Estimation of haplotypes

Cavan Reilly

October 9, 2019

Table of contents

Overview

If there is substantial LD over a region containing a disease mutation, collections of SNPs within this region should all be associated with the disease phenotype.

If we test each SNP we introduce test multiplicity problems that could be avoided if we looked at an association between a multilocus haplotype and the disease trait.

However multilocus haplotypes can't be determined directly from the genotypic data in humans.

Thus we need methods for estimating haplotypes and testing for an association between these haplotypes and a trait of interest.

Overview

We need to take care to fully incorporate all sources of variability when looking at associations between traits and haplotypes.

In particular, we can't simply estimate the haplotype for each individual, then pretend that we know these haplotypes and test for an association between the occurrence of the haplotype and the disease trait.

This will overstate our certainty about the haplotypes and will lead to false positives.

Overview

The text discusses 2 approaches, an EM algorithm based approach and a Bayesian approach.

These 2 approaches are not very different: in particular they use the same model, but they differ with regard to the method of inference.

The method that uses the EM algorithm uses maximum likelihood to estimate the parameters in the model, whereas the Bayesian approach uses the posterior mode to estimate the haplotype for each subject.

From a practical perspective this distinction is like using the median or the mean to estimate the center of a distribution: we have the model that our measurements are actually measurements of the same quantity but that quantity is subject to measurement error.

Frequentist vs. Bayesian statistics

Most people learn *frequentist* statistics when they learn statistics.

Let's say a *parameter* is a numerical quantity that governs the probability distribution of our measurements (we could be more general, but this is fine).

We use Roman letters to indicate data and we use Greek letters to indicate parameters.

For example, when we say $y \sim \mathrm{N}(\mu, 1)$ where μ is a parameter.

We don't observe μ, only data centered at that value.

Frequentist vs. Bayesian statistics

In frequentist statistics, we assume that parameters are fixed unknown constants.

So a confidence interval is a random interval (since it is based on the data and the data is modeled as random) that will contain the unknown parameter with some probability.

Any given confidence interval either contains the parameter or it doesn't, but we don't know if that is true or not for any particular confidence interval.

This makes sense if you are a governmental regulatory agency: you will mistakenly approve a drug once in a while but you have no idea when you will make those mistakes.

So we control the overall probability of making such mistakes.

Frequentist vs. Bayesian statistics

In Bayesian statistics, parameters are treated as random variables: what is crucial is what information one conditions upon when making statements.

We think of the data as being generated conditional upon certain values of the parameters: so we don't say simply $y \sim \mathrm{N}(\mu, 1)$ but rather $y|\mu \sim \mathrm{N}(\mu, 1)$.

So if we think we observe $y|\mu$ then the question is how to say something about μ once we observe data, y, i.e. make statements about $p(\mu|y)$.

Frequentist vs. Bayesian statistics

Bayes theorem is a simple result that allows one to say something about $p(\mu|y)$ in terms of $p(y|\mu)$ and states that

$$p(\mu|y) \propto p(y|\mu)p(\mu),$$

where the proportionality constant doesn't depend on μ.

The factor $p(y|\mu)$ is the likelihood and $p(\mu)$ is called the *prior*.

Most researchers who use Bayesian methods try to do so in a way that minimizes the impact of the prior as it reflects what we know about the parameter prior to observing the data (and we usually don't know much).

Frequentist vs. Bayesian statistics

This means that what we know about the parameter after observing the data, $p(\mu|y)$, called the *posterior distribution*, is mainly driven by the likelihood.

Hence if a Bayesian approach and a frequentist approach use the same likelihood (i.e. the same probability model), they should pretty much give the same answer-this is in fact what we observe in practice.

The main exception to this rule are situations in which one has many parameters relative to the number of observations.

In this case the primary method of frequentist inference, namely maximum likelihood, is not justified and the approximations that these methods use are probably not of very high quality.

Using the EM algorithm to estimate haplotypes

The expectation and maximization (EM) algorithm is a general statistical algorithm for computing maximum likelihood estimates or posterior modes.

It is useful when you can think of some of the data as "missing data" and the problem is such that if you had the missing data you could estimate the remaining parameters in a straightforward fashion.

We will call the "remaining parameters" the *unknown parameters* below to distinguish them from the "missing data" parameters.

We can treat the haplotype of each subject as missing data: if we had this piece of information estimating the haplotype frequencies would be simple (these frequencies are part of the model for our data).

Using the EM algorithm to estimate haplotypes

If we assume that we have p markers and all markers only take 2 values then the model for haplotype analysis is that each subject has 2 copies of one of 2^p possible haplotypes.

If $p = 2$ then there are 4 possible haplotypes and each subject has 2 copies from this set.

The model for haplotypes then is quite simple: each observation contributes 2 pieces of information where each piece of information is from a finite set.

The multinomial model is used as a model for the occurrence of each of the haplotypes, hence if there are 4 possible haplotypes there are 3 independent parameters and these parameters are the probability of observing each of the haplotypes.

Note that these models assume Hardy Weinberg equilibrium in that genotype probabilities depend solely on allele haplotype frequencies.

Using the EM algorithm to estimate haplotypes

The EM algorithm consists of applying 2 steps at each iteration of the algorithm:

- in the *E step* we compute the expected value of the logarithm of the likelihood for the missing and observed data conditional on a current value for the unknown parameters and the observed data,
- in the *M step* we maximize the result from the E-step with respect to the unknown parameters to get a new value for the unknown parameters.

Using the EM algorithm to estimate haplotypes

Note that in the E-step we compute the expected value of a random quantity and this involves summing over all possible values of the haplotype.

If there are many markers then this can be slow as the number of terms in the summation will increases exponentially with the number of markers.

We can also get an estimate of the haplotype for each individual, but these MLEs are difficult to interpret as the number of such parameters increases with the sample size.

Using the EM algorithm to estimate haplotypes in R

We will examine estimating haplotypes using the actinin3 gene within self declared Caucasians and African Americans.

Hence after loading the appropriate package and setting up the data we apply the haplotype estimation function to the subsets of data.

We also supply a value to this function that provides a lower bound for the frequency of a haplotype.

```
> attach(fms)
> install.packages("haplo.stats")
> library(haplo.stats)
> Geno <- cbind(substr(actn3_r577x,1,1), substr(actn3_r577x,2,2),
+    substr(actn3_rs540874,1,1), substr(actn3_rs540874,2,2),
+    substr(actn3_rs1815739,1,1), substr(actn3_rs1815739,2,2),
+    substr(actn3_1671064,1,1), substr(actn3_1671064,2,2))
```

Using the EM algorithm to estimate haplotypes in R

```
> SNPnames <- c("actn3_r577x", "actn3_rs540874",
+   "actn3_rs1815739", "actn3_1671064")
> Geno.C <- Geno[Race=="Caucasian" & !is.na(Race),]
> HaploEM <- haplo.em(Geno.C, locus.label=SNPnames,
+   control=haplo.em.control(min.posterior=1e-4))
```

We now examine the output from this function

Using the EM algorithm to estimate haplotypes in R

```
> HaploEM
=============================================================
                        Haplotypes
=============================================================
    actn3_r577x actn3_rs540874 actn3_rs1815739 actn3_1671064 hap.freq
1             C              A               C             G  0.00261
2             C              A               T             A  0.00934
3             C              A               T             G  0.01354
4             C              G               C             A  0.47294
5             C              G               C             G  0.01059
6             T              A               C             A  0.00065
7             T              A               T             G  0.39891
8             T              G               C             A  0.08557
9             T              G               T             A  0.00065
10            T              G               T             G  0.00520
=============================================================
                          Details
=============================================================
lnlike = -1285.406
lr stat for no LD = 2780.769 , df = 5 , p-val = 0
```

Using the EM algorithm to estimate haplotypes in R

We now do the same calculations for the African Americans.

```
> Geno.AA <- Geno[Race=="African Am" & !is.na(Race),]
> HaploEM2 <- haplo.em(Geno.AA, locus.label=SNPnames,
+    control=haplo.em.control(min.posterior=1e-4))
```

Using the EM algorithm to estimate haplotypes in R

```
> HaploEM2
=============================================================================
                                  Haplotypes
=============================================================================
  actn3_r577x actn3_rs540874 actn3_rs1815739 actn3_1671064  hap.freq
1           C              A               C              A  0.01157
2           C              A               C              G  0.08130
3           C              A               T              G  0.03764
4           C              G               C              A  0.57762
5           C              G               C              G  0.01139
6           T              A               C              A  0.00015
7           T              A               T              G  0.17166
8           T              G               C              A  0.10833
9           T              G               C              G  0.00033
=============================================================================
                                   Details
=============================================================================
lnlike = -84.97891
lr stat for no LD = 119.7087 , df = 4 , p-val = 0
```

Computing individual level haplotypes

While we can use maximum likelihood to consistently estimate the probability of each haplotype, we can use the connection between MLEs and Bayesian methods to determine the posterior probability that each subject has each possible haplotype.

The EM results above can be interpreted as the posterior mode with a prior that specifies all haplotypes are equally likely for all subjects and all possible values for the haplotype frequencies are also equally likely.

Computing individual level haplotypes

For example if there are 2 biallelic markers and someone has haplotype (AB, ab) or (Ab, aB) then if we have haplotype probabilities θ_1, θ_2, θ_3, and θ_4 for the haplotypes AB, Ab, aB, and ab then under Hardy Weinberg equilibrium we get:

posterior probability of (AB, ab) is $p_1 \propto \theta_1\theta_4$

posterior probability of (Ab, aB) is $p_2 \propto \theta_2\theta_3$.

Computing individual level haplotypes in R

The HaploEm objects we created before store the information necessary to compute the posterior probabilities for each subject.

The `nreps` feature keeps information on how many possible haplotypes are possible for each subject, while the `hap1code` and `hap2code` features tell us which of the haplotypes are possible where the numbers refer to the rows of the HaploEm output we examined previously

```
> HaploEM$nreps[1:5]
indx.subj
1 2 3 4 5
1 2 2 2 1
```

Computing individual level haplotypes in R

```
> HaploEM$indx.subj[1:8]
[1] 1 2 2 3 3 4 4 5
> HaploEM$hap1code[1:8]
[1] 4 3 4 3 4 3 4 4
> HaploEM$hap2code[1:8]
[1] 4 8 7 8 7 8 7 4
```

So subjects 1 and 5 have only 1 pair of possible haplotypes and it is number 4 in the previous list while subjects 2, 3, and 4 have either (3, 8) or (4, 7).

Computing individual level haplotypes in R

We can also take a look at the posterior probabilities of these haplotypes

```
> HaploEM$post[1:8]
[1] 1.000000000 0.006102808 0.993897192 0.006102808 0.993897192 0.006102808
[7] 0.993897192 1.000000000
```

So the (4, 7) pair seems more likely. As a check we can do these calculations directly.

Computing individual level haplotypes in R

```
> HapProb <- HaploEM$hap.prob
> HapProb
[1] 0.0026138447 0.0093400121 0.0135382726 0.4729357032 0.0105890282
[6] 0.0006518550 0.3989126970 0.0855667219 0.0006548104 0.0051970549
> q1 <- prod(HapProb[c(3,8)])
> q2 <- prod(HapProb[c(4,7)])
> q1 / (q1+q2)
[1] 0.006102807
> q2 / (q1+q2)
[1] 0.9938972
```

Testing for differences in haplotype frequency

Often interest lies testing for differences between haplotype frequencies in 2 populations.

For example we saw that the haplotype CGCA is observed with a probability of 0.47 in Caucasians but is observed with probability 0.58 in African Americans: is this difference too large to just be chance variation?

Using the general theory of maximum likelihood estimation we can obtain the standard errors of the haplotype frequencies using the observed Fisher information matrix.

Testing for differences in haplotype frequency

We can then use these estimated standard errors to compute a 95% confidence interval for the difference in the frequencies.

```
FreqDiff <- HaploEM2$hap.prob[4] -
+   HaploEM$hap.prob[4]
s1 <- HapFreqSE(HaploEM)[4]
s2 <- HapFreqSE(HaploEM2)[4]
SE <- sqrt(s1^2 + s2^2)
CI <- c(FreqDiff - 1.96*SE, FreqDiff + 1.96*SE)
CI
[1] -0.00339528 0.21277255
```

Note that since this interval contains 0 we can't exclude a value of 0 for this difference, hence there is insufficient evidence to support a claim for a difference in these probabilities.

Bayesian methods for haplotype estimation

In practice, Bayesian methods were not very useful until the early 1990s when Markov chain Monte Carlo (MCMC) methods were introduced.

They were not useful because the computations involved were not really feasible: we need to be able to do high dimensional numerical integration to use Bayesian methods.

A number of MCMC algorithms are available for carrying out these numerical integration.

The Gibbs sampler is a popular MCMC algorithm and is widely used in phylogenetic analysis, sequence motif discovery and haplotype estimation.

Bayesian methods for haplotype estimation

To see how this algorithm works, let's suppose there are p parameters and denote them $\theta_1, \theta_2, \ldots, \theta_p$.

For concreteness suppose these parameters indicate which of a set of 4 haplotypes a collection of individuals have at 2 loci.

To use *Monte Carlo estimation*, we draw samples from the joint posterior distribution of all samples

$$p(\theta_1, \theta_2, \ldots, \theta_p | y).$$

If we had such samples, say 1000 of them, then by looking at the relative frequency that a given subject had each of the haplotypes we could estimate the probability of having each haplotype for this subject.

Bayesian methods for haplotype estimation

If we then estimate an individual's haplotype with the most likely haplotype, that would be an example of using Monte Carlo estimation to find the posterior mode which we then use to estimate a haplotype.

In MCMC we generate a Markov chain whose limiting distribution is the joint posterior distribution that we want samples from.

Bayesian methods for haplotype estimation

The Gibbs sampler generates this Markov chain by starting the algorithm at the value $\theta_1, \theta_2, \ldots, \theta_p$ and successively sampling from the following probability distributions:

$$p(\theta_1|\theta_2, \ldots, \theta_p, y) \text{ to get } \theta_1^*$$

$$p(\theta_2|\theta_1^*, \theta_3, \ldots, \theta_p, y) \text{ to get } \theta_2^*$$

$$\ldots$$

$$p(\theta_p|\theta_1^*, \theta_2^* \ldots, \theta_{p-1}^*, y) \text{ to get } \theta_p^*.$$

This process will give rise to a new sample $\theta_1^*, \theta_2^*, \ldots, \theta_p^*$ and so one can draw a new sample based on this sample.

Bayesian methods for haplotype estimation

It is a *Markov chain* because $\theta_1^*, \theta_2^*, \ldots, \theta_p^*$ will depend on $\theta_1, \theta_2, \ldots, \theta_p$.

By sampling likely haplotypes for all subjects the algorithm doesn't need to consider every possible haplotype unlike the EM algorithm (which must sum over every possible haplotype during the E-step).

This property of the Gibbs sampler makes it better suited to deal with situations where there are many possible haplotypes, i.e. when there are many markers and/or these markers have many alleles.

Bayesian methods for haplotype estimation

While the EM algorithm will converge to a maximum, it may be only a local maximum.

While the Gibbs sampler may also get trapped in a local mode, it does have a chance of escaping such a mode and finding the true regions of parameter space with high posterior probability.

The program PHASE and its extensions can be used to run the Gibbs sampler to sample haplotypes.

Testing for haplotype trait associations

While estimating haplotype frequencies and testing for differences in these frequencies between populations is of interest, we usually want to test for an association between a haplotype and a trait.

As previously noted, we generally can't simply treat estimated haplotypes as known and then test for an association.

We will discuss 3 approaches to this problem:

- haplotype trend regression
- multiple imputation
- a model based approach that estimates haplotypes using the trait information

Haplotype trend regression

If we know the haplotypes without error and wanted to assess the impact of a certain haplotype on a continuous trait, we could create an explanatory variable that encodes the number of copies of the haplotype in each individual (0, 1, or 2).

We could then fit a regression model with the trait as the outcome and the number of copies of the haplotype as the explanatory variable

In *haplotype trend regression* we use the expected number of copies of the haplotype under consideration conditional on the genotype as the explanatory variable.

Haplotype trend regression

For example, if a subject has 2 possible haplotype pairs $H_1 = (h_1, h_4)$ and $H_2 = (h_2, h_3)$ with probabilities p_1 and p_2 respectively, then the conditional expectation of the number of copies of each member of the pairs is just 1 times these probabilities.

Suppose there were only these 4 observed haplotypes in all subjects and we wanted to test for an effect of all possible haplotypes.

In this case the subject with H_1 and H_2 would have 4 predictor variables with $x_1 = x_4 = p_1$ and $x_2 = x_3 = p_2$.

Haplotype trend regression in R

We can use the usual approach to testing for a difference in the magnitude of the residuals to test for differences between linear models.

This test between regression models is based on an F test and can be done using the anova command as follows.

```
HapMat <- HapDesign(HaploEM)
Trait <- NDRM.CH[Race=="Caucasian" & !is.na(Race)]
mod1 <- (lm(Trait~HapMat))
mod2 <- (lm(Trait~1))
anova(mod2,mod1)
Analysis of Variance Table
Model 1:  Trait   1
Model 2:  Trait   HapMat
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    776 881666
2    766 869272 10     12394 1.0921 0.3653
```

Haplotype trend regression in R

Note that the textbook is in error here as it reports a test with 12 degrees of freedom which means that there were 12 different haplotypes found (I think the text must have used a different `min.posterior` in the call to `haplo.em`).

Haplotype associations via multiple imputation

We have previously noted that substituting the estimated haplotypes and pretending they are known is not valid as such an approach will underestimate the variance.

In *multiple imputation* we repeatedly sample haplotypes and perform the subsequent association testing.

We then average over all of the results from imputing, and if we use the correct standard error we can get a valid statistical procedure.

This standard error must account for the variation given a particular imputed set of haplotypes and the variation arising from all of the possible haplotypes.

Haplotype imputation in R

Note that there is a typo in the text for this example: there is an `h9` where there should be an `h8`.

First we set up the data and create holders for the results from the multiple imputations.

```
Nobs <- sum(Race=="Caucasian", na.rm=T)
Nhap <- length(HaploEM$hap.prob)
D <- 1000
Est <- rep(0,D)
SE <- rep(0,D)
```

Haplotype imputation in R

Then we create a loop to sample haplotypes.

```
for (nimput in 1:D){
  Xmat <- matrix(data=0,nrow=Nobs,ncol=Nhap)
  for (i in 1:Nobs){
    IDSeq <- seq(1:sum(HaploEM$nreps))[HaploEM$indx.subj==i]
    if (length(IDSeq)>1){Samp <- sample(IDSeq,size=1,
      prob=HaploEM$post[IDSeq])}
    if (length(IDSeq)==1){Samp <- IDSeq}
    Xmat[i,HaploEM$hap1code[Samp]] <-1
    Xmat[i,HaploEM$hap2code[Samp]] <-1
  }
  h8 <- Xmat[,8]>=1
  Est[nimput] <- summary(lm(Trait~h8))$coefficients[2,1]
  SE[nimput] <- summary(lm(Trait~h8))$coefficients[2,2]
}
MeanEst <- mean(Est)
Wd <- mean(SE^2)
Bd <- (1/(D-1))*sum((Est-MeanEst)^2)
Td <- Wd + ((D+1)/D)*Bd
nu <- (D-1)*(1 + (1/(D+1))*(Wd/Bd))^2
1-pt(MeanEst/sqrt(Td),df=nu)
[1] 0.06187731
```

Haplotype testing using trait information

If a subject's haplotype is ambiguous, information about a trait is potentially informative about the haplotype.

For instance, it may be that all subjects with one of the possible haplotypes have similar trait values and these trait values differ from those with the other haplotype.

In this case we would conclude that the haplotype for the ambiguous subject is likely the haplotype of those with similar trait values.

Haplotype testing using trait information

The basic idea is to extend the haplotype model based on the multinomial distribution to allow the probabilities of each haplotype to depend on the trait values.

The exact manner in which this dependence occurs depends on the nature of the trait variable: we use logistic regression for binary traits and linear regression for continuous data.

There are methods that allow departures from Hardy Weinberg equilibrium, but the functions we will consider assume Hardy Weinberg equilibrium.

Haplotype testing using trait information in R

We can use the `haplo.glm` function in R much in the same way that we use the regular `glm`, although it doesn't have the complete functionality of `glm`.

We again examine associations between the actinin 3 gene and change in muscle strength.

First we set up the genotype data and our trait then call the function. To view a table of p-values you must use the summary command

```
Geno.C <- setupGeno(Geno.C)
Trait <- NDRM.CH[Race=="Caucasian" & !is.na(Race)]
Dat <- data.frame(Geno.C=Geno.C, Trait=Trait)
> h1 <- haplo.glm(Trait~Geno.C,data=Dat,
+   allele.lev=attributes(Geno.C)$unique.alleles)
```

Haplotype testing using trait information in R

```
> summary(h1)
Call:
haplo.glm(formula = Trait ~ Geno.C, data = Dat, allele.lev =
attributes(Geno.C)$unique.alleles)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-54.982  -24.143   -7.389   16.700  196.348
Coefficients:
                coef       se   t.stat  pval
(Intercept) 50.67787  2.21715 22.85724 0.000
Geno.C.3     8.49595  0.61133 13.89750 0.000
Geno.C.5    -0.44085  7.27971 -0.06056 0.952
Geno.C.8     2.01114  1.89143  1.06329 0.288
Geno.C.9     8.42214  3.50991  2.39953 0.017
Geno.C.rare  3.98509  6.29417  0.63314 0.527
(Dispersion parameter for gaussian family taken to be 1129.036)
    Null deviance:  864661 on 761 degrees of freedom
Residual deviance:  853551 on 756 degrees of freedom
AIC: 7526.6

Number of Fisher Scoring iterations:  47
```

Haplotype testing using trait information in R

```
Haplotypes:
            loc.1 loc.2 loc.3 loc.4 hap.freq
Geno.C.3        C     A     T     G 0.012490
Geno.C.5        C     G     C     G 0.010776
Geno.C.8        T     A     T     G 0.402424
Geno.C.9        T     G     C     A 0.083942
Geno.C.rare     *     *     *     * 0.018135
haplo.base      C     G     C     A 0.472233
Degrees of Freedom:  761 Total (i.e.  Null); 756 Residual
Subjects removed by NAs in y or x, or all NA in geno
  yxmiss genomiss
      14       15
    Null Deviance:  864660
 Residual Deviance:  853550

               AIC: 7526.6
```

Haplotype testing using trait information in R

The most common haplotype is taken as the reference group so that the p-values reported in the output are for testing for a difference between someone with 2 copies of this haplotype and someone with 1 copy of the other haplotypes.

So we conclude that having a single `CATG` increases the percent change in muscle strength by 8.50 ($p < 0.001$) compared to someone with 2 copies of the base haplotype `CGCA` and only 1% of this population has this haplotype.

Haplotype testing using trait information in R

There are some useful options when one uses this function.

For example, we can change the base haplotype. This is useful when comparing across ethnic groups as the most common haplotype may differ.

```
summary(haplo.glm(Trait~Geno.C,data=Dat,
allele.lev=attributes(Geno.C)$unique.alleles,
+   control=haplo.glm.control(haplo.base=9)))
Call:
haplo.glm(formula = Trait ~ Geno.C, data = Dat, control =
haplo.glm.control(haplo.base = 9),
    allele.lev = attributes(Geno.C)$unique.alleles)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-54.982  -24.143   -7.389   16.700  196.348
Coefficients:
                coef       se   t.stat  pval
(Intercept) 67.52215  5.32869 12.67145 0.000
Geno.C.3     0.07381  4.59843  0.01605 0.987
Geno.C.4    -8.42214  3.14163 -2.68082 0.008
Geno.C.5    -8.86299  7.34521 -1.20664 0.228
Geno.C.8    -6.41100  3.07598 -2.08422 0.037
Geno.C.rare -4.43705  6.49639 -0.68300 0.495
```

Haplotype testing using trait information in R

```
(Dispersion parameter for gaussian family taken to be 1129.036)
    Null deviance:  864661 on 761 degrees of freedom
Residual deviance:  853551 on 756 degrees of freedom
AIC: 7526.6
Number of Fisher Scoring iterations:  47
Haplotypes:
             loc.1 loc.2 loc.3 loc.4 hap.freq
Geno.C.3         C     A     T     G  0.01249
Geno.C.4         C     G     C     A  0.47223
Geno.C.5         C     G     C     G  0.01078
Geno.C.8         T     A     T     G  0.40242
Geno.C.rare      *     *     *     *  0.01813

haplo.base       T     G     C     A  0.08394
```

Haplotype testing using trait information in R

Other options include the ability to change the genetic model: for example we may want to allow for a dominance effect rather than the additive effects we have used thus far.

To do this one again uses the `control` argument, but specifies

```
control=haplo.glm.control(haplo.effect="dominant")
```

Cases with missing trait or covariate values are ignored but missing genotype data can be handled with the EM algorithm.

The main drawback of the `haplo.glm` methodology is the assumption of Hardy Weinberg equilibrium.

With the 2 step approach that uses multiple imputation, one can first estimate the haplotype frequencies within each ethnic group then combine data across the ethnic groups to get a more powerful test.