics

--------------------------------------------------------------------------------
```

```
global-stat = 33.37945, df = 20, p-val = 0.03064
```

Haplotype-specific Scores

```
      DQB DRB B  Hap-Freq Hap-Score p-val
 [1,] 21  3   8  0.10408  -2.39631  0.01656
 [2,] 21  7   13 0.01072  -2.29908  0.0215
 [3,] 31  4   44 0.02849  -2.24273  0.02491
 [4,] 51  1   44 0.01731  -0.99357  0.32043
 [5,] 63  13  44 0.01606  -0.84453  0.39837
 [6,] 31  11  44 0.01059  -0.54951  0.58266
 [7,] 63  2   7  0.01333  -0.50736  0.6119
 [8,] 32  4   60 0.0306   -0.46606  0.64118
 [9,] 21  7   44 0.02332  -0.41942  0.67491
[10,] 62  2   44 0.01367  -0.26221  0.79316
```

7.6 Haplotype Scores, Adjusted for Covariates

First set up a covariate matrix, with the first column for male (1 if male; 0 if female), and the second column for age (in months). Then use the matrix as an argument to haplo.score. When adjusting for covariates, all score statistics can change, though not by much in this example.

```
> x.ma <- cbind(male, age)
> score.gaus.adj <- haplo.score(resp, geno, trait.type = "gaussian",
+     offset = NA, x.adj = x.ma, skip.haplo = 0.005, locus.label = label,
+     miss.val = 0, simulate = FALSE)
> print(score.gaus.adj, nlines = 10)
```

Global Score Statistics

```
--------------------------------------------------------------------------------

global-stat = 46.68398, df = 38, p-val = 0.15764


--------------------------------------------------------------------------------

                          Haplotype-specific Scores

--------------------------------------------------------------------------------

      DQB DRB B  Hap-Freq Hap-Score p-val
 [1,] 21  3   8  0.10408  -2.4097   0.01597
 [2,] 21  7   13 0.01072  -2.28963  0.02204
 [3,] 31  4   44 0.02849  -2.25293  0.02426
 [4,] 63  13  60 0.00575  -1.77427  0.07602
 [5,] 62  2   35 0.00751  -1.21288  0.22517
 [6,] 51  1   44 0.01731  -0.98763  0.32333
 [7,] 63  13  44 0.01606  -0.83952  0.40118
 [8,] 33  7   57 0.00682  -0.60404  0.54582
 [9,] 31  11  44 0.01059  -0.547    0.58438
[10,] 63  2   7  0.01333  -0.48483  0.6278
```

7.7 Permutation p-values

Permutation p-values are computed when simulate=TRUE. In addition to the global statistic and haplotype-specific statistics, the function computes a "max-stat" statistic and corresponding permutation p-value. The max-stat is the maximum among all haplotype-specific score statistics. Because the distribution of this statistic is unknown, the p-value for max-stat is given only when permutations are requested. If only a few haplotypes are associated with the trait, the max-stat should have greater power than the global statistic.

The score.sim.control function manages simulation control parameters. Simulated statistics are based on randomly permuting the trait and covariates (same order for both), but not the geno matrix, and then computing the

haplotype score statistics, adjusted for covariates. haplo.score employs the simulation p-value precision criteria of Besag and Clifford[1]. These criteria ensure that the permutation p-values for both the global and the maximum score statistics are precise for small p-values. The algorithm performs a user-defined minimum number of permutations (min.sim) to guarantee sufficient precision for the simulated p-values for score statistics of individual haplotypes. Permutations beyond this minimum are then conducted until the sample standard errors for simulated p-values for both the global and max score statistics are less than a threshold (p.threshold * p-value). The default value for p.threshold= $\frac{1}{4}$ provides a two-sided 95% confidence interval for the p-value with a width that is approximately as wide as the p-value itself. Effectively, simulations are more precise for smaller p-values. The following example illustrates computation of permutation p-values with min.sim=1000.

```
> score.bin.sim <- haplo.score(y.bin, geno, trait.type = "binomial",
+     offset = NA, x.adj = NA, skip.haplo = 0.005, locus.label = label,
+     miss.val = 0, simulate = TRUE, sim.control = score.sim.control())
> print(score.bin.sim, nlines = 10)

-------------------------------------------------------------------------------

                            Global Score Statistics

-------------------------------------------------------------------------------

global-stat = 61.82492, df = 38, p-val = 0.00861

-------------------------------------------------------------------------------

                        Global Simulation p-value Results

-------------------------------------------------------------------------------

Global sim. p-val =  0.00598
Max-Stat sim. p-val =  0.00563
Number of Simulations, Global:  2842 , Max-Stat: 2842

-------------------------------------------------------------------------------
```

```
                        Haplotype-specific Scores

-------------------------------------------------------------------------------

      DQB DRB B  Hap-Freq Hap-Score p-val   sim p-val
 [1,] 62  2   7  0.05098  -2.19387  0.02824 0.03272
 [2,] 51  1   35 0.03018  -1.58421  0.11315 0.13476
 [3,] 63  13  7  0.01655  -1.56008  0.11874 0.19177
 [4,] 21  7   7  0.01246  -1.47495  0.14023 0.15588
 [5,] 64  13  35 0.00897  -1.27347  0.20285 0.32231
 [6,] 63  13  62 0.00866  -1.14173  0.25356 0.28325
 [7,] 32  8   7  0.00682  -1.10475  0.26927 0.54293
 [8,] 64  13  63 0.00682  -1.10475  0.26927 0.57037
 [9,] 64  13  60 0.00654  -1.09005  0.27569 0.18825
[10,] 51  1   51 0.00727  -1.08131  0.27956 0.3677
```

7.8 Combine Score and Group Results: haplo.score.merge

When analyzing a qualitative trait, such as binary, it can be helpful to align the results from haplo.score with haplo.group. To do so, use the function haplo.score.merge, as illustrated in the following example:

```
> merge.bin <- haplo.score.merge(score.bin, group.bin)
> print(merge.bin, nlines = 10)

-------------------------------------------------------------------------------

              Haplotype Scores, p-values, and Frequencies By Group

-------------------------------------------------------------------------------

   DQB DRB  B Hap.Score   p.val Hap.Freq y.bin.0 y.bin.1
1   62   2  7   -2.1939 0.02824  0.05098 0.06789 0.01587
2   51   1 35   -1.5842 0.11315  0.03018 0.03754 0.00907
3   63  13  7   -1.5601 0.11874  0.01655 0.02176      NA
4   21   7  7   -1.4749 0.14023  0.01246 0.01969      NA
5   64  13 35   -1.2735 0.20285  0.00897 0.01318      NA
```

```
6   63  13 62   -1.1417 0.25356  0.00866 0.01274      NA
7   32   8  7   -1.1047 0.26927  0.00682 0.00955      NA
8   64  13 63   -1.1047 0.26927  0.00682 0.00955      NA
9   64  13 60   -1.0900 0.27569  0.00654 0.00637      NA
10  51   1 51   -1.0813 0.27956  0.00727 0.00955      NA
```

Explanation of Results

The first column is a row index, the next columns (3 in this example) illustrate the haplotype, the Hap.Score column is the score statistic and p.val the corresponding χ^2 p-value. Hap.prob is the haplotype frequency for the total sample, and the remaining two columns are the estimated haplotype frequencies for each of the group levels (y.bin in this example). The default print method only prints results for haplotypes appearing in the haplo.score output. To view all haplotypes, use the print option all.haps=TRUE, which prints all haplotypes from the haplo.group output. The output is ordered by the score statistic, but the order.by parameter can specify ordering by haplotypes or by haplotype frequency. See the help file for print.haplo.score.merge for details on printing options.

7.9 Score Tests on Sub-Haplotypes: haplo.score.slide

To evaluate the association of sub-haplotypes (subsets of alleles from the full haplotype) with a trait, the user can evaluate a "window" of alleles by haplo.score, and slide this window across the entire haplotype. This procedure is implemented by the function haplo.score.slide. To illustrate this method, we use all 11 loci in the demo data, hla.demo.

First, make the geno matrix and the locus labels for the 11 loci. Then use haplo.score.slide for a window of 3 loci (n.slide=3), which will slide along the haplotype for all 9 contiguous subsets of size 3, using the same gaussian trait as above.

```
> geno.11 <- hla.demo[, -c(1:4)]
> label.11 <- c("DPB", "DPA", "DMA", "DMB", "TAP1", "TAP2", "DQB",
+     "DQA", "DRB", "B", "A")
> score.slide.gaus <- haplo.score.slide(resp, geno.11, trait.type = "gaussian",
+     n.slide = 3, skip.haplo = 0.005, locus.label = label.11)
> print(score.slide.gaus)
```

```
  start.locus score.global.p score.global.p.sim score.max.p.sim
1           1      0.2963703                 NA              NA
2           2      0.0078762                 NA              NA
3           3      0.2272005                 NA              NA
4           4      0.7663358                 NA              NA
5           5      0.2172384                 NA              NA
6           6      0.2111682                 NA              NA
7           7      0.2178191                 NA              NA
8           8      0.1052554                 NA              NA
9           9      0.0404776                 NA              NA
```

Explanation of Results

The first column is the row index of the nine calls to haplo.score, the second column is the number of the starting locus of the sub-haplotype, the third column is the global score statistic p-value. The last two columns are the simulated p-values for the global and maximum score statistics, respectively. If you specify simulate=TRUE in the function call, the simulated p-values would be present.

7.10 Plot Results from haplo.score.slide

The results from haplo.score.slide can be easily viewed in a plot shown in Figure 2.

The x-axis has tick marks for each locus, and the y-axis is the $-log_{10}(pval)$. To select which p-value to plot, use the parameter pval, with choices "global", "global.sim", and "max.sim" corresponding to p-values described above. If the simulated p-values were not computed, the default is to plot the global p-values. For each p-value, a horizontal line is drawn at the height of $-log_{10}(pval)$ across the loci over which it was calculated. For example, the p-value score.global.p = 0.0078762 for loci 2-4 will plot as a horizontal line plotted at $y = 2.1$ covering the 2^{nd}, 3^{rd}, and 4^{th} x-axis tick marks.

8 Regression Models: haplo.glm

The function haplo.glm computes the regression of a trait on haplotypes, and possibly other covariates and their interactions with haplotypes. Although

```
> plot(score.slide.gaus)
```

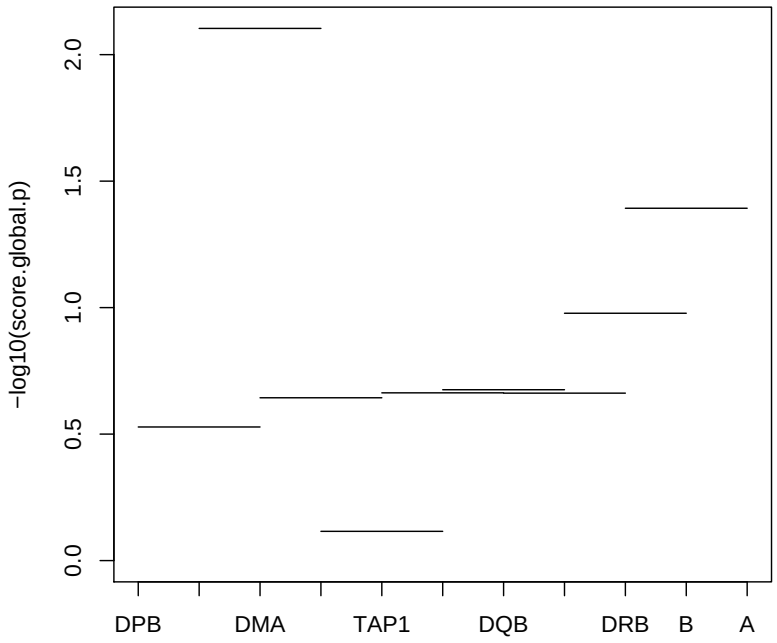

Figure 2: Global p-values for sub-haplotypes; Gaussian Response

this function is based on a generalized linear model, only two types of traits are currently supported: 1) quantitative traits with a normal (gaussian) distribution and identity link, and 2) binomial traits with a logit-link function. The effects of haplotypes on the link function can be modeled as either additive, dominant (heterozygotes and homozygotes for a particular haplotype assumed to have equivalent effects), or recessive (homozygotes of a particular haplotype considered to have an alternative effect on the trait). The basis of the algorithm is a two-step iteration process; the posterior probabilities of pairs of haplotypes per subject are used as weights to update the regression coefficients, and the regression coefficients are used to update the posterior probabilities. See Lake et al.[5] for details.

8.1 Setting Up the data.frame

A critical distinction between haplo.glm and all other functions in Haplo Stats is that the definition of the regression model follows the S-PLUS/R formula standard. So, a data.frame must be defined, and this data.frame must contain the trait, a special kind of genotype matrix (called geno in this example) that contains the genotypes of the marker loci, and possibly other covariates and weights for the subjects. The key feature of this data.frame is how geno is created. Prepare the geno matrix with the function setupGeno, which handles character, numeric, or factor alleles, and keeps the columns of geno as a single unit when inserting into (and extracting from) a data.frame. The setupGeno function recodes alleles to integer values (the allele codes become an attribute of the returned object), and returns a model.matrix, which can then be inserted into a data.frame. Prepare the geno matrix, then create a data.frame for use in haplo.glm.

```
> geno <- as.matrix(hla.demo[, c(17, 18, 21:24)])
> geno <- setupGeno(geno, miss.val = c(0, NA))
> y.bin <- 1 * (hla.demo$resp.cat == "low")
> my.data <- data.frame(geno, age = age, male = male, y = resp,
+     y.bin = y.bin)
```

8.2 Regression for a Quantitative Trait

The following illustrates how to fit a regression of quantitative trait y on the haplotypes defined by the geno matrix, and the covariate male. The control parameter, haplo.freq.min, is discussed below under the heading **Explanation of Results**, as well as in section 8.5. The results are shown below.

```
> fit.gaus <- haplo.glm(y ~ male + geno, family = gaussian, data = my.data,
+     na.action = "na.geno.keep", locus.label = label, allele.lev = attributes(g
+     control = haplo.glm.control(haplo.freq.min = 0.02))
> print(fit.gaus)

Call:
haplo.glm(formula = y ~ male + geno, family = gaussian, data = my.data,
    na.action = "na.geno.keep", locus.label = label, allele.lev = attributes(geno)
    control = haplo.glm.control(haplo.freq.min = 0.02))
```

```
Coefficients:
               coef    se t.stat    pval
(Intercept)  1.0644 0.343  3.105 0.00217
male         0.0974 0.155  0.627 0.53119
geno.17      0.2802 0.435  0.643 0.52062
geno.34     -0.3171 0.343 -0.923 0.35684
geno.77      0.2217 0.361  0.614 0.54014
geno.78      1.1414 0.384  2.974 0.00328
geno.100     0.5556 0.364  1.525 0.12871
geno.138     0.9823 0.303  3.239 0.00139
geno.rare    0.3976 0.182  2.186 0.02992

Haplotypes:
           DQB DRB  B hap.freq
geno.17     21   7 44   0.0229
geno.34     31   4 44   0.0286
geno.77     32   4 60   0.0302
geno.78     32   4 62   0.0239
geno.100    51   1 35   0.0301
geno.138    62   2  7   0.0502
geno.rare    *   *  *   0.7100
haplo.base  21   3  8   0.1041
```

Explanation of Results

The above table for Coefficients lists the estimated regression coefficient (coef), its standard error (se), the corresponding t-statistic (t.stat) , and p-value (pval). The labels for haplotype coefficients are a pasting of the matrix defining the genotypes (geno in the above example) and the haplotype numbers. The haplotypes corresponding to these haplotype numbers are listed in the above table under Haplotypes, along with the estimates of the haplotype frequencies (hap.freq). The rare haplotypes (those with frequencies less than haplo.freq.min = 0.02 in the above example) are pooled into a single category labeled geno.rare. The haplo.freq.min parameter works the same as skip.haplo described in section 7.5. The haplotype chosen as the base-line category for the design matrix (most frequent haplotype is the default) is labeled as haplo.base.

8.3 Fitting Haplotype x Covariate Interactions

Interactions are fit by the standard S-language model syntax, using a "*" in the model formula to indicate main effects and interactions.

```
> fit.inter <- haplo.glm(y ~ male * geno, family = gaussian, data = my.data,
+     na.action = "na.geno.keep", locus.label = label, allele.lev = attributes(g
+     control = haplo.glm.control(haplo.freq.min = 0.02))
> print(fit.inter)

Call:
haplo.glm(formula = y ~ male * geno, family = gaussian, data = my.data,
    na.action = "na.geno.keep", locus.label = label, allele.lev = attributes(geno)
    control = haplo.glm.control(haplo.freq.min = 0.02))

Coefficients:
                  coef    se  t.stat    pval
(Intercept)     0.9754 0.301  3.2354 0.00142
male            0.2581 0.311  0.8308 0.40705
geno.17         0.1444 0.443  0.3257 0.74495
geno.34        -0.1716 0.596 -0.2878 0.77381
geno.77         0.8052 0.591  1.3627 0.17447
geno.78         0.4956 0.488  1.0157 0.31099
geno.100        0.5231 0.400  1.3093 0.19189
geno.138        1.1570 0.353  3.2801 0.00122
geno.rare       0.4555 0.182  2.5015 0.01315
male:geno.17    0.5087 0.711  0.7157 0.47502
male:geno.34   -0.2814 0.667 -0.4218 0.67365
male:geno.77   -0.9008 0.697 -1.2925 0.19764
male:geno.78    1.2638 0.654  1.9335 0.05456
male:geno.100   0.0507 0.660  0.0769 0.93881
male:geno.138  -0.4459 0.519 -0.8584 0.39168
male:geno.rare -0.0979 0.205 -0.4768 0.63402

Haplotypes:
           DQB DRB  B hap.freq
geno.17     21   7 44   0.0235
geno.34     31   4 44   0.0285
geno.77     32   4 60   0.0306
```

```
geno.78     32   4 62   0.0241
geno.100    51   1 35   0.0301
geno.138    62   2  7   0.0505
geno.rare    *   *  *   0.7086
haplo.base  21   3  8   0.1041
```

Explanation of Results

The listed results are as explained under section 8.2. The only difference is that the interaction coefficients are labeled as a pasting of the covariate (male in this example) and the name of the haplotype.

8.4 Regression for a Binomial Trait

The following illustrates the fitting of a binomial trait:

```
> fit.bin <- haplo.glm(y.bin ~ male + geno, family = binomial,
+     data = my.data, na.action = "na.geno.keep", locus.label = label,
+     allele.lev = attributes(geno)$unique.alleles, control = haplo.glm.control(
+     x = TRUE)
> print(fit.bin)

Call:
haplo.glm(formula = y.bin ~ male + geno, family = binomial, data = my.data,
    na.action = "na.geno.keep", locus.label = label, allele.lev = attributes(geno)
    control = haplo.glm.control(haplo.freq.min = 0.02), x = TRUE)

Coefficients:
              coef    se t.stat     pval
(Intercept)  1.546 0.419  3.686 2.90e-04
male        -0.480 0.324 -1.482 1.40e-01
geno.17     -0.723 0.696 -1.039 3.00e-01
geno.34      0.364 0.623  0.585 5.59e-01
geno.77     -0.988 0.689 -1.435 1.53e-01
geno.78     -1.409 0.773 -1.823 6.97e-02
geno.100    -2.591 0.709 -3.656 3.23e-04
geno.138    -2.716 0.758 -3.582 4.23e-04
geno.rare   -1.261 0.253 -4.975 1.35e-06
```

```
Haplotypes:
           DQB DRB  B hap.freq
geno.17     21   7 44   0.0230
geno.34     31   4 44   0.0284
geno.77     32   4 60   0.0306
geno.78     32   4 62   0.0235
geno.100    51   1 35   0.0298
geno.138    62   2  7   0.0518
geno.rare    *   *  *   0.7088
haplo.base  21   3  8   0.1041
```

Explanation of Results

The underlying methods for haplo.glm are based on a prospective likelihood. Normally, this type of likelihood works well for case-control studies with standard covariates. For ambiguous haplotypes, however, one needs to be careful when interpreting the results from fitting haplo.glm to case-control data. Because cases are over-sampled, relative to the population prevalence (or incidence, for incidence cases), haplotypes associated with disease will be over-represented in the case sample, and so estimates of haplotype frequencies will be biased. Positively associated haplotypes will have haplotype frequency estimates that are higher than the population haplotype frequency. To avoid this problem, one can weight each subject. The weights for the cases should be the population prevalence, and the weights for controls should be 1 (assuming the disease is rare in the population, and controls are representative of the general population). See Stram[7] for background on using weights, and see the help file for haplo.glm for how to implement weights.

The estimated regression coefficients for case-control studies can be biased by either a large amount of haplotype ambiguity and mis-specified weights, or by departures from Hardy Weinberg equilibrium of the haplotypes in the pool of cases and controls. Generally, the bias is small, but tends to be towards the null of no association. See Stram[7] and Epstein[3] for further details.

8.5 Control Parameters and Genetic Models

A key parameter for haplo.glm is control, which is a list of parameters that control the procedures of haplo.glm. This control list is set up by the function haplo.glm.control. One parameter in the control function is haplo.effect, which instructs whether the haplotype effects are fit as additive, dominant, or recessive. That is, haplo.effect determines whether the covariate (x) coding of haplotypes is "additive" (causing $x = 0$, 1, or 2, the count of a particular haplotype), "dominant" (causing $x = 1$ if heterozygous or homozygous carrier of a particular haplotype; x = 0 otherwise), or "recessive" (causing $x = 1$ if homozygous for a particular haplotype; $x = 0$ otherwise). See the help file for haplo.glm.control for further control parameters. The example below illustrates the fit of a "dominant" effect of haplotypes.

```
> fit.dom <- haplo.glm(y ~ male + geno, family = gaussian, data = my.data,
+     na.action = "na.geno.keep", locus.label = label, allele.lev = attributes(g
+     control = haplo.glm.control(haplo.effect = "dom", haplo.freq.min = 0.02))
> print(fit.dom)

Call:
haplo.glm(formula = y ~ male + geno, family = gaussian, data = my.data,
    na.action = "na.geno.keep", locus.label = label, allele.lev = attributes(geno)
    control = haplo.glm.control(haplo.effect = "dom", haplo.freq.min = 0.02))

Coefficients:
               coef    se t.stat     pval
(Intercept)  1.6493 0.373  4.416 1.61e-05
male         0.0797 0.157  0.507 6.13e-01
geno.17     -0.0604 0.423 -0.143 8.87e-01
geno.34     -0.6650 0.364 -1.827 6.91e-02
geno.77     -0.0734 0.347 -0.212 8.33e-01
geno.78      0.8537 0.364  2.344 2.00e-02
geno.100     0.2470 0.346  0.715 4.76e-01
geno.138     0.6729 0.282  2.389 1.78e-02
geno.rare    0.1120 0.340  0.329 7.42e-01

Haplotypes:
           DQB DRB  B hap.freq
geno.17     21   7 44   0.0230
```

geno.34	31	4	44	0.0286
geno.77	32	4	60	0.0302
geno.78	32	4	62	0.0239
geno.100	51	1	35	0.0300
geno.138	62	2	7	0.0502
geno.rare	*	*	*	0.7100
haplo.base	21	3	8	0.1041

9 License and Warranty

License:

Daniel J. Schaid, Ph.D.
Division of Biostatistics
Harwick Building - Room 775
Mayo Clinic
200 First St., SW
Rochester, MN 55905
phone: 507-284-0639
fax: 507-284-9542
email: schaid@mayo.edu

10 Acknowledgements

This research was supported by United States Public Health Services, National Institutes of Health; Contract grant numbers R01 DE13276, R01 GM 65450, N01 AI45240, and R01 2AI33144. The hla.demo data is kindly provided by Gregory A. Poland, M.D. and the Mayo Vaccine Research Group for illustration only, and may not be used for publication.

Appendix

A Counting Haplotype Pairs When Marker Phenotypes Have Missing Alleles

The following describes the process for counting the number of haplotype pairs that are consistent with a subject's observed marker phenotypes, allowing for some loci with missing data. Note that we refer to marker phenotypes, but our algorithm is oriented towards typical markers that have a one-to-one correspondence with their genotypes. We first describe how to count when none of the loci have missing alleles, and then generalize to allow loci to have either one or two missing alleles. When there are no missing alleles, note that homozygous loci are not ambiguous with respect to the underlying haplotypes, because at these loci the underlying haplotypes will not differ if we interchange alleles between haplotypes. In contrast, heterozygous loci are ambiguous, because we do not know the haplotype origin of the distinguishable alleles (i.e., unknown linkage phase). However, if there is only one heterozygous locus, then it doesn't matter if we interchange alleles, because the pair of haplotypes will be the same. In this situation, if parental origin of alleles were known, then interchanging alleles would switch parental origin of haplotypes, but not the composition of the haplotypes. Hence, ambiguity arises only when there are at least two heterozygous loci. For each heterozygous locus beyond the first one, the number of possible haplotypes increases by a factor of 2, because we interchange the two alleles at each heterozygous locus to create all possible pairs of haplotypes. Hence, the number of possible haplotype pairs can be expressed as 2^x, where $x = H - 1$, if H (the number of heterozygous loci) is at least 2, otherwise $x = 0$.

Now consider a locus with missing alleles. The possible alleles at a given locus are considered to be those that are actually observed in the data. Let a_i denote the number of distinguishable alleles at the locus. To count the number of underlying haplotypes that are consistent with the observed and missing marker data, we need to enumerate all possible genotypes for the loci with missing data, and consider whether the imputed genotypes are heterozygous or homozygous.

To develop our method, first consider how to count the number of genotypes at a locus, say the i^{th} locus, when either one or two alleles are missing.

This locus could have either a homozygous or heterozygous genotype, and both possibilities must be considered for our counting method. If the locus is considered as homozygous, and there is one allele missing, then there is only one possible genotype; if there are two alleles missing, then there are a_i possible genotypes. A function to perform this counting for homozygous loci is denoted $f(a_i)$. If the locus is considered as heterozygous, and there is one allele missing, then there are $a_i - 1$ possible genotypes; if there are two alleles missing, then there are $\frac{a_i(a_i-1)}{2}$ possible genotypes. A function to perform this counting for heterozygous loci is denoted $g(a_i)$ These functions and counts are summarized in Table A.1.

Table A.1: Factors for when a locus having missing allele(s) is counted as homozygous($f()$) or heterozygous($g()$)

Number of missing alleles	Homozygous function $f(a_i)$	Heterozygous function $g(a_i)$
1	1	$a_i - 1$
2	a_i	$\frac{a_i(a_i-1)}{2}$

Now, to use these genotype counting functions to determine the number of possible haplotype pairs, first consider a simple case where only one locus, say the i^{th} locus, has two missing alleles. Suppose that the phenotype has H heterozygous loci (H is the count of heterozygous loci among those without missing data). We consider whether the locus with missing data is either homozygous or heterozygous, to give the count of possible haplotype pairs as

$$a_i 2^x + \left[\frac{a_i(a_i-1)}{2}\right] 2^{x+1} \tag{1}$$

where again $x = H - 1$ if H is at least 2, otherwise x = 0. This special case can be represented by our more general genotype counting functions as

$$f(a_i)\, 2^x + g(a_i)\, 2^{x+1} \tag{2}$$

When multiple loci have missing data, we need to sum over all possible combinations of heterozygous and homozygous genotypes for the incomplete loci. The rows of Table A.2 below present these combinations for up to

$m = 3$ loci with missing data. Note that as the number of heterozygous loci increases (across the columns of Table A.2), so too does the exponent of 2. To calculate the total number of pairs of haplotypes, given observed and possibly missing genotypes, we need to sum the terms in Table A.2 across the appropriate row. For example, with $m = 3$, there are eight terms to sum over. The general formulation for this counting method can be expressed as

$$TotalPairs = \sum_{j=0}^{m} \sum_{combo} C(combo, j) \tag{3}$$

where combo is a particular pattern of heterozygous and homozygous loci among the loci with missing values (e.g., for $m = 3$, one combination is the first locus heterozygous and the 2^{nd} and 3^{rd} third as homozygous), and $C(combo, j)$ is the corresponding count for this pattern when there are i loci that are heterozygous (e.g., for $m = 3$ and $j = 1$, as illustrated in Table A.2).

Table A.2: Genotype counting terms when m loci have missing alleles, grouped by number of heterozygous loci (out of m)

m	$j = 0\, of\, m$	$j = 1\, of\, m$	$j = 2\, of\, m$	$j = 3\, of\, m$
0	2^x			
1	$f(a_1)2^x$	$g(a_1)2^{x+1}$		
2	$f(a_1)f(a_2)2^x$	$g(a_1)f(a_2)2^{x+1}$ $f(a_1)g(a_2)2^{x+1}$	$g(a_1)g(a_2)2^{x+1}$	
3	$f(a_1)f(a_2)f(a_3)2^x$	$g(a_1)f(a_2)f(a_3)2^{x+1}$ $f(a_1)g(a_2)f(a_3)2^{x+1}$ $f(a_1)f(a_2)g(a_3)2^{x+1}$	$g(a_1)g(a_2)f(a_3)2^{x+2}$ $g(a_1)f(a_2)g(a_3)2^{x+2}$ $f(a_1)g(a_2)g(a_3)2^{x+2}$	$g(a_1)g(a_2)g(a_3)2^{x+2}$

References

[1] Besag J, Clifford P (1991) Sequential Monte Carlo p-Values. *Biometrika* 78:301-304

[2] Clayton, David. Personal web page, software list. April 1, 2004. <http://www-gene.cimr.cam.ac.uk/clayton/software/>.

[3] Epstein M, Satten G (2003) Inference on haplotype effects in case-control studies using unphased genotype data. Submitted.

[4] Harrell, FE. Regression Modeling Strategies, Springer-Verlag, NY, 2001.

[5] Lake S, Lyon H, Silverman E, Weiss S, Laird N, Schaid D (2003) Estimation and tests of haplotype-environment interaction when linkage phase is ambiguous. *Human Heredity* 55:56-65

[6] Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA (2002) Score tests for association between traits and haplotypes when linkage phase is ambiguous. *Am J Hum Genet* 70:425-34

[7] Stram D, Pearce C, Bretsky P, Freedman M, Hirschhorn J, Altshuler D, Kolonel L, Henderson B, Thomas D (2003) Modeling and E-M estimation of haplotype-specific relative risks from genotype data for case-control study of unrelated individuals. *Hum Hered* 55:179-190