haplo.score

Score Tests for Association of Traits with Haplotypes when

Linkage Phase is Ambiguous

Daniel J. Schaid and David E. Tines

Mayo Clinic
Rochester, MN
E-mail contact: schaid@mayo.edu

The changes for the R version of the library (and this document) were made by
Gregory R. Warnes <gregory_r_warnes@groton.pfizer.com>

I. Brief Description

This suite of R routines, referred to as "haplo.score", can be used to compute score statistics to test associations between haplotypes and a wide variety of traits, including binary, ordinal, quantitative, and Poisson. These methods assume that all subjects are unrelated and that haplotypes are ambiguous (due to unknown linkage phase of the genetic markers). The methods provide several different global and haplotype-specific tests for association, as well as provide adjustment for non-genetic covariates and computation of simulation p-values (which may be needed for sparse data). Details on the background and theory of the score statistics can be found in the following reference:

> Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. Score tests for association of traits with haplotypes when linkage phase is ambiguous. American J Human Genetics, February, 2002.

II. Important Details

The current R version of `haplo.score` (1.0) was translated from version 1.0 for S-PLUS, which written for S-PLUS version 6.0 on a Unix operating system. The `haplo.score` library may be installed using the standard R mechanism. See the R documentation and or the INSTALL online help page, which can be accessed by typing

```
?INSTALL
```

at the R command prompt.

III. Getting Started by Example

After installing the haplo.score package, the routines and an example data set are available by starting R and loading the `haplo.score` library. The easiest way to get started is by following an example. The experienced R user may want to skip the example and simply view the details in the help file for `haplo.score`. In the following R example session, indented text following the prompt ">" is what the user would type into an R session, as well as results printed by R.

Example R Session

To the illustrate use of the `haplo.score` analyses, first load the `haplo.score` library. To do this type

```
> library(haplo.score)
```

and then load the example `hla.demo` data set using

```
> data(hla.demo)
```

Now look at the names of variables in `hla.demo`.

```
> names(hla.demo)
 [1] "resp"     "resp.cat" "male"     "age"      "DQB.a1"   "DQB.a2"
 [7] "DRB.a1"   "DRB.a2"   "B.a1"     "B.a2"
```

The variables 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`	sex code with 1="male" , 0="female"
`age`	age (months) at immunization

The remaining variables are genotypes for 3 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 to attach `hla.demo`, so the variables can be accessed by simply typing their names.

```
> attach(hla.demo)
```

For convenience, create a separate data frame for the 3 loci, and call this `geno` (mainly to make it easier to refer to the loci, distinct from the other variables in `hla.demo`).

```
> geno <- hla.demo[,-c(1:4)]
```

Create some labels for the loci.

```
> locus.label <-c("DQB","DRB","B")
```

Quantitative Trait Analysis:

First, analyze the quantitative trait called `resp`. A quantitative trait is identified by the option `trait.type="gaussian"` (reminding us that a gaussian distribution is assumed for the distribution of the error terms).

The other arguments in the function are defined in the help file, viewed by typing `help(haplo.score).`

```
> hap.gaus <- haplo.score(resp, geno, trait.type="gaussian",
+                         offset = NA, x.adj = NA, skip.haplo=.005,
+                         locus.label=locus.label, miss.val=0, n.sim=0)
```

To view results, we can either type the name of the object, `hap.gaus`, or `print(hap.gaus).`

```
> print(hap.gaus)
-------------------------------------------------------------------------------
                          Global Score Statistics
-------------------------------------------------------------------------------

global-stat = 46.60471, df = 40, p-val = 0.21918

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

      DQB DRB  B Hap-Freq Hap-Score   p-val
 [1,] 21  3   8  0.10501  -2.45007  0.01428
 [2,] 21  7   13 0.01082  -2.31499  0.02061
 [3,] 31  4   44 0.02871  -2.26739  0.02337
 [4,] 63  13  60 0.00558  -1.65137  0.09866
 [5,] 31  11  27 0.00609  -1.05623  0.29086
 [6,] 62  2   35 0.01046  -0.98429  0.32497
 [7,] 51  1   44 0.01744  -0.9311   0.3518
 [8,] 63  13  44 0.01555  -0.71345  0.47557
 [9,] 33  7   57 0.00688  -0.59746  0.5502
[10,] 63  2   7  0.01376  -0.55206  0.58091
[11,] 31  11  44 0.01002  -0.53055  0.59573
[12,] 32  4   60 0.0309   -0.4914   0.62314
[13,] 21  7   44 0.02354  -0.44142  0.65891
[14,] 33  9   60 0.00688  -0.43678  0.66227
[15,] 21  3   35 0.00575  -0.41964  0.67475
[16,] 62  2   44 0.0138   -0.28164  0.77822
[17,] 62  2   60 0.00515  -0.24514  0.80635
[18,] 62  2   18 0.01556  -0.23514  0.8141
[19,] 21  7   62 0.00768  -0.05942  0.95262
[20,] 51  1   27 0.01415  0.05732   0.95429
[21,] 32  8   7  0.00688  0.09216   0.92657
[22,] 31  4   13 0.00524  0.13048   0.89619
[23,] 31  11  37 0.00688  0.15753   0.87483
[24,] 31  11  51 0.01095  0.16185   0.87142
[25,] 31  4   60 0.00692  0.1961    0.84453
```

```
[26,] 31  11  38 0.00688  0.33018   0.74126
[27,] 64  13  7  0.00961  0.46477   0.6421
[28,] 31  11  35 0.01728  0.4883    0.62534
[29,] 51  1   51 0.0074   0.5015    0.61602
[30,] 63  13  38 0.00688  0.6685    0.50381
[31,] 63  13  62 0.0084   0.76891   0.44195
[32,] 51  1   35 0.02967  0.79307   0.42773
[33,] 64  13  63 0.00688  0.88207   0.37774
[34,] 31  11  62 0.00608  0.9689    0.3326
[35,] 32  4   7  0.01718  0.99641   0.31905
[36,] 64  13  35 0.00672  1.25833   0.20827
[37,] 21  7   7  0.01257  1.26578   0.20559
[38,] 63  13  7  0.01605  2.19393   0.02824
[39,] 32  4   62 0.02371  2.35151   0.0187
[40,] 62  2   7  0.05073  2.39238   0.01674
```

Explanation of table results:

The first column (e.g., [1,]) gives row numbers.
The next 3 columns are the alleles making up 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.
`p-val` is the asymptotic chi-square p-value.

In our example, the option `n.sim=0` in the function `haplo.score` implied no simulation p-values (simulation p-values are computed when `n.sim` >0.)

Note that this table is sorted according to `Hap-Score`.

Plots and Haplotype Labels

Another convenient way to view results is a plot of the haplotype frequencies (Hap-Freq) versus the haplotype score statistics (Hap-Score).

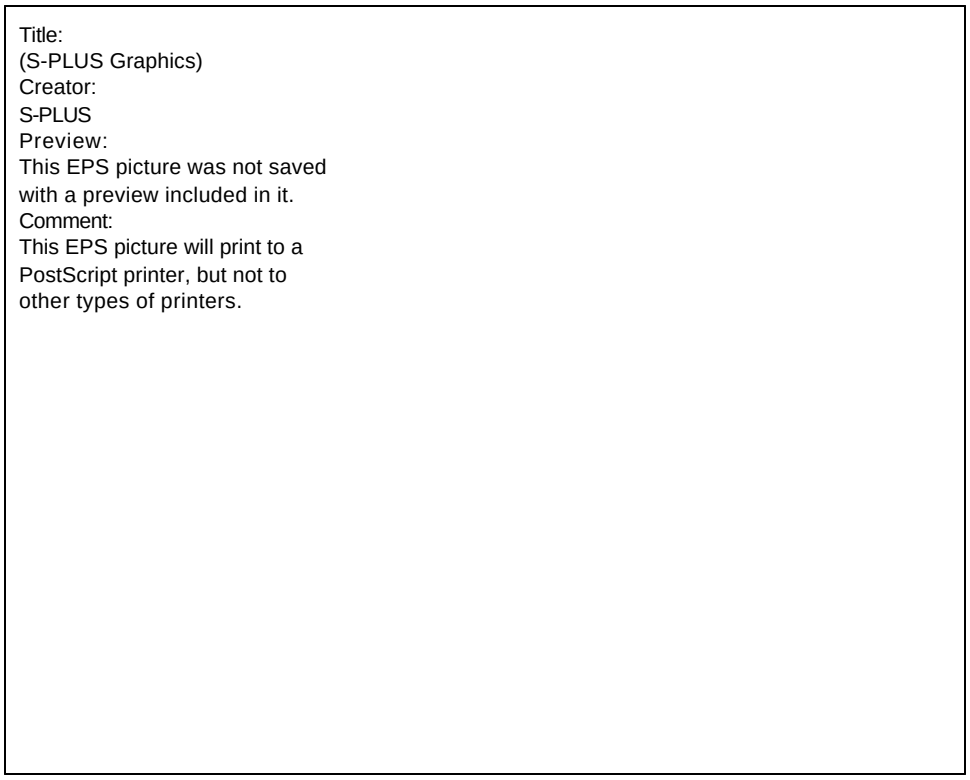
```
> plot(hap.gaus)
> title("Figure 1. Haplotype Score Statistics\nQuantitative Response")
```

Some points on the plot may be of interest, perhaps due to their score statistic, or their haplotype frequency. To put haplotype labels on individual points in the plot, use the function `locator.haplo`. To use this feature, first type the following command:

```
> locator.haplo(hap.gaus)
```

where `hap.gaus` is the object where the results are stored.

Now, with your left mouse button, click on all the points of interest. After all your chosen points are clicked, click on the middle mouse button. Viola! All the points are labeled with their haplotype labels, as illustrated in Figure 1.



Note that some of the haplotype labels may overlap, so that it may be necessary to clean up the coordinates. To do this, you can save the x-y coordinates and haplotype text,

```
> loc.gaus <- locator.haplo(hap.gaus)
```

and then fix some of the x-y coordinates in the `loc.gaus` list.

Skipping Rare Haplotypes

For the above analyses, we used the option `skip.haplo=.005`, to pool all haplotypes with frequencies <.005 into a common group. Let's try a different cut-off, such as `skip.haplo=.01`.

```
> hap.gaus2 <- haplo.score(resp, geno, trait.type="gaussian",
+                          offset = NA, x.adj = NA, skip.haplo=.01,
+                          locus.label=locus.label, miss.val=0, n.sim=0)


> print(hap.gaus2)

-------------------------------------------------------------------------------
                          Global Score Statistics
-------------------------------------------------------------------------------
global-stat = 35.01088, df = 21, p-val = 0.02816
```

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

      DQB DRB  B Hap-Freq Hap-Score   p-val
 [1,] 21  3   8  0.10501  -2.45007  0.01428
 [2,] 21  7   13 0.01082  -2.31499  0.02061
 [3,] 31  4   44 0.02871  -2.26739  0.02337
 [4,] 62  2   35 0.01046  -0.98429  0.32497
 [5,] 51  1   44 0.01744  -0.9311   0.3518
 [6,] 63  13  44 0.01555  -0.71345  0.47557
 [7,] 63  2   7  0.01376  -0.55206  0.58091
 [8,] 31  11  44 0.01002  -0.53055  0.59573
 [9,] 32  4   60 0.0309   -0.4914   0.62314
[10,] 21  7   44 0.02354  -0.44142  0.65891
[11,] 62  2   44 0.0138   -0.28164  0.77822
[12,] 62  2   18 0.01556  -0.23514  0.8141
[13,] 51  1   27 0.01415  0.05732   0.95429
[14,] 31  11  51 0.01095  0.16185   0.87142
[15,] 31  11  35 0.01728  0.4883    0.62534
[16,] 51  1   35 0.02967  0.79307   0.42773
[17,] 32  4   7  0.01718  0.99641   0.31905
[18,] 21  7   7  0.01257  1.26578   0.20559
[19,] 63  13  7  0.01605  2.19393   0.02824
[20,] 32  4   62 0.02371  2.35151   0.0187
[21,] 62  2   7  0.05073  2.39238   0.01674
```

Note that by using a different value for `skip.haplo`, the global statistic and its p-value change (due to decreased `df`), but the haplotype-specific scores do not change.

Haplotype scores, adjusted for sex and age

First set up a matrix, with the first column for sex (male=1/female=0), and the second column for age (in months).

```
> x.ma <- cbind(male, age)
```

Now use this matrix as an argument to `haplo.score`.

```
> hap.gaus.adj <- haplo.score(resp, geno, trait.type="gaussian",
+                         offset = NA, x.adj = x.ma, skip.haplo=.005,
+                         locus.label=locus.label, miss.val=0, n.sim=0)
> print(hap.gaus.adj)


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

global-stat = 46.6244, df = 40, p-val = 0.2186
```

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

      DQB DRB  B Hap-Freq Hap-Score   p-val
 [1,] 21  3   8  0.10501  -2.45887  0.01394
 [2,] 21  7   13 0.01082  -2.30155  0.02136
 [3,] 31  4   44 0.02871  -2.26812  0.02332
  .
  .
  .
[38,] 63  13  7  0.01605  2.18738   0.02871
[39,] 32  4   62 0.02371  2.33344   0.01963
[40,] 62  2   7  0.05073  2.37943   0.01734
```

When adjusting for covariates, all score statistics can change, although not by much in this example.

Ordinal Traits

We will create an ordinal trait, converting `resp.cat` (a factor with levels "low", "normal", "high") to numeric values, `y.ord` (with levels 1, 2, 3).

```
> y.ord <- as.numeric(resp.cat)
```

Now use the option `trait.type = "ordinal"`

```
> hap.ord <- haplo.score(y.ord, geno, trait.type="ordinal",
+                        offset = NA, x.adj = NA, skip.haplo=.005,
+                        locus.label=locus.label, miss.val=0, n.sim=0)
> print(hap.ord)
-------------------------------------------------------------------------------
                           Global Score Statistics
-------------------------------------------------------------------------------

global-stat = 65.12193, df = 40, p-val = 0.00727

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

      DQB DRB  B Hap-Freq Hap-Score   p-val
 [1,] 21  7   13 0.01082  -3.69342  0.00022
 [2,] 21  3   8  0.10501  -2.83638  0.00456
 [3,] 31  4   44 0.02871  -2.63765  0.00835
  .
  .
  .
[38,] 32  4   62 0.02371  1.89113   0.05861
[39,] 63  13  7  0.01605  1.89743   0.05777
[40,] 62  2   7  0.05073  2.42248   0.01542
```

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. (see Harrell, FE. Regression Modeling Strategies, Springer-Verlag, NY, 2001). 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. To check whether your local system has these libraries, type

> library()

and you should then be able to examine the list of libraries available to you.

Binary Traits

Because "low" responders are of primary interest, let's create a binary trait that has values of 1 when response is "low", and 0 otherwise.

```
> y.bin <-ifelse(y.ord==1,1,0)
```

Now use the option `trait.type = "binomial"` in the `haplo.score` routine

```
> hap.bin <- haplo.score(y.bin, geno, trait.type="binomial",
+                        offset = NA, x.adj = NA, skip.haplo=.005,
+                        locus.label=locus.label, miss.val=0, n.sim=0)


> print(hap.bin)
-------------------------------------------------------------------------------
                          Global Score Statistics
-------------------------------------------------------------------------------

global-stat = 65.87372, df = 40, p-val = 0.00614

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

      DQB DRB  B Hap-Freq Hap-Score   p-val
 [1,] 62  2   7  0.05073  -2.14567  0.0319
 [2,] 51  1   35 0.02967  -1.70405  0.08837
 [3,] 63  13  7  0.01605  -1.69539  0.09
  .
  .
  .
[38,] 31  4   44 0.02871  2.53351   0.01129
[39,] 21  7   13 0.01082  3.69342   0.00022
[40,] 21  3   8  0.10501  3.82268   0.00013
```

Simulation p-values

Simulation p-values are computed when `n.sim` > 0, and the value of `n.sim` directs the number of simulations to perform. The following example illustrates computation of 1,000 simulations.

```
> hap.bin.sim <- haplo.score(y.bin, geno, trait.type="binomial",
+                          offset = NA, x.adj = NA, skip.haplo=.005,
+                          locus.label=locus.label, miss.val=0, n.sim=1000)

> print(hap.bin.sim)

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

global-stat = 65.87372, df = 40, p-val = 0.00614, sim p-val =  0.004

max-stat sim. p-val =  0.001

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

      DQB DRB  B Hap-Freq Hap-Score   p-val sim p-val
 [1,] 62  2   7  0.05073  -2.14567  0.0319  0.031
 [2,] 51  1   35 0.02967  -1.70405  0.08837 0.112
 [3,] 63  13  7  0.01605  -1.69539  0.09    0.188
  .
  .
  .
[38,] 31  4   44 0.02871  2.53351   0.01129 0.006
[39,] 21  7   13 0.01082  3.69342   0.00022 0.003
[40,] 21  3   8  0.10501  3.82268   0.00013 0
```

When simulations are requested, simulated p-values are computed for the `global-stat` (which has an asymptotic chi-square distribution), as well as `max-stat` (the maximum absolute value of the haplotype-specific scores). Furthermore, simulated p-values are computed for each of the haplotype-specific scores.

IV. Future Directions and Feedback

We would like your feedback on the use of this software. If you find strange results or behavior of the software, or you would like to make suggestions on features that you would find helpful, please let us know.

Some of our future plans include:

- use of the standard model formula syntax ,
- extensions for analysis of censored survival data, longitudinal data, and matched case-control designs.

You can send suggestions by email to schaid@mayo.edu.

Acknowledgements

This research was supported by United States Public Health Services, National Institutes of Health; Contract grant numbers R01 DE13276, 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.