Haplo Stats

(version 1.2.1)

Statistical Methods for Haplotypes When Linkage Phase is Ambiguous

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

May 9, 2005

Contents

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's for the regression of a trait on haplotypes, with the option of 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 1.2.1 package is written for both S-PLUS (version 6.2.1) and R (version 2.1.0) for Unix. It has been placed on the Comprehensive R Archive Network (CRAN), and is made available on other systems through CRAN. 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 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.

For users who are new to the S-PLUS or R environments, note the following basic concepts. In the following examples, a user enters the indented text following the prompt ">", and the output results follow. Later, when executing a function in the session, the general syntax will appear like *'myresult <- myfunction(x)'* where the results of *myfunction*, operating on *x*, are saved in *myresult*. Then a user may print *myresult* or make use of it in a calculation.

3.1 Example Data

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

```
## if local library, use:
        ## library(haplo.stats, lib.loc="/local/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"
 [6] "DPB.a2"   "DPA.a1"   "DPA.a2"   "DMA.a1"   "DMA.a2"
[11] "DMB.a1"   "DMB.a2"   "TAP1.a1"  "TAP1.a2"  "TAP2.a1"
[16] "TAP2.a2"  "DQB.a1"   "DQB.a2"   "DQA.a1"   "DQA.a2"
[21] "DRB.a1"   "DRB.a2"   "B.a1"     "B.a2"     "A.a1"
[26] "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.

3.2 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")
```

3.3 Random Numbers and Setting Seed

Random numbers 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 the exact results in this user guide, you must set the *.Random.seed* before any function which uses random numbers (*haplo.em, haplo.score, haplo.glm, haplo.group, haplo.cc*) using these steps. We illustrate this below, and invisibly reset the seed to the same vector in making the rest of 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 version 2.1.0 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 4.4).

3.4 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. The saved result, *geno.desc* is a data frame, so 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 4.2. With *geno* rows #81 and #137 included, *haplo.em* requires about 4 minutes of CPU time, while without those two rows it takes just over 1 second. It is best to use the information provided by subjects with missing alleles, but the results from *summaryGeno* can guide which subjects could be removed if computation issues arise.

4 Haplotype Frequency Estimation: *haplo.em*

4.1 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[4]. 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.

4.2 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*).

4.3 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       25      168   1.00000
5       4       13       39   0.28621
```

```
6       4       17       38   0.71379
7       5       94       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
```

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
78         1       32        4     62       31       11
13         2       21        7      7       62        2
138        2       62        2      7       21        7
25         3       31        1     27       63       13
13.1       4       21        7      7       31        7
17         4       21        7     44       31        7
94         5       42        8     55       31       11
     hap2.B posterior
78       61   1.00000
13       44   0.12532
138      44   0.87468
```

```
25       62   1.00000
13.1     44   0.28621
17        7   0.71379
94       51   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
```

4.4 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 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 control parameters 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*). 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 the syntax for setting the number of tries to 20, and the batch size to 2.

```
> save.em <- haplo.em(geno = geno, locus.label = label,
+     miss.val = c(0, NA), control = haplo.em.control(n.try = 20,
+         insert.batch.size = 2))
```

4.5 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.bin <- 1 * (hla.demo$resp.cat == "low")
> 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.

5 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.[8].

5.1 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 5.5. 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 = 5/(2 * nrow(geno)), locus.label = label,
+     simulate = FALSE)
> print(score.gaus, nlines = 10)

 ------------------------------------------------------------
                  Global Score Statistics
 ------------------------------------------------------------

 global-stat = 30.6353, df = 18, p-val = 0.03171



 ------------------------------------------------------------
                 Haplotype-specific Scores
 ------------------------------------------------------------

       DQB DRB B  Hap-Freq Hap-Score p-val
  [1,] 21  3   8  0.10408  -2.39631  0.01656
  [2,] 31  4   44 0.02849  -2.24273  0.02491
  [3,] 51  1   44 0.01731  -0.99357  0.32043
  [4,] 63  13  44 0.01606  -0.84453  0.39837
  [5,] 63  2   7  0.01333  -0.50736  0.6119
  [6,] 32  4   60 0.0306   -0.46606  0.64118
  [7,] 21  7   44 0.02332  -0.41942  0.67491
  [8,] 62  2   44 0.01367  -0.26221  0.79316
  [9,] 62  2   18 0.01545  -0.21493  0.82982
 [10,] 51  1   27 0.01505  0.01539   0.98772
```

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. Note, the score statistic should not be interpreted as a measure of the haplotype effect.

- *p-val* is the asymptotic chi-square (1 df) p-value, calculated from the square of the score statistic.

5.2 Ordinal Trait Analysis

To create an ordinal trait, convert *resp.cat* (described above) 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 = 5/(2 *
+         nrow(geno)), locus.label = label, miss.val = 0,
+     simulate = FALSE)
> print(score.ord, nlines = 7)

 ------------------------------------------------------------
                  Global Score Statistics
 ------------------------------------------------------------

 global-stat = 35.06701, df = 18, p-val = 0.00927



 ------------------------------------------------------------
                 Haplotype-specific Scores
 ------------------------------------------------------------

     DQB DRB B  Hap-Freq Hap-Score p-val
 [1,] 21  3   8  0.10408  -2.79247  0.00523
 [2,] 31  4   44 0.02849  -2.61319  0.00897
 [3,] 63  13  44 0.01606  -0.69172  0.48911
 [4,] 51  1   44 0.01731  -0.62185  0.53404
 [5,] 62  2   18 0.01545  -0.51357  0.60755
 [6,] 21  7   44 0.02332  -0.28576  0.77506
 [7,] 32  4   60 0.0306   -0.18264  0.85508
```

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.[6]. 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.

5.3 Binary Trait Analysis

Let us assume that "low" responders are of primary interest, so we create a binary trait that has values of 1 when *resp.cat* is "low", and 0 otherwise. Then in *haplo.score* specify the parameter *trait.type="binomial"*.

```
> y.bin <- 1 * (hla.demo$resp.cat == "low")
> score.bin <- haplo.score(y.bin, geno, trait.type = "binomial",
+     offset = NA, x.adj = NA, skip.haplo = 5/(2 *
+         nrow(geno)), locus.label = label, miss.val = 0,
+     simulate = FALSE)
> print(score.bin, nlines = 10)

 ------------------------------------------------------------
                  Global Score Statistics
 ------------------------------------------------------------

 global-stat = 33.70125, df = 18, p-val = 0.01371



 ------------------------------------------------------------
                 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,] 32  4   7  0.01678  -1.00091  0.31687
  [6,] 32  4   62 0.02349  -0.6799   0.49657
  [7,] 51  1   27 0.01505  -0.66509  0.50599
  [8,] 31  11  35 0.01754  -0.5838   0.55936
  [9,] 31  11  51 0.01137  -0.43721  0.66196
 [10,] 51  1   44 0.01731  0.00826   0.99341
```

5.4 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*). This plot, and the syntax for creating it, are shown in Figure 1 at the end of this manual.

Some points on the plot may be of interest. To identify individual points on the plot, use *locator.haplo(score.gaus)*, which is similar to *locator()*. Use the mouse to select points on the plot. After points are chosen, click on the middle mouse button, and the points are labeled with their haplotype labels. Note, in contructing Figure 1, we had to define which points to label, and then assign labels in the same way as done within the *locator.haplo* function.

5.5 Skipping Rare Haplotypes

For the quantitative trait analyses, the *skip.haplo* parameter controls which rare haplotypes are pooled into a common group. As a guideline, you may wish to set *skip.haplo* to calculate scores for haplotypes with expected haplotype counts of 5 or greater. We concentrate on this expected count because it adjusts to the size of the input data. If N is the number of subjects and f the haplotype frequency, then the expected haplotype count is $count = 2 \times N \times f$. So you can choose $skip.haplo = \frac{count}{2 \times N}$. Here we try a different cut-off than before, *skip.haplo=.02*, which corresponds to expected haplotype counts of $2 \times 220 \times .02 = 8.8$. In the output, notice the global statistic and its p-value change because of the fewer haplotypes, while the haplotype-specific scores change only slightly.

```
> score.gaus.02 <- haplo.score(resp, geno, trait.type = "gaussian",
+     offset = NA, x.adj = NA, skip.haplo = 0.02,
+     locus.label = label, miss.val = 0, simulate = FALSE)
> print(score.gaus.02)

 ------------------------------------------------------------
                  Global Score Statistics
 ------------------------------------------------------------

 global-stat = 20.66451, df = 7, p-val = 0.0043



 ------------------------------------------------------------
                 Haplotype-specific Scores
 ------------------------------------------------------------

     DQB DRB B  Hap-Freq Hap-Score p-val
[1,] 21  3   8  0.10408  -2.39631  0.01656
```

```
[2,] 31  4   44 0.02849  -2.24273  0.02491
[3,] 32  4   60 0.0306   -0.46606  0.64118
[4,] 21  7   44 0.02332  -0.41942  0.67491
[5,] 51  1   35 0.03018  0.69696   0.48583
[6,] 32  4   62 0.02349  2.37619   0.01749
[7,] 62  2   7  0.05098  2.39795   0.01649
```

5.6 Haplotype Scores, Adjusted for Covariates

First set up a matrix of covariates, 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 = 5/(2 *
+         nrow(geno)), locus.label = label, miss.val = 0,
+     simulate = FALSE)
> print(score.gaus.adj, nlines = 10)

 ------------------------------------------------------------
                  Global Score Statistics
 ------------------------------------------------------------

 global-stat = 31.02908, df = 18, p-val = 0.02857



 ------------------------------------------------------------
                Haplotype-specific Scores
 ------------------------------------------------------------

      DQB DRB B  Hap-Freq Hap-Score p-val
 [1,] 21  3   8  0.10408  -2.4097   0.01597
 [2,] 31  4   44 0.02849  -2.25293  0.02426
 [3,] 51  1   44 0.01731  -0.98763  0.32333
 [4,] 63  13  44 0.01606  -0.83952  0.40118
 [5,] 63  2   7  0.01333  -0.48483  0.6278
 [6,] 32  4   60 0.0306   -0.46476  0.64211
 [7,] 21  7   44 0.02332  -0.41249  0.67998
 [8,] 62  2   44 0.01367  -0.26443  0.79145
```

```
 [9,] 62  2   18 0.01545  -0.20425  0.83816
[10,] 51  1   27 0.01505  0.02243   0.9821
```

5.7 Simulation p-values

When *simulate=TRUE*, *haplo.score* gives simulated p-values. Simulated haplotype score statistics are the re-calculated score statistics from a permuted re-ordering of the trait and covariates and the original ordering of the genotype matrix. The simulated p-value for the global score statistic (*Global sim. p-val*) is the number of times the simulated global score statistic exceeds the observed, divided by the total number of simulations. Likewise, simulated p-value for the maximum score statistic (*Max-stat sim. p-val*) is the number of times the simulated maximum haplotype score statistic exceeds the observed maximum score statistic, divided by the total number of simulations. The maximum score statistic is the maximum of the square of the haplotype-specific score statistics, which has an unknown distribution, so its significance can only be given by the simulated p-value. Intuitively, if only one or two haplotypes are associated with the trait, the maximum score statistic should have greater power to detect association than the global statistic.

The *score.sim.control* function manages control parameters for simulations. The *haplo.score* function employs the simulation p-value precision criteria of Besag and Clifford[1]. These criteria ensure that the simulated 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-stat* and *max-stat* 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 simulation p-values with *min.sim=1000*.

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

-----------------------------------------------------------
                  Global Score Statistics
-----------------------------------------------------------

global-stat = 33.70125, df = 18, p-val = 0.01371

-----------------------------------------------------------
             Global Simulation p-value Results
-----------------------------------------------------------
```

```
Global sim. p-val =  0.0095
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,] 32  4   7  0.01678  -1.00091  0.31687 0.26882
 [6,] 32  4   62 0.02349  -0.6799   0.49657 0.53624
 [7,] 51  1   27 0.01505  -0.66509  0.50599 0.64286
 [8,] 31  11  35 0.01754  -0.5838   0.55936 0.59078
 [9,] 31  11  51 0.01137  -0.43721  0.66196 0.91872
[10,] 51  1   44 0.01731  0.00826   0.99341 1
[11,] 32  4   60 0.0306   0.03181   0.97462 0.95215
[12,] 62  2   44 0.01367  0.16582   0.8683  0.91661
[13,] 63  13  44 0.01606  0.22059   0.82541 0.73962
[14,] 63  2   7  0.01333  0.2982    0.76555 0.77164
[15,] 62  2   18 0.01545  0.78854   0.43038 0.6608
[16,] 21  7   44 0.02332  0.84562   0.39776 0.3962
[17,] 31  4   44 0.02849  2.50767   0.01215 0.01196
[18,] 21  3   8  0.10408  3.77763   0.00016 0.00035
```

6 Regression Models: *haplo.glm*

The *haplo.glm* function computes the regression of a trait on haplotypes, and possibly other covariates and their interactions with haplotypes. Although 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 haplotype posterior probabilities. See Lake et al.[7] for details.

6.1 Preparing the *data.frame* for *haplo.glm*

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 (see *lm glm*). So, a *data.frame* must be defined, and this object must contain the trait, a special kind of genotype matrix (*geno.glm* for this example) that contains the genotypes of the marker loci, and optionally other covariates and weights. The key features of this *data.frame* are in how we set up a genotype matrix specific for use in *haplo.glm*. We prepare *geno.glm* using *setupGeno*, which handles character, numeric, or factor alleles, and keeps the columns of the genotype matrix as a single unit when inserting into (and extracting from) a *data.frame*. The *setupGeno* function also recodes alleles to integer values (the initial allele codes become an attribute of the returned object), and returns a *model.matrix*, which can then be included in a *data.frame*. In the example below we prepare *geno.glm*, then create a *data.frame* object (*my.data*) for use in *haplo.glm*.

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

6.2 Handling Rare Haplotypes

An issue in *haplo.glm* is to decide which haplotypes to put in the model. We have used the *haplo.freq.min* parameter as a cut-off for the haplotypes to be modeled based on their frequency. However, we have found both haplotype effect and corresponding standard errors to be unreliable for a haplotype with a low frequency in the sample. Therefore, a cut-off should be chosen based on the expected count of the haplotype in the sample. The default for choosing a cut-off is the same for setting *skip.haplo* in *haplo.score*, where the minimum haplotype frequency $f = \frac{count}{2 \times N}$, where N is the number of subjects, and *count* the expected count of haplotypes in the sample. This calculation is based on the formula for the expected count, $count = f \times 2 \times N$.

The default minimum frequency cut-off in *haplo.glm* is based on a minimum expected count of 5. Two control parameters may be used to control this setting. The previously used *haplo.freq.min* may be set to a different minimum haplotype frequency. Alternatively, the *haplo.min.count* control parameter can be set by the user to select the cut-off minimum expected haplotype count.

6.3 Regression for a Quantitative Trait

The following illustrates how to fit a regression of a quantitative trait y on the haplotypes estimated from the *geno.glm* matrix, and the covariate *male*. For *na.action*, we use *na.geno.keep*, which allows missing values in the genotype matrix, but removes a subject with missing values (*NA*) in either

the response or covariate. The *setupGeno* function recoded alleles to numeric values in *geno.glm* numbered starting with 1, but we can preserve the original allele values by setting the *allele.lev* parameter to be the *unique.alleles* attribute from *geno.glm*.

```
> fit.gaus <- haplo.glm(y ~ male + geno.glm, family = gaussian,
+     data = my.data, na.action = "na.geno.keep",
+     locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
+     control = haplo.glm.control(haplo.min.count = 5))
> print(fit.gaus)

  Call:
haplo.glm(formula = y ~ male + geno.glm,
    family = gaussian, data = my.data, na.action = "na.geno.keep",
    locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
    control = haplo.glm.control(haplo.min.count = 5))

Coefficients:
                 coef    se  t.stat    pval
(Intercept)    0.9918 0.349  2.8393 0.00499
male           0.1281 0.161  0.7962 0.42684
geno.glm.13    1.1208 0.539  2.0791 0.03889
geno.glm.17    0.2713 0.441  0.6155 0.53895
geno.glm.34   -0.2573 0.347 -0.7408 0.45970
geno.glm.50    0.7687 0.485  1.5846 0.11463
geno.glm.55    0.4538 0.566  0.8018 0.42364
geno.glm.69    1.1080 0.552  2.0057 0.04624
geno.glm.77    0.2336 0.355  0.6572 0.51178
geno.glm.78    1.2370 0.387  3.1928 0.00164
geno.glm.99    0.4800 0.501  0.9573 0.33957
geno.glm.100   0.6125 0.375  1.6342 0.10378
geno.glm.102  -0.1097 0.447 -0.2453 0.80650
geno.glm.138   0.9849 0.305  3.2342 0.00143
geno.glm.140   0.4224 0.482  0.8756 0.38228
geno.glm.143   0.0215 0.500  0.0430 0.96571
geno.glm.155   0.3706 0.522  0.7104 0.47830
geno.glm.162   1.3679 0.472  2.8974 0.00418
geno.glm.165   0.1172 0.460  0.2550 0.79896
geno.glm.rare  0.3936 0.189  2.0837 0.03846

Haplotypes:
              DQB DRB  B hap.freq
```

```
geno.glm.13    21   7  7   0.0124
geno.glm.17    21   7 44   0.0229
geno.glm.34    31   4 44   0.0286
geno.glm.50    31  11 35   0.0170
geno.glm.55    31  11 51   0.0114
geno.glm.69    32   4  7   0.0150
geno.glm.77    32   4 60   0.0319
geno.glm.78    32   4 62   0.0239
geno.glm.99    51   1 27   0.0150
geno.glm.100   51   1 35   0.0300
geno.glm.102   51   1 44   0.0176
geno.glm.138   62   2  7   0.0510
geno.glm.140   62   2 18   0.0154
geno.glm.143   62   2 44   0.0141
geno.glm.155   63   2  7   0.0136
geno.glm.162   63  13  7   0.0161
geno.glm.165   63  13 44   0.0165
geno.glm.rare   *   *  *   0.5434
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 concatenation of the name of the genotype matrix (*geno.glm*) and unique haplotype codes assigned within *haplo.glm*. The haplotypes corresponding to these haplotype codes are listed in the *Haplotypes* table, along with the estimates of the haplotype frequencies (*hap.freq*). The rare haplotypes, those with expected counts less than *haplo.min.count=5* (equivalent to having frequencies less than *haplo.freq.min = 0.01136* in the above example), are pooled into a single category labeled *geno.glm.rare*. The haplotype chosen as the baseline category for the design matrix (most frequent haplotype is the default) is labeled as *haplo.base*; more information on the baseline may be found in section 6.6.2.

6.4 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.glm, family = gaussian,
+     data = my.data, na.action = "na.geno.keep",
+     locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
```

```
+     control = haplo.glm.control(haplo.min.count = 5))
> print(fit.inter)

   Call:
 haplo.glm(formula = y ~ male * geno.glm,
     family = gaussian, data = my.data, na.action = "na.geno.keep",
     locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
     control = haplo.glm.control(haplo.min.count = 5))

 Coefficients:
                       coef    se t.stat     pval
 (Intercept)         0.6807 0.290  2.347 2.00e-02
 male                0.5003 0.318  1.575 1.17e-01
 geno.glm.13         0.5473 0.390  1.403 1.62e-01
 geno.glm.17         0.3003 0.469  0.640 5.23e-01
 geno.glm.34        -0.0409 0.593 -0.069 9.45e-01
 geno.glm.50         1.0612 0.451  2.355 1.96e-02
 geno.glm.55         0.8761 0.505  1.735 8.44e-02
 geno.glm.69         0.9983 0.337  2.965 3.43e-03
 geno.glm.77         0.9384 0.591  1.588 1.14e-01
 geno.glm.78         0.6302 0.506  1.244 2.15e-01
 geno.glm.99         0.5981 0.462  1.293 1.97e-01
 geno.glm.100        0.7198 0.399  1.803 7.31e-02
 geno.glm.102       -0.1355 0.501 -0.271 7.87e-01
 geno.glm.138        1.3569 0.357  3.801 1.96e-04
 geno.glm.140        0.3777 0.447  0.846 3.99e-01
 geno.glm.143       -0.8084 0.580 -1.394 1.65e-01
 geno.glm.155        1.4905 0.551  2.706 7.45e-03
 geno.glm.162        1.4008 0.453  3.090 2.31e-03
 geno.glm.165        0.0519 0.296  0.175 8.61e-01
 geno.glm.rare       0.6155 0.197  3.131 2.03e-03
 male:geno.glm.13    1.1326 0.308  3.681 3.06e-04
 male:geno.glm.17    0.4201 0.746  0.563 5.74e-01
 male:geno.glm.34   -0.3481 0.676 -0.515 6.07e-01
 male:geno.glm.50   -1.2600 0.134 -9.374 0.00e+00
 male:geno.glm.55   -1.2429 0.177 -7.008 4.56e-11
 male:geno.glm.69    0.4219 0.314  1.342 1.81e-01
 male:geno.glm.77   -1.0033 0.694 -1.445 1.50e-01
 male:geno.glm.78    1.1132 0.697  1.596 1.12e-01
 male:geno.glm.99   -0.2310 0.292 -0.792 4.29e-01
 male:geno.glm.100  -0.0882 0.631 -0.140 8.89e-01
```

```
male:geno.glm.102   0.2329 0.668  0.349 7.28e-01
male:geno.glm.138  -0.6347 0.511 -1.241 2.16e-01
male:geno.glm.140   1.2916 0.120 10.765 0.00e+00
male:geno.glm.143   1.6021 0.828  1.934 5.46e-02
male:geno.glm.155  -2.0260 0.725 -2.795 5.74e-03
male:geno.glm.162  -0.2029 0.392 -0.518 6.05e-01
male:geno.glm.165   0.1541 0.259  0.596 5.52e-01
male:geno.glm.rare -0.2787 0.236 -1.183 2.38e-01

Haplotypes:
              DQB DRB  B hap.freq
geno.glm.13    21   7  7   0.0127
geno.glm.17    21   7 44   0.0232
geno.glm.34    31   4 44   0.0285
geno.glm.50    31  11 35   0.0168
geno.glm.55    31  11 51   0.0114
geno.glm.69    32   4  7   0.0140
geno.glm.77    32   4 60   0.0320
geno.glm.78    32   4 62   0.0243
geno.glm.99    51   1 27   0.0149
geno.glm.100   51   1 35   0.0300
geno.glm.102   51   1 44   0.0178
geno.glm.138   62   2  7   0.0514
geno.glm.140   62   2 18   0.0154
geno.glm.143   62   2 44   0.0144
geno.glm.155   63   2  7   0.0136
geno.glm.162   63  13  7   0.0161
geno.glm.165   63  13 44   0.0166
geno.glm.rare   *   *  *   0.5427
haplo.base     21   3  8   0.1042
```

Explanation of Results

The listed results are as explained under section 6.3. The only difference is that the interaction coefficients are labeled as a concatenation of the covariate (*male* in this example) and the name of the haplotype as described above.

6.5 Regression for a Binomial Trait

Next we illustrate the fitting of a binomial trait with the same genotype matrix and covariate.

```
> fit.bin <- haplo.glm(y.bin ~ male + geno.glm,
+     family = binomial, data = my.data, na.action = "na.geno.keep",
+     locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
+     control = haplo.glm.control(haplo.min.count = 5))
> print(fit.bin)

   Call:
 haplo.glm(formula = y.bin ~ male + geno.glm,
     family = binomial, data = my.data, na.action = "na.geno.keep",
     locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
     control = haplo.glm.control(haplo.min.count = 5))

 Coefficients:
                   coef       se    t.stat     pval
 (Intercept)     1.740 3.35e-01  5.19e+00 5.22e-07
 male           -0.558 3.39e-01 -1.65e+00 1.01e-01
 geno.glm.13   -17.975 1.50e-08 -1.20e+09 0.00e+00
 geno.glm.17    -0.761 7.25e-01 -1.05e+00 2.95e-01
 geno.glm.34     0.250 6.22e-01  4.02e-01 6.88e-01
 geno.glm.50    -2.283 2.22e-01 -1.03e+01 0.00e+00
 geno.glm.55    -1.772 2.63e-01 -6.75e+00 1.61e-10
 geno.glm.69    -2.533 2.85e-01 -8.87e+00 4.44e-16
 geno.glm.77    -1.124 6.72e-01 -1.67e+00 9.58e-02
 geno.glm.78    -1.651 8.02e-01 -2.06e+00 4.10e-02
 geno.glm.99    -1.838 4.05e-01 -4.54e+00 9.78e-06
 geno.glm.100   -2.750 5.13e-01 -5.36e+00 2.33e-07
 geno.glm.102   -0.974 8.49e-01 -1.15e+00 2.52e-01
 geno.glm.138   -2.847 7.69e-01 -3.70e+00 2.79e-04
 geno.glm.140   -0.827 8.27e-01 -1.00e+00 3.18e-01
 geno.glm.143   -0.633 9.31e-01 -6.80e-01 4.97e-01
 geno.glm.155   -1.588 8.48e-01 -1.87e+00 6.27e-02
 geno.glm.162  -17.572 1.75e-08 -1.00e+09 0.00e+00
 geno.glm.165   -1.040 8.54e-01 -1.22e+00 2.24e-01
 geno.glm.rare  -1.241 2.45e-01 -5.08e+00 8.83e-07

 Haplotypes:
               DQB DRB  B hap.freq
 geno.glm.13    21   7  7   0.0129
 geno.glm.17    21   7 44   0.0226
 geno.glm.34    31   4 44   0.0286
 geno.glm.50    31  11 35   0.0170
```

```
geno.glm.55    31  11 51   0.0115
geno.glm.69    32   4  7   0.0169
geno.glm.77    32   4 60   0.0305
geno.glm.78    32   4 62   0.0236
geno.glm.99    51   1 27   0.0152
geno.glm.100   51   1 35   0.0298
geno.glm.102   51   1 44   0.0175
geno.glm.138   62   2  7   0.0515
geno.glm.140   62   2 18   0.0155
geno.glm.143   62   2 44   0.0140
geno.glm.155   63   2  7   0.0129
geno.glm.162   63  13  7   0.0159
geno.glm.165   63  13 44   0.0164
geno.glm.rare   *   *  *   0.5440
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 incident 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[9] 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[9] and Epstein[5] for further details.

6.6 Control Parameters

Additional parameters are handled using *control*, which is a list of parameters providing additional functionality in *haplo.glm*. This list is set up by the function *haplo.glm.control*. See the help file (*help(haplo.glm.control)*) for a full list of control parameters, with details of their usage. Some of the options are described here.

6.6.1 Controlling Genetic Models: *haplo.effect*

The *haplo.effect* control parameter for *haplo.glm* instructs whether the haplotype effects are fit as additive, dominant, or recessive. That is, *haplo.effect* determines whether the covariate (x) coding of haplotypes follows the values in **Table 1** for each effect type. Heterozygous means a subject has one copy of a particular haplotype, and homozygous means a subject has two copies of a particular haplotype.

Table 1: Coding haplotype covariates in a model matrix

Model Effects	Heterozygous	Homozygous
recessive	0	1
additive	1	2
dominant	1	1

Note that in a recessive model, the haplotype effects are estimated only from subjects who are homozygous for haplotype. This means that subjects who are homozygotes for the baseline haplotype and subjects who are heterozygous make up the baseline group. Some of the haplotypes which meet the *haplo.freq.min* and *haplo.count.min* cut-offs may occur as homozygous in only a few of the subjects. In that case, the estimated effect may be unreliable. Below we domonstrate how to use earlier results from *haplo.em* to find the number of homozygous subjects for each haplotype.

```
> is.homzyg <- save.em$hap1code == save.em$hap2code
> homzyg.counts <- table(save.em$hap1code[is.homzyg])
> homzyg.counts

 4 12 34
 2  1  1
```

The above calculation uses *hap1code* and *hap2code* from within *save.em*, which are haplotype codes for each subject's haplotype pair. The *homzyg.counts* result is a table showing the haplotype codes (row 1) and how many times they appear homozygous in any subject (row2). Since the haplotype coded as 4 is homozygous in two subjects, but none are over a count of five, so the recessive model is not appropriate for the *hla.demo* dataset.

The default *haplo.effect* is *additive*, whereas the example below illustrates the fit of a *dominant* effect of haplotypes for the gaussian trait with the gender covariate.

```
> fit.dom <- haplo.glm(y ~ male + geno.glm, family = gaussian,
+     data = my.data, na.action = "na.geno.keep",
+     locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
+     control = haplo.glm.control(haplo.effect = "dominant",
```

```
+         haplo.min.count = 5))
> print(fit.dom)

   Call:
 haplo.glm(formula = y ~ male + geno.glm,
     family = gaussian, data = my.data, na.action = "na.geno.keep",
     locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
     control = haplo.glm.control(haplo.effect = "dominant",
         haplo.min.count = 5))

 Coefficients:
                    coef    se  t.stat     pval
 (Intercept)    1.42992 0.318  4.4908 1.20e-05
 male           0.12606 0.163  0.7755 4.39e-01
 geno.glm.13    0.84568 0.521  1.6223 1.06e-01
 geno.glm.17    0.02521 0.426  0.0592 9.53e-01
 geno.glm.34   -0.52496 0.371 -1.4133 1.59e-01
 geno.glm.50    0.50410 0.467  1.0791 2.82e-01
 geno.glm.55    0.18271 0.556  0.3286 7.43e-01
 geno.glm.69    0.79200 0.529  1.4964 1.36e-01
 geno.glm.77   -0.00494 0.337 -0.0147 9.88e-01
 geno.glm.78    0.99835 0.371  2.6931 7.68e-03
 geno.glm.99    0.27543 0.497  0.5546 5.80e-01
 geno.glm.100   0.37100 0.362  1.0254 3.06e-01
 geno.glm.102  -0.35014 0.433 -0.8079 4.20e-01
 geno.glm.138   0.74078 0.282  2.6289 9.23e-03
 geno.glm.140   0.15969 0.467  0.3417 7.33e-01
 geno.glm.143  -0.19277 0.502 -0.3837 7.02e-01
 geno.glm.155   0.02909 0.500  0.0582 9.54e-01
 geno.glm.162   1.14736 0.464  2.4708 1.43e-02
 geno.glm.165  -0.11327 0.449 -0.2525 8.01e-01
 geno.glm.rare  0.20226 0.260  0.7791 4.37e-01

 Haplotypes:
               DQB DRB  B hap.freq
 geno.glm.13    21   7  7   0.0124
 geno.glm.17    21   7 44   0.0232
 geno.glm.34    31   4 44   0.0286
 geno.glm.50    31  11 35   0.0169
 geno.glm.55    31  11 51   0.0115
 geno.glm.69    32   4  7   0.0152
```

```
geno.glm.77    32   4 60   0.0318
geno.glm.78    32   4 62   0.0239
geno.glm.99    51   1 27   0.0150
geno.glm.100   51   1 35   0.0300
geno.glm.102   51   1 44   0.0176
geno.glm.138   62   2  7   0.0509
geno.glm.140   62   2 18   0.0154
geno.glm.143   62   2 44   0.0140
geno.glm.155   63   2  7   0.0136
geno.glm.162   63  13  7   0.0161
geno.glm.165   63  13 44   0.0164
geno.glm.rare   *   *  *   0.5434
haplo.base     21   3  8   0.1041
```

6.6.2 Selecting the Baseline Haplotype (NEW)

The haplotype chosen for the baseline in the model is the one with the highest frequency. Sometimes the most frequent haplotype may be an at-risk haplotype, and so the measure of its effect is desired. To specify a more appropriate haplotype as the baseline in the binomial example, choose from the list of other common haplotypes, *fit.bin$haplo.common*. To specify an alternative baseline, such as haplotype 77, use the control parameter *haplo.base* and haplotype code, as in the example below.

```
> fit.bin$haplo.common

  [1]  13  17  34  50  55  69  77  78  99 100 102 138 140 143
 [15] 155 162 165

> fit.bin$haplo.freq.init[fit.bin$haplo.common]

  [1] 0.01245945 0.02332031 0.02848720 0.01753713 0.01137492
  [6] 0.01677789 0.03060053 0.02349463 0.01504752 0.03018431
 [11] 0.01730642 0.05097906 0.01545287 0.01367136 0.01332757
 [16] 0.01655073 0.01606451

> fit.bin.base77 <- haplo.glm(y.bin ~ male + geno.glm,
+     family = binomial, data = my.data, na.action = "na.geno.keep",
+     locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
+     control = haplo.glm.control(haplo.base = 77,
+         haplo.min.count = 5))
> print(fit.bin.base77)

   Call:
haplo.glm(formula = y.bin ~ male + geno.glm,
```

```
    family = binomial, data = my.data, na.action = "na.geno.keep",
    locus.label = label, allele.lev = attributes(geno.glm)$unique.alleles,
    control = haplo.glm.control(haplo.base = 77,
        haplo.min.count = 5))

Coefficients:
                  coef       se    t.stat     pval
(Intercept)    -0.5077 2.87e-01 -1.77e+00 7.88e-02
male           -0.5578 3.26e-01 -1.71e+00 8.87e-02
geno.glm.4      1.1237 3.30e-01  3.41e+00 7.87e-04
geno.glm.13   -16.8518 2.91e-09 -5.79e+09 0.00e+00
geno.glm.17     0.3625 9.74e-02  3.72e+00 2.58e-04
geno.glm.34     1.3735 2.20e-01  6.24e+00 2.51e-09
geno.glm.50    -1.1590 2.38e-02 -4.87e+01 0.00e+00
geno.glm.55    -0.6484 2.02e-02 -3.20e+01 0.00e+00
geno.glm.69    -1.4092 2.02e-02 -6.98e+01 0.00e+00
geno.glm.78    -0.5269 5.77e-02 -9.14e+00 0.00e+00
geno.glm.99    -0.7145 4.49e-02 -1.59e+01 0.00e+00
geno.glm.100   -1.6262 3.98e-02 -4.09e+01 0.00e+00
geno.glm.102    0.1496 4.65e-02  3.22e+00 1.51e-03
geno.glm.138   -1.7230 9.60e-02 -1.80e+01 0.00e+00
geno.glm.140    0.2968 6.75e-02  4.40e+00 1.77e-05
geno.glm.143    0.4904 4.87e-02  1.01e+01 0.00e+00
geno.glm.155   -0.4640 2.51e-02 -1.85e+01 0.00e+00
geno.glm.162  -16.4486 5.44e-09 -3.02e+09 0.00e+00
geno.glm.165    0.0832 5.85e-02  1.42e+00 1.57e-01
geno.glm.rare  -0.1177 2.14e-01 -5.49e-01 5.84e-01

Haplotypes:
              DQB DRB  B hap.freq
geno.glm.4     21   3  8   0.1041
geno.glm.13    21   7  7   0.0129
geno.glm.17    21   7 44   0.0226
geno.glm.34    31   4 44   0.0286
geno.glm.50    31  11 35   0.0170
geno.glm.55    31  11 51   0.0115
geno.glm.69    32   4  7   0.0169
geno.glm.78    32   4 62   0.0236
geno.glm.99    51   1 27   0.0152
geno.glm.100   51   1 35   0.0298
geno.glm.102   51   1 44   0.0175
```

```
geno.glm.138   62   2  7   0.0515
geno.glm.140   62   2 18   0.0155
geno.glm.143   62   2 44   0.0140
geno.glm.155   63   2  7   0.0129
geno.glm.162   63  13  7   0.0159
geno.glm.165   63  13 44   0.0164
geno.glm.rare   *   *  *   0.5440
haplo.base     32   4 60   0.0305
```

Explanation of Results

The above model has the same haplotypes as *fit.bin*, except haplotype 4, the old baseline, now has an effect estimate while haplotype 77 is the new baseline. Due to randomness in the starting values of the haplotype frequency estimation, different runs of *haplo.glm* may result in a different set of haplotypes meeting the minimum counts requirement for being modeled. Therefore, once you have arrived at a suitable model, and you wish to modify it by changing baseline and/or effects, you can make results consistent by controlling the randomness using *set.seed*, as described in section 3.3. In this document, we use the same seed before making *fit.bin* and *fit.bin.base77*.

7 Extended Applications

The following functions are designed to wrap the functionality of the major functions in Haplo Stats into other useful applications.

7.1 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  -2.19387 0.02824  0.05098 0.06789 0.01587
2   51   1 35  -1.58421 0.11315  0.03018 0.03754 0.00907
3   63  13  7  -1.56008 0.11874  0.01655 0.02176      NA
```

```
4   21   7  7  -1.47495 0.14023  0.01246 0.01969      NA
5   32   4  7  -1.00091 0.31687  0.01678 0.02628 0.00794
6   32   4 62  -0.67990 0.49657  0.02349 0.01911      NA
7   51   1 27  -0.66509 0.50599  0.01505 0.01855 0.00907
8   31  11 35  -0.58380 0.55936  0.01754 0.01982 0.01587
9   31  11 51  -0.43721 0.66196  0.01137 0.01321      NA
10  51   1 44   0.00826 0.99341  0.01731 0.01595 0.00000
```

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.Freq* is the haplotype frequency for the total sample, and the remaining 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 frequencyies.

7.2 Case-Control Haplotype Analysis: *haplo.cc* (NEW)

It is possible to combine the results of *haplo.score*, *haplo.group*, and *haplo.glm* for case-control data, all performed within *haplo.cc*. The function peforms a score test and a glm on the same haplotypes. Haplotypes used in the analysis have an expected count at least as large as *haplo.min.count*, which is explained in section 6.2.
Below, we execute *haplo.cc*, view a print-out of the results, then look at the names of the objects stored within the *cc.hla* result.

```
> cc.hla <- haplo.cc(y = y.bin, geno = geno, haplo.min.count = 5,
+     locus.label = label)
> print(cc.hla, nlines = 25, digits = 2)

  ------------------------------------------------------------
                   Global Score Statistics
  ------------------------------------------------------------

  global-stat = 34, df = 18, p-val = 0.014

  ------------------------------------------------------------
               Counts for Cases and Controls
  ------------------------------------------------------------
```

```
control    case
    157      63


------------------------------------------------------------
        Haplotype Scores, p-values, Hap-Frequencies
               (hf), and Odds Ratios (95% CI)
------------------------------------------------------------

    DQB DRB  B Hap-Score   p-val pool.hf control.hf case.hf
151  62   2  7   -2.1939 0.02824  0.0491     0.0679 1.6e-02
101  51   1 35   -1.5842 0.11315  0.0302     0.0376 8.8e-03
166  63  13  7   -1.5601 0.11874  0.0142     0.0218      NA
22   21   7  7   -1.4750 0.14023  0.0124     0.0197      NA
79   32   4  7   -1.0009 0.31687  0.0251     0.0263 7.9e-03
78   32   4 62   -0.6799 0.49657  0.0188     0.0191      NA
100  51   1 27   -0.6651 0.50599  0.0144     0.0161 8.8e-03
29   31  11 35   -0.5838 0.55936  0.0176     0.0198 1.6e-02
34   31  11 51   -0.4372 0.66196  0.0111     0.0125 7.9e-03
103  51   1 44    0.0083 0.99341  0.0177     0.0178 7.7e-08
77   32   4 60    0.0318 0.97462  0.0307     0.0315 4.0e-02
148  62   2 44    0.1658 0.86830  0.0135     0.0133      NA
162  63  13 44    0.2206 0.82541  0.0161     0.0164 9.3e-03
169  63   2  7    0.2982 0.76555  0.0136     0.0103 1.6e-02
145  62   2 18    0.7885 0.43038  0.0156     0.0126 2.4e-02
17   21   7 44    0.8456 0.39776  0.0236     0.0175 4.8e-02
51   31   4 44    2.5077 0.01215  0.0284     0.0145 6.3e-02
12   21   3  8    3.7776 0.00016  0.1040     0.0697 1.9e-01
1    21   1  8        NA      NA  0.0023     0.0033      NA
2    21  10  8        NA      NA  0.0018     0.0032      NA
3    21  13  8        NA      NA  0.0027         NA      NA
4    21   2 18        NA      NA  0.0023     0.0032      NA
5    21   2  7        NA      NA  0.0023     0.0032      NA
6    21   3 18        NA      NA  0.0046     0.0064      NA
7    21   3 35        NA      NA  0.0057     0.0064      NA
    glm.eff OR.lower       OR OR.upper
151     Eff  1.5e-02 6.7e-02  3.0e-01
101     Eff  2.2e-02 7.9e-02  2.8e-01
166     Eff  2.5e-08 2.5e-08  2.5e-08
22      Eff  1.6e-08 1.6e-08  1.6e-08
79      Eff  4.1e-02 8.0e-02  1.5e-01
```

```
78      Eff  4.4e-02 2.1e-01  1.0e+00
100     Eff  1.0e-01 1.9e-01  3.6e-01
29      Eff  8.9e-02 1.5e-01  2.4e-01
34        R  1.8e-01 2.9e-01  4.7e-01
103     Eff  8.4e-02 4.4e-01  2.3e+00
77      Eff  8.5e-02 3.0e-01  1.1e+00
148     Eff  8.5e-02 5.3e-01  3.3e+00
162     Eff  6.5e-02 3.5e-01  1.9e+00
169     Eff  4.6e-02 2.3e-01  1.2e+00
145     Eff  1.2e-01 5.7e-01  2.8e+00
17      Eff  1.3e-01 5.3e-01  2.2e+00
51      Eff  3.7e-01 1.2e+00  4.1e+00
12     Base       NA 1.0e+00       NA
1         R  1.8e-01 2.9e-01  4.7e-01
2         R  1.8e-01 2.9e-01  4.7e-01
3         R  1.8e-01 2.9e-01  4.7e-01
4         R  1.8e-01 2.9e-01  4.7e-01
5         R  1.8e-01 2.9e-01  4.7e-01
6         R  1.8e-01 2.9e-01  4.7e-01
7         R  1.8e-01 2.9e-01  4.7e-01
```

```
> names(cc.hla)
```

```
[1] "cc.df"       "group.count" "score.lst"   "fit.lst"
[5] "ci.prob"
```

Explanation of Results

First, from the names function we see that *cc.hla* also contains *score.lst* and *fit.lst*, which are the *haplo.score* and *haplo.glm* objects, respectively. For the printed results of *haplo.cc*, first are the global statistics from *haplo.score*, followed by cell counts for cases and controls. The last portion of the output is a data frame containing combined results for individual haplotypes:

- **Hap-Score:** haplotype score statistic
- **p-val:** haplotype score statistic p-value
- **sim p-val:** (if simulations performed) simulated p-value for the haplotype score statistic
- **pool.hf:** haplotype frequency for the pooled sample
- **control.hf:** haplotype frequencies for the control sample only
- **case.hf:** haplotype frequencies for the case sample only

- **glm.eff:** one of three ways the haplotype appeared in the glm model: *Eff*: modeled as an effect; *Base*: part of the baseline; and *R*: a rare haplotype, included in the effect of pooled rare haplotypes
- **OR.lower:** Odds Ratio confidence interval lower limit
- **OR:** Odds Ratio for each effect in the model
- **OR.upper:** Odds Ratio confidence interval upper limit

Significance levels are indicated by the p-values for the score statistics, and the odds ratio (OR) confidence intervals for the haplotype effects. Note that the Odds Ratios are effect sizes of haplotypes, assuming haplotype effects are multiplicative. Since this last table has many columns, lines are wrapped in the output in this manual. You can align wrapped lines by the haplotype code which appears on the far left. Alternatively, instruct the print function to only print *digits* significant digits, and set the width settings for output in your session using the *options()* function.

7.3 Score Tests on Sub-Haplotypes: *haplo.score.slide* (NEW)

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 previously defined gaussian trait *resp*.

```
> 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 = 5/(2 *
+         nrow(geno.11)), locus.label = label.11)
> print(score.slide.gaus)

  start.loc score.global.p global.p.sim max.p.sim
1         1       0.215498           NA        NA
2         2       0.093664           NA        NA
3         3       0.390424           NA        NA
4         4       0.487713           NA        NA
5         5       0.137468           NA        NA
6         6       0.149241           NA        NA
7         7       0.110008           NA        NA
```

```
8          8          0.009963          NA          NA
9          9          0.029047          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 for each call. 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.

7.3.1 Plot Results from *haplo.score.slide*

The results from *haplo.score.slide* can be easily viewed in a plot shown in Figure 2, at the end of this document. 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 = 0.009963* for loci 8-10 is plotted as a horizontal line at $y = 2.002$ spanning the 8^{th}, 9^{th}, and 10^{th} x-axis tick marks.

7.4 Scanning Haplotypes Within a Fixed-Width Window: *haplo.scan* (NEW)

Another method to search for a candidate locus within a genome region is *haplo.scan*. This method searches for a region for which the haplotypes have the strongest association with a binary trait by sliding a window of fixed width over each marker locus, and then scans over all haplotype lengths within each window. This latter step, scanning over all possible haplotype lengths within a window, distinguishes *haplo.scan* from *haplo.score.slide* (which considers only the maximum haplotype length within a window). To acount for unknown linkage phase, the function *haplo.em* is called prior to scanning, to create a list of haplotype pairs and posterior probabilities. To illustrate the scanning of window, consider a 10-locus dataset. When placing a window of width 3 over locus 5, the possible haplotype lengths that contain locus 5 are three (loci 3-4-5, 4-5-6, and 5-6-7), two (loci 4-5 and 5-6) and one (locus 5). For each of these loci subsets a score statistic is computed, which is based on the difference between the mean vector of haplotype counts for cases and that for controls. The maximum of these score statistics, over all possible haplotype lengths within a window, is the locus-specific test statistic, or the locus scan statistic. The global test statistic is the maximum over all computed score statistics. To compute p-values, the case/control status is randomly permuted. Below we run *haplo.scan* on the 11-locus HLA dataset with a binary response and a window width of 3, but first we use the results of *summaryGeno* to choose subjects with less than $50,000$ haplotype pairs to speed calculations with all 11 polymorphic loci with many missing alleles.

```
> geno.11 <- hla.demo[, -c(1:4)]
> y.bin <- 1 * (hla.demo$resp.cat == "low")
> hla.summary <- summaryGeno(geno.11, miss.val = c(0,
+     NA))
> many.haps <- (1:length(y.bin))[hla.summary[, 4] >
+     50000]
> geno.scan <- geno.11[-many.haps, ]
> y.scan <- y.bin[-many.haps]
> scan.hla <- haplo.scan(y.scan, geno.scan, width = 3,
+     sim.control = score.sim.control(min.sim = 100,
+         max.sim = 100), em.control = haplo.em.control())
> print(scan.hla)

   Call:
 haplo.scan(y = y.scan, geno = geno.scan,
     width = 3, em.control = haplo.em.control(),
     sim.control = score.sim.control(min.sim = 100,
         max.sim = 100))

 ============================================================
          Locus Scan-statistic Simulated P-values
 ============================================================

           loc-1 loc-2 loc-3 loc-4 loc-5 loc-6 loc-7 loc-8
 sim.p-val  0.03  0.02  0.03  0.01  0.01  0.03  0.01  0.03
           loc-9 loc-10 loc-11
 sim.p-val  0.01   0.01   0.01


      Loci with max scan statistic:   2
  Max-Stat Simulated Global p-value:   0.02
             Number of Simulations:   100
```

Explanation of Results

In the output we report the simulated p-values for each locus test statistic. Additionally, we report the loci (or locus) which provided the maximum observed test statistic, and the *Max-Stat Simulated Global p-value* is the simulated p-value for that maximum statistic. We print the number of simulations, because they are performed until p-value precision criteria are met, as described in section 5.7. We would typically allow simulations to run under default parameters rather than limiting to 100 by the control parameters.

8 License and Warranty

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

9 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] Cheng R, Ma JZ, Wright FA, Lin S, Gau X, Wang D, Elston RC, Li MD. (2003) Nonparametric disequilibrium mapping of functional sites using haplotypes of multiple tightly linked single-nucleotide polymorphism markers. *Genetics* 164:1175-1187.

[3] Cheng R, Ma JZ, Elston RC, Li MD. (2005) Fine Mapping Functional Sites or Regions from Case-Control Data Using Haplotypes of Multiple Linked SNPs. *Annals of Human Genetics* 69: 102-112.

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

[5] Epstein MP, Satten GA (2003) Inference on haplotype effects in case-control studies using unphased genotype data. *Am J Hum Genet* 73: 1316-1329.

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

[7] 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

[8] 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

[9] 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

```
> plot(score.gaus)
> cat("These next two steps substitute for doing: \n",
+     "\t > locator.haplo(score.gaus)\n")

 These next two steps substitute for doing:
         > locator.haplo(score.gaus)

> pts.haplo <- list(x.coord = c(0.05098, 0.03018,
+     0.1), y.coord = c(2.1582, 0.45725, -2.1566),
+     hap.txt = c("62:2:7", "51:1:35", "21:3:8"))
> text(x = pts.haplo$x.coord, y = pts.haplo$y.coord,
+     labels = pts.haplo$hap.txt)

 NULL
```

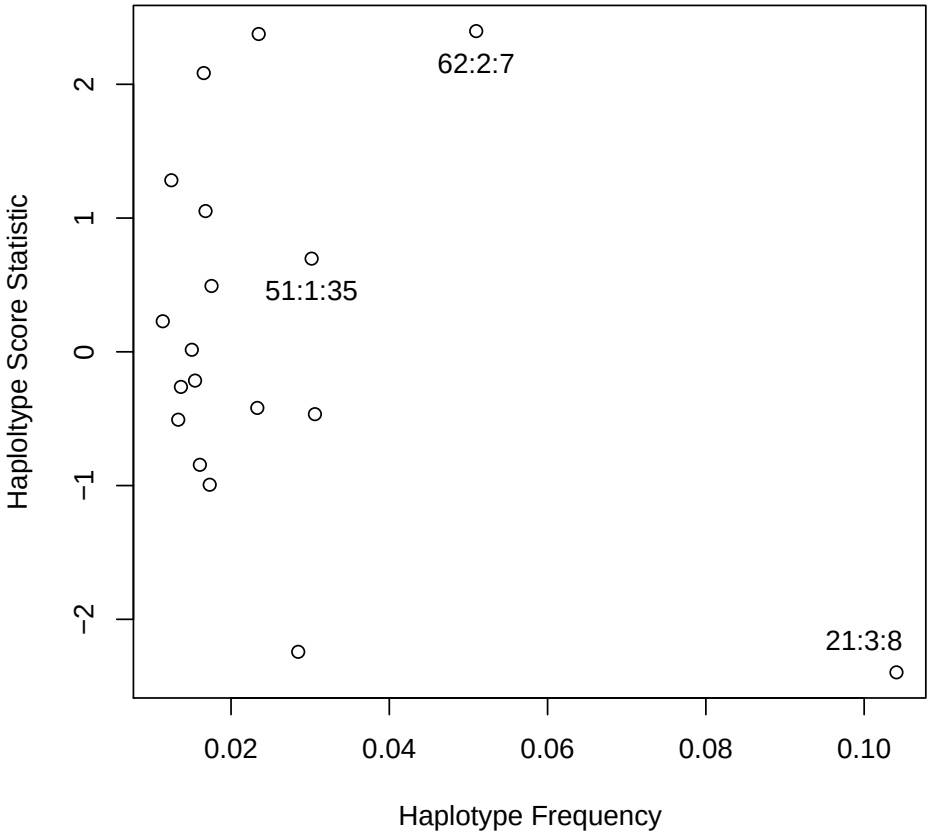

Figure 1: Haplotype Statistics: Score vs. Frequency, Quantitative Response

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

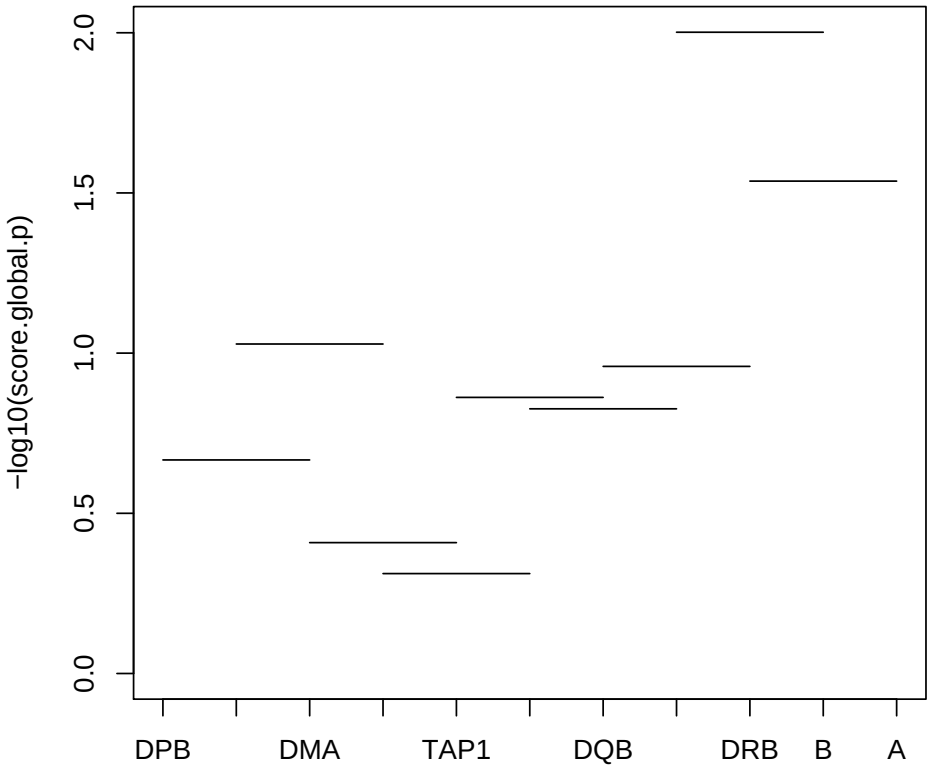

Figure 2: Global p-values for sub-haplotypes; Gaussian Response