Estimation of haplotypes

Cavan Reilly

October 7, 2013
Table of contents

Bayesian methods for haplotype estimation

Testing for haplotype trait associations
- Haplotype trend regression
- Haplotype associations via multiple imputation
- Haplotype testing using trait information
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.
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
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.

```r
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       | 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.
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)
Then we create a loop to sample haplotypes.

```r
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.
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

```r
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
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 1.89143 4.46329 0.000
 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

<table>
<thead>
<tr>
<th>Geno.C.3</th>
<th>C</th>
<th>A</th>
<th>T</th>
<th>G</th>
<th>0.012490</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geno.C.5</td>
<td>C</td>
<td>G</td>
<td>C</td>
<td>G</td>
<td>0.010776</td>
</tr>
<tr>
<td>Geno.C.8</td>
<td>T</td>
<td>A</td>
<td>T</td>
<td>G</td>
<td>0.402424</td>
</tr>
<tr>
<td>Geno.C.9</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td>A</td>
<td>0.083942</td>
</tr>
<tr>
<td>Geno.C.rare</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0.018135</td>
</tr>
<tr>
<td>haplo.base</td>
<td>C</td>
<td>G</td>
<td>C</td>
<td>A</td>
<td>0.472233</td>
</tr>
</tbody>
</table>

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
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
```

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:

<table>
<thead>
<tr>
<th>loc.1</th>
<th>loc.2</th>
<th>loc.3</th>
<th>loc.4</th>
<th>hap.freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geno.C.3</td>
<td>C</td>
<td>A</td>
<td>T</td>
<td>G</td>
</tr>
<tr>
<td>Geno.C.4</td>
<td>C</td>
<td>G</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>Geno.C.5</td>
<td>C</td>
<td>G</td>
<td>C</td>
<td>G</td>
</tr>
<tr>
<td>Geno.C.8</td>
<td>T</td>
<td>A</td>
<td>T</td>
<td>G</td>
</tr>
<tr>
<td>Geno.C.rare</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>haplo.base</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td>A</td>
</tr>
</tbody>
</table>
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.