Journal of Statistical Software

May 2006, Volume 16, Issue 2. *http://www.jstatsoft.org/*

hapassoc: Software for likelihood inference of trait associations with SNP haplotypes and other attributes

Kelly Burkett
Simon Fraser University

Jinko Graham
Simon Fraser University

Brad McNeney
Simon Fraser University

Abstract

Complex medical disorders, such as heart disease and diabetes, are thought to involve a number of genes which act in conjunction with lifestyle and environmental factors to increase disease susceptibility. Associations between complex traits and single nucleotide polymorphisms (SNPs) in candidate genomic regions can provide a useful tool for identifying genetic risk factors. However, analysis of trait associations with single SNPs ignores the potential for extra information from haplotypes, combinations of variants at multiple SNPs along a chromosome inherited from a parent. When haplotype-trait associations are of interest and haplotypes of individuals can be determined, generalized linear models (GLMs) may be used to investigate haplotype associations while adjusting for the effects of non-genetic cofactors or attributes. Unfortunately, haplotypes cannot always be determined cost-effectively when data is collected on unrelated subjects. Uncertain haplotypes may be inferred on the basis of data from single SNPs. However, subsequent analyses of risk factors must account for the resulting uncertainty in haplotype assignment in order to avoid potential errors in interpretation. To account for such uncertainty, we have developed **hapassoc**, software for R implementing a likelihood approach to inference of haplotype and non-genetic effects in GLMs of trait associations. We provide a description of the underlying statistical method and illustrate the use of **hapassoc** with examples that highlight the flexibility to specify dominant and recessive effects of genetic risk factors, a feature not shared by other software that restricts users to additive effects only. Additionally, **hapassoc** can accommodate missing SNP genotypes for limited numbers of subjects.

Keywords: genetic association, single-nucleotide polymorphisms, haplotypes, generalized linear models, missing data, EM algorithm.

1. Introduction and Background

The identification of genetic factors influencing susceptibility to complex diseases such as cancer and diabetes is important for improving our understanding of disease pathways. Genetic factors are measured at multiple sites of known genomic location (*genetic markers*) to determine if variants at these sites are associated with the disease trait. A common type of genetic marker in association studies is the *single nucleotide polymorphism* (SNP). SNPs generally have only two different variants, called *alleles*, which can be labelled as 0 or 1. For each marker, an individual inherits one allele from their mother, and one from their father. The *genotype* is the particular two alleles that are inherited. The three possible genotypes for a SNP marker are 0/0, 0/1 and 1/1, where "/" is used to separate the alleles inherited from each parent. The 0/0 and 1/1 genotypes are called *homozygous* (both alleles are the same) and the 0/1 genotype is called *heterozygous* (the two alleles are different).

A *haplotype* denotes the alleles at multiple markers that are inherited together on the same chromosome from a parent. For example, if an individual inherits a 0 allele at one SNP and a 0 allele at another SNP from their mother, the haplotype they inherit is 00. Because the SNPs may be close together on the same chromosome, their alleles are not necessarily independent. The *haplotype phase* is the combination of the two haplotypes that an individual inherits. For example, the phased genotype 00/01 indicates that the two haplotypes inherited were 00 and 01; the genotype at the first SNP is 0/0 and the genotype at the second SNP is 0/1.

When associations between haplotypes and disease outcomes or *traits* are of interest, a potential difficulty is that haplotype phase is not necessarily known because, typically, genotypes are only measured at individual markers. For a subject who is heterozygous at one or fewer markers, the haplotype phase can be inferred from the genotypes at individual markers. However, for a subject who is heterozygous at two or more markers, the phase can not be determined with certainty. This is illustrated in figure 1. Current cost-effective genotyping technology gives the observed genotypes at the two SNPs as (0/1, 0/1). The observed genotypes could involve the inheritance of haplotypes 00 and 11, or of haplotypes 01 and 10. Therefore, two possible haplotype phasings, 00/11 and 01/10, are compatible with the observed data.

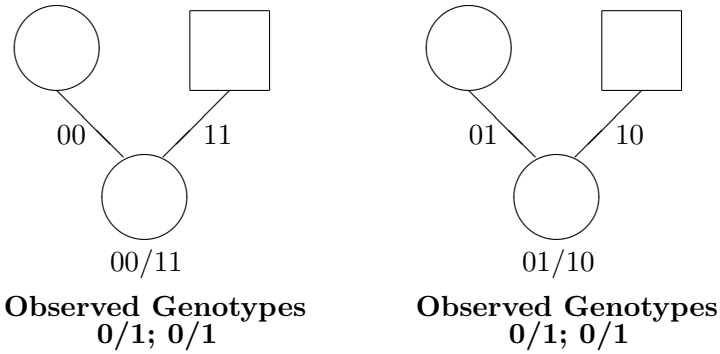

Figure 1: Illustration of unknown haplotype phase for certain observed genotypes

The analysis of haplotype-trait associations therefore involves handling the missing phase data. To describe such associations, Lake *et al.* (2003) and Burkett *et al.* (2004) adopt a *generalized linear model* (GLM) framework (McCullagh and Nelder 1983). GLMs provide the flexibility to incorporate non-genetic risk factors or potential confounding variables such as

age or sex as covariates. They also allow the incorporation of interaction between genetic and environmental risk factors, a current research focus in the study of complex diseases. Finally, GLMs can accommodate a variety of disease outcomes including dichotomous, count and continuous traits. The method of Burkett *et al.* (2004) is implemented in **hapassoc**, a contributed package for R. Standard population- and quantitative-genetic assumptions, such as population Hardy-Weinberg equilibrium and independence of haplotypes and non-genetic covariates, are made. Parameter estimates are obtained by use of an expectation-maximization (EM) algorithm, called the method of weights (Ibrahim 1990), for missing categorical covariates. Standard errors that account for the extra uncertainty due to missing phase are calculated using Louis' method (Louis 1982). The statistical approach implemented in **hapassoc** is briefly described and the use of the program is illustrated with examples.

2. Statistical Description

Let y_i be the measured trait and x_i be the vector of covariates for the ith subject, assuming known haplotype phase. The covariates x_i provide information about the individual's haplotypes and non-genetic cofactors. Let $\mathsf{P}(x_i \mid \gamma)$ be the probability of covariates x_i, parametrized by γ. For notational convenience, suppose γ is a column-vector. A GLM specifies the conditional probability of the trait given the covariates as

$$\mathsf{P}(y_i \mid x_i, \beta, \phi) = \exp\left[\frac{y_i\nu_i - b(\nu_i)}{a_i(\phi)} + c(y_i, \phi)\right],$$

where ν_i is the canonical parameter for a probability distribution in the exponential family, ϕ is a dispersion parameter, $a_i(\phi) = \phi/m_i$ for a known constant m_i, and $b(\cdot)$ and $c(\cdot,\cdot)$ are known functions. A link function $g(\mu_i) = \eta_i$ relates the linear model $\eta_i = x_i^T\beta$ to the mean trait value $\mu_i = b'(\nu_i)$. The regression coefficients β describe the effects of haplotypes and non-genetic cofactors, including possible interactions, on the mean trait value. The parameters that describe the trait and covariate data are thus $\theta = (\beta^T, \phi, \gamma^T)^T$. However, we only consider ϕ and γ as needed for inference about β, which is of primary interest.

If haplotype phase were known and complete covariate information were available on all subjects, the complete-data log-likelihood for the ith individual would be

$$l_{ci}(\theta) = \log\left[\mathsf{P}(y_i \mid x_i, \beta, \phi) \times \mathsf{P}(x_i \mid \gamma)\right].$$

Haplotypes are not necessarily known, and therefore for some subjects the genetic components of the covariate vector x_i are not available. However, the observed genotypes at individual SNPs constrain the potential haplotype configurations. Hereafter, let $x_{obs,i}$ be the observed covariate information for the ith subject and x_i^j be the jth possible covariate vector (i.e. covariates that code haplotype information and non-genetic covariates) consistent with $x_{obs,i}$.

We account for missing haplotype phase by use of the EM algorithm (Dempster *et al.* 1977), in which maximum likelihood estimates $\widehat{\theta}$ are obtained by iteratively maximizing the conditional expectation of the complete-data log-likelihood, $Q(\theta \mid \theta_t)$, given the observed data and current parameter estimates θ_t to determine new parameter estimates θ_{t+1}. Assuming independent subjects, $Q(\theta \mid \theta_t) = \sum_i Q_i(\theta \mid \theta_t)$, where $Q_i(\theta \mid \theta_t) = \mathsf{E}\left[l_{ci}(\theta) \mid y_i, x_{obs,i}, \theta_t\right]$.

Our EM algorithm is based on the method of weights (Ibrahim 1990), an implementation for generalized linear models with categorical covariates that may be missing (see Horton and

Laird 1999, for a review). Let $l_{ci}^j(\theta) = \log \mathsf{P}(y_i, x_i^j \mid \theta)$ be the complete-data log-likelihood for the ith subject if that subject had covariate vector x_i^j. Then

$$Q_i(\theta \mid \theta_t) = \sum_j l_{ci}^j(\theta) \mathsf{P}(x_i^j \mid y_i, x_{obs,i}, \theta_t) = \sum_j l_{ci}^j(\theta)\ w_{ij}(\theta_t),$$

where $w_{ij}(\theta_t) = \mathsf{P}(x_i^j \mid y_i, x_{obs,i}, \theta_t)$ are the "weights". Note that each subject's contribution $Q_i(\theta \mid \theta_t)$ to $Q(\theta \mid \theta_t)$ is summed over the set of covariate vectors consistent with their observed data $(y_i, x_{obs,i})$. The EM algorithm involves computing weights for each possible complete-data covariate vector (E-step), and computing new parameter estimates by maximizing a weighted log-likelihood function (M-step). Upon convergence, standard errors can be calculated. We implemented this EM algorithm and the calculation of standard errors in R (http://www.r-project.org) as described in detail below. Our implementation is freely available on the Comprehensive R Archive Network (CRAN) as a package called **hapassoc**.

2.1. EM algorithm implemented in hapassoc

Our implementation of the algorithm is summarized by the following steps.

1. For each individual with unknown haplotypes, add a "pseudo-individual" for each possible haplotype combination consistent with that individual's observed genotypes. The genetic data for each pseudo-individual is a different haplotype configuration; the non-genetic data is the same for each pseudo-individual. Initial values for regression coefficients and haplotype frequencies are set.

2. At the tth iteration:

   (a) E-step: Compute the expected log-likelihood conditional on the observed data and the current parameter estimates $\theta_t = (\beta_t^T, \phi_t, \gamma_t^T)^T$.

   This step simplifies to weighting each pseudo-individual by the conditional probability of the corresponding haplotype configuration given the observed data:

$$w_{ij}(\theta_t) = \mathsf{P}(x_i^j \mid y_i, x_{obs,i}\ \theta_t) = \frac{\mathsf{P}(y_i \mid x_i^j, \beta_t, \phi_t)\mathsf{P}(x_i^j \mid \gamma_t)}{\sum_k \mathsf{P}(y_i \mid x_i^k, \beta_t, \phi_t)\mathsf{P}(x_i^k \mid \gamma_t)}.$$

   Calculation of $\mathsf{P}(y_i \mid x_i^j, \beta_t, \phi_t)$ is straightforward using the GLM and the estimates β_t and ϕ_t from fitting the model in the M-step at iteration t. To specify the covariate distribution $\mathsf{P}(x_i^j \mid \gamma_t)$, two standard assumptions from the genetics literature are made. The first is independence of the genetic and non-genetic factors:

$$\mathsf{P}(x_i^j \mid \gamma_t) = \mathsf{P}(x_{gi}^j \mid \gamma_t)\mathsf{P}(x_{ei}^j \mid \gamma_t),$$

   where x_{gi}^j and x_{ei}^j code, respectively, the haplotype and non-genetic covariate information. If only the haplotype information is missing, this assumption means that

$$w_{ij}(\theta_t) = \frac{\mathsf{P}(y_i \mid x_i^j, \beta_t, \phi_t)\mathsf{P}(x_{gi}^j \mid \gamma_t)}{\sum_k \mathsf{P}(y_i \mid x_i^k, \beta_t, \phi_t)\mathsf{P}(x_{gi}^k \mid \gamma_t)} \equiv w_{ij}(\beta_t, \phi_t, \gamma_{gt})$$

so that only the distribution $\mathsf{P}(x_{gi}^j|\ \gamma_t)$ has to be specified. Such a simplification is convenient because specifying $\mathsf{P}(x_{ei}^j|\ \gamma_t)$ can be difficult for non-genetic covariates that are continuous, for example.

In general, the frequencies of haplotype configurations are not identifiable from the genotypes at individual SNPs. For example, haplotype configurations resulting in two or more heterozygous SNP genotypes are never observed unambiguously. However, under Hardy-Weinberg proportions (HWP), the probability of a haplotype configuration can be written in terms of the haplotype frequencies, allowing both the frequencies of haplotypes and of haplotype configurations to be estimated from genotypes at individual SNPs. The assumption of HWP holds if the haplotype inherited from the mother is independent of the haplotype inherited from the father. Assuming HWP also leads to a considerable reduction in the number of parameters γ_g describing the haplotype covariate distribution $\mathsf{P}(x_{gi} \mid \gamma_g)$. For example, when considering haplotypes of 3 SNPs, there are $2^3 = 8$ possible haplotypes compared to $8(9)/2 = 36$ possible phase configurations. The substantive reduction in the number of genetic parameters leads to faster convergence of the EM algorithm and greater stability of estimation.

(b) M-step: Maximize the conditional expected log-likelihood $Q(\theta \mid \theta_t)$ to determine the $(t+1)$th parameter estimate.

Under the assumption of independence of the genetic and non-genetic covariates, $Q(\theta \mid \theta_t)$ may be re-expressed as a sum of components involving one of either (β, ϕ), γ_g or γ_e:

$$\begin{aligned}
Q(\theta \mid \theta_t) &= \sum_i Q_i(\theta \mid \theta_t) = \sum_i \sum_j w_{ij}(\beta_t, \phi_t, \gamma_{gt}) l_{ci}^j(\theta) \\
&= \sum_i \sum_j w_{ij}(\beta_t, \phi_t, \gamma_{gt}) \log \left[\mathsf{P}(y_i \mid x_i^j, \beta, \phi) \mathsf{P}(x_{gi}^j \mid \gamma_g) \mathsf{P}(x_{ei} \mid \gamma_e) \right] \\
&= \sum_i \sum_j w_{ij}(\beta_t, \phi_t, \gamma_{gt}) \log \mathsf{P}(y_i \mid x_i^j, \beta, \phi) \\
&\quad + \sum_i \sum_j w_{ij}(\beta_t, \phi_t, \gamma_{gt}) \log \mathsf{P}(x_{gi}^j \mid \gamma_g) \\
&\quad + \sum_i \sum_j w_{ij}(\beta_t, \phi_t, \gamma_{gt}) \log \mathsf{P}(x_{ei} \mid \gamma_e) \\
&\equiv Q(\beta, \phi \mid \beta_t, \phi_t, \gamma_{gt}) + Q(\gamma_g \mid \beta_t, \phi_t, \gamma_{gt}) + Q(\gamma_e \mid \beta_t, \phi_t, \gamma_{gt})
\end{aligned}$$

Therefore, each component is maximized separately to update the estimates of β, ϕ and γ_g.

The summand $Q(\beta, \phi \mid \beta_t, \phi_t, \gamma_{gt})$ is a weighted log-likelihood for regression and dispersion parameters in a GLM. The R `glm` function is used to estimate β_{t+1} with the $w_{ij}(\beta_t, \phi_t, \gamma_{gt})$ input to the `glm` function through the "weights" option. The summand $Q(\gamma_g \mid \beta_t, \phi_t, \gamma_{gt})$ is a weighted multinomial log-likelihood and so its maximization is also straightforward. Since estimates of γ_e are not required for the weights or for updating estimates of the other parameters, γ_e can be ignored when finding maximum-likelihood estimates of the regression parameters β.

3. Repeat E- and M-steps until convergence.

4. Calculate the standard errors using values from the final GLM.

   The variance-covariance matrix of $\widehat{\theta}$ can be approximated by the inverse of the observed information matrix $I(\widehat{\theta})$. Louis (1982) gives an expression for the observed information in missing data problems. With ambiguous haplotype configurations, the expression consists of a term for the expected complete-data information given the observed genotypes minus a penalty for unknown phase (Schaid *et al.* 2002). Factorization of the likelihood implies that only the observed information $I(\widehat{\beta}, \widehat{\phi}, \widehat{\gamma_g})$ is required to obtain standard errors for $\widehat{\beta}$. The variance of $\widehat{\beta}$ is approximated by the appropriate submatrix of $I^{-1}(\widehat{\beta}, \widehat{\phi}, \widehat{\gamma_g})$. Appendix A provides details of the derivation of $I^{-1}(\widehat{\beta}, \widehat{\phi}, \widehat{\gamma_g})$.

3. Using hapassoc and its features

Our contributed R package **hapassoc** can be downloaded from the CRAN webpage using the install command at the R command line (>):

```
> install.packages('hapassoc')
```

hapassoc does not depend on any optional R packages. After it has been installed, the package may be loaded with:

```
> library(hapassoc)
```

There are four main functions for the user of **hapassoc**: `pre.hapassoc`, `hapassoc`, `summary.hapassoc` and `anova.hapassoc`. The function `pre.hapassoc` takes the input dataset, with columns of genotype data, and outputs an augmented dataset consisting of all pseudo-individuals. The function `hapassoc` fits the user-specified generalized linear model, with haplotypes as covariates. The `summary` function provides a summary of the results while `anova` returns a likelihood ratio test of two nested models fit with `hapassoc`. Further details on each of these functions are now given.

The function `pre.hapassoc` pre-processes the input dataset for `hapassoc`. This pre-processing involves converting the genotype data into haplotype data and adding rows to the input dataset representing haplotype configurations compatible with the observed genotype data for a subject. If the haplotype configuration of a subject is known, no extra rows are added for this subject. The pre-processing also involves obtaining initial estimates for haplotype frequencies based on the genotype data from all subjects combined. Haplotypes with non-negligible frequency below a user-defined pooling tolerance (`pooling.tol`) are then grouped together. The function requires the input dataset to have the following form:

1. Each row represents an individual. Columns represent the trait (response), non-genetic covariate information and genotype information.

2. The columns of genotype information may have one of two formats. For the "allelic" format, each genotype is denoted by two columns, one column for each allele, with alleles coded as either 0 or 1. For the "genotypic" format, the genotype at each locus is given in one column by a character string or factor. Only two possible variants or alleles are

permitted, forming three possible genotypes. Genotypes with more than two alleles will cause `pre.hapassoc` to issue an error and stop execution. Examples of both allelic and genotypic formats are given in section 4.

3. If there are n SNPs, the final n columns of the "genotypic" dataset or $2n$ columns of the "allelic" dataset contain the genotype data. By default, `pre.hapassoc` prints a list of genotype variables used to form haplotypes and a list of other "non-genetic" variables passed to the function.

4. There are no restrictions on the non-genetic covariate columns, except that missing data should be coded as `NA`. Note that only missing genotype data is handled by **hapassoc**; individuals with missing non-genetic data will be removed from the dataset and a warning will be given to the user.

5. Missing allelic data should be coded as `NA` or ' '. Missing genotypic data should be coded as, e.g., 'A', if one allele is missing and the other allele is the nucleotide adenine (say), and ' ' or `NA` if both alleles are missing.

This function has a number of optional parameters that may be set by the user. The option `maxMissingGenos` specifies the maximum number of SNPs which may be missing genotypes for a subject in order for that subject to be included in the analysis, with a default of 1. Each SNP with a missing genotype leads to more haplotype configurations being added for that subject in the augmented dataset. To avoid convergence problems due to haplotypes of low frequency, `pooling.tol` can be used to specify a threshold frequency below which haplotypes will be pooled into a single category. The threshold is applied to the initial haplotype frequencies provided by `pre.hapassoc`. The "pooled" category returned by `pre.hapassoc` is not updated with the haplotype frequencies from subsequent iterations of the EM algorithm in `hapassoc`. The default pooling threshold is set to 0.05. Similarly, the parameter `zero.tol` gives the frequency below which it is assumed that a haplotype does not exist. The default threshold is the inverse of ten times the total number of haplotypes in all subjects. Pseudo-individuals carrying haplotypes of estimated frequency below this zero threshold will be removed from the augmented dataset. The `verbose` flag, if `TRUE` (the default), causes `pre.hapassoc` to print a list of the genotype variables used to form haplotypes and a list of other non-genetic variables.

The function `pre.hapassoc` will return a list with components required for `hapassoc`. The components include two data frames, `haploMat` and `haploDM`, with the haplotype information. The data frame `haploMat` consists of two columns, one for each of the inferred haplotypes, and a row for each pseudo-individual in the dataset. The data frame `haploDM` consists of a column for each of the haplotypes inferred to be present in the dataset and a row for each pseudo-individual. Each pseudo-individual is given a code of 0, 1 or 2 in each column for the number of copies they have of that haplotype. Therefore, all rows of `haploDM` should sum to 2, as each individual can have only two haplotypes. The non-haplotype data is returned in the data frame `nonHaploDM`. Some rows of `nonHaploDM` may be identical, as they represent the trait value and non-genetic covariate information for the same individual who has multiple haplotype configurations compatible with their observed genotype data. Other components of the list returned by `pre.hapassoc` include a vector `initFreq` of initial estimates of all haplotype frequencies (including pooled haplotypes), vectors `zeroFreqHaplos`

and pooledHaplos of the zero-frequency and pooled haplotypes, respectively, and a vector wt of the initial estimates for the weights. For more information on this function, type

```
> help(pre.hapassoc)
```

The function hapassoc uses the list returned from pre.hapassoc as input, and fits the linear model provided in the model-formula argument. The model formula is specified the same way as it would be for the glm or lm functions in R. The elements of the model formula must have names from among the names of columns in the data frames nonHaploDM and haploDM. The availability of haploDM provides the user with the flexibility to specify recessive, dominant or other types of genetic risk models. See the examples in section 4 below for more details on specifying the model equation. As with glm, the family of the GLM is specified with the family parameter. Currently the binomial, gaussian, poisson, and Gamma families are supported by hapassoc. Unlike glm, we have opted for a binomial family as the default, since in most of the biomedical applications we have encountered binary traits (e.g. diseased/not diseased) have been investigated. Other optional parameters can be set including the initial estimates of haplotype frequencies and of regression coefficients used to start the EM algorithm, the maximum number of iterations for the algorithm, the tolerance for convergence, and a verbose flag that controls printing of the iteration number and the value of the covergence criterion at each iteration.

The function returns a list of class "hapassoc" with components such as the estimated regression coefficients, the estimated haplotype frequencies and an estimate of their joint variance-covariance matrix. The associated help file, found by typing help(hapassoc), provides a detailed description of the components of the returned list. The function summary.hapassoc is the summary method for the class and provides a nicer way of viewing the results. The summary includes a table of the estimated regression coefficients and the associated standard errors, Z scores and p-values, a table of the estimated haplotype frequencies and their standard errors, and the estimated dispersion parameter (when appropriate). Likelihood ratio tests to compare two nested models fit with hapassoc may be performed with anova.hapassoc.

4. Examples

This section illustrates the use of hapassoc with examples of logistic and linear regression, when input genotypes are in "allelic" and "genotypic" format, respectively.

4.1. Logistic regression with input genotypes in "allelic" format

The "hypoDat" dataset from the hapassoc package will be used for this example. The dataset can be loaded into R with the command

```
> data(hypoDat)
```

The first five rows of the dataset are shown

```
> hypoDat[1:5,]
```

```
  affected       attr M1.1 M1.2 M2.1 M2.2 M3.1 M3.2
1        0 -1.0457877    0    0    0    1    1    1
2        0 -1.3604013    0    0    0    0    0    1
3        0  0.5672941    0    0    0    1    0    1
4        0 -0.2220068    1    1    0    0    0    0
5        0  1.8051683    0    0    0    1    0    1
```

The data consists of eight columns, the affection status (0/1), a continuous non-genetic covariate, and information on three diallelic genetic markers. The genotype data is in "allelic" format, so there are six genotype columns for the three SNPs. The genotype of the first SNP is represented by the two columns M1.1 and M1.2. For example, the fourth individual has genotype 1/1 (two copies of allele 1) at the first SNP.

The genotype data is converted to haplotype data by the `pre.hapassoc` function. Data frames passed to `pre.hapassoc` are assumed to contain the response, an arbitrary number of non-genetic covariates and the genotype data. The user must tell the function explicitly how to separate the genetic and non-genetic data by specifying `numSNPs` and by passing a data frame whose last 2×`numSNPs` (allelic format) or last `numSNPs` (genotypic format) columns comprise the data on genotypes. The data frame `hypoDat` is in allelic format and so the appropriate call is

```
> example1.haplos <- pre.hapassoc(hypoDat,numSNPs=3)

Haplotypes will be based on the following SNPs (allelic format):
 SNP 1: M1.1/M1.2
 SNP 2: M2.1/M2.2
 SNP 3: M3.1/M3.2
Remaining variables are:
 affected, attr
```

As an aside, if only the first and third SNPs were of interest for haplotype effects, the call to `pre.hapassoc` could be modified to

```
> example2.haplos <- pre.hapassoc(hypoDat[,c(1:4,7:8)],numSNPs=2)

Haplotypes will be based on the following SNPs (allelic format):
 SNP 1: M1.1/M1.2
 SNP 2: M3.1/M3.2
Remaining variables are:
 affected, attr
```

Similarly, if only the first two SNPs were of interest the call would be

```
> example2.haplos <- pre.hapassoc(hypoDat[,1:6],numSNPs=2)

Haplotypes will be based on the following SNPs (allelic format):
 SNP 1: M1.1/M1.2
 SNP 2: M2.1/M2.2
Remaining variables are:
 affected, attr
```

Finally, if the last two SNPs were of interest the possible calls are

```
> example2.haplos <- pre.hapassoc(hypoDat[,c(1:2,5:8)],numSNPs=2)

Haplotypes will be based on the following SNPs (allelic format):
 SNP 1: M2.1/M2.2
 SNP 2: M3.1/M3.2
Remaining variables are:
 affected, attr

> example2.haplos <- pre.hapassoc(hypoDat,numSNPs=2)

Haplotypes will be based on the following SNPs (allelic format):
 SNP 1: M2.1/M2.2
 SNP 2: M3.1/M3.2
Remaining variables are:
 affected, attr, M1.1, M1.2
```

We prefer the first of these calls since it will exclude the data on the first SNP, `M1.1` and `M1.2`, from the non-haplotype data frame `nonHaploDM` output by `pre.hapassoc`. The second call includes the columns `M1.1` and `M1.2` describing the first SNP in `nonHaploDM`.

The first two individuals in the `hypoDat` dataset have haplotype configurations that can be inferred with certainty. However, the haplotype configuration of the third individual is uncertain. This subject either has haplotypes 000/011 or 001/010. The uncertain haplotype phasing for this subject is reflected in the data frames `haploDM` and `nonHaploDM` returned by `pre.hapassoc`. The rows of `haploDM` corresponding to the first five rows of `hypoDat` are

```
> example1.haplos$haploDM[1:7,]

  h000 h001 h010 h011 h100 pooled
1    0    1    0    1    0      0
2    1    1    0    0    0      0
3    1    0    0    1    0      0
4    0    1    1    0    0      0
5    0    0    0    0    2      0
6    1    0    0    1    0      0
7    0    1    1    0    0      0
```

Rows three and four of `haploDM` correspond to row three of `hypoDat` for the third subject with the uncertain haplotype configuration. The column for the "pooled" category of `haploDM` is comprised of haplotypes 101, 110 and 111 (see the estimated haplotype frequencies below). The rows of `nonHaploDM` corresponding to the first five subjects in `hypoDat` are:

```
> example1.haplos$nonHaploDM[1:7,]
```

```
  affected       attr
1        0 -1.0457877
2        0 -1.3604013
3        0  0.5672941
4        0  0.5672941
5        0 -0.2220068
6        0  1.8051683
7        0  1.8051683
```

As with `HaploDM`, rows three and four of `nonHaploDM` are pseudo-individuals that correspond to the subject in row three of `hypoDat`, and therefore have the same trait and non-genetic covariate data.

The initial estimates of haplotype frequencies are

```
> example1.haplos$initFreq

      h000       h001       h010       h011       h100       h101       h110
0.25179111 0.26050418 0.23606001 0.09164470 0.10133627 0.02636844 0.01081260
      h111
0.02148268
```

Haplotypes h101, h110, and h111 all have frequencies below the default pooling threshold of 0.05 and so they comprise the pooled category in `example1.haplos$haploDM`.

Logistic regression is now performed to determine the effects of haplotypes on affection status, after adjusting for the non-genetic covariate "attr".

```
> example1.regr1<-hapassoc(affected ~ attr+h000+h010+h011+h100+pooled,
+                          example1.haplos,family=binomial())
```

The same regression can be fit with

```
> example1.regr<-hapassoc(affected ~ .,baseline="h001",
+                         example1.haplos,family=binomial())
```

where the "." in the model formula is an R short form for the other columns in the data matrix. Haplotype h001 was chosen as the baseline haplotype in this example because it is the most frequent (the default in `hapassoc`). Users are free to choose their own baseline haplotype, but are cautioned that using a rare haplotype (or the pooled category, if rare) as the baseline can lead to unstable parameter estimates.

Other formula syntax is supported in the hapassoc function. For example,

```
> example1.regr<-hapassoc(affected == 0 ~ attr+h000+h010+h011+h100+pooled,
+                         example1.haplos,family=binomial())
```

can be used to specify that the probability that affection status is 0 is modelled, rather than 1. This feature can be useful if the binary outcome is specified by character strings, such

as "yes"/"no", since the formula may then be specified as `hapassoc(affected == 'yes' ~ ...)`
Additionally, different models for haplotype effects can be fit using R formula syntax. The default is an additive effect for each copy of the haplotype on the scale of the linear predictor. The following command fits recessive effects for haplotypes h000 and h001 (i.e., the haplotype has an effect only if the individual has two copies).

```
> example1.regr2 <- hapassoc(affected ~ attr + I(h000==2) + I(h001==2),
+                            example1.haplos, family=binomial())
```

The output of `hapassoc` may be summarized with:

```
> summary(example1.regr)

Call:
hapassoc(form = affected == 0 ~ attr + h000 + h010 + h011 + h100 +
    pooled, haplos.list = example1.haplos, family = binomial())

Number of subjects used in analysis: 100

Coefficients:
               Estimate Std. Error      zscore   Pr(>|z|)
(Intercept)  1.24304712  0.7826121  1.58833114 0.11221148
attr        -0.74117082  0.2919206 -2.53894658 0.01111868
h000        -1.15178187  0.5942128 -1.93833229 0.05258270
h010         0.59528765  0.6570407  0.90601329 0.36492882
h011         0.03426242  0.9155250  0.03742379 0.97014710
h100         0.85628090  1.0199383  0.83954187 0.40116530
pooled      -0.38745841  0.8775181 -0.44153893 0.65882289

Estimated haplotype frequencies:
         Estimate Std. Error
f.h000 0.26732097 0.03930403
f.h001 0.25180046 0.03865720
f.h010 0.21981267 0.03877367
f.h011 0.10106590 0.02949351
f.h100 0.09501225 0.02369630
f.h101 0.02586632 0.01412337
f.h110 0.01785410 0.01384235
f.h111 0.02126733 0.01247509

(Dispersion parameter for binomial family taken to be 1)

Log-likelihood: -322.155788349086
```

A global hypothesis test for haplotype effects may be performed with:

```
> example1.regr2 <- hapassoc(affected ~ attr, example1.haplos, family=binomial())
> anova(example1.regr1,example1.regr2)

        hapassoc: likelihood ratio test

Full model: affected ~ attr + h000 + h010 + h011 + h100 + pooled
Reduced model: affected ~ attr

LR statistic = 11.2107 , df = 5 , p-value =  0.0474
```

4.2. Linear regression with input genotypes in "genotypic" format

An example of the "genotypic" input format is found in the dataset `hypoDatGeno`. The first five rows are shown below.

```
> data(hypoDatGeno)
> hypoDatGeno[1:5,]

  affected       attr M1 M2 M3
1        0 -1.0457877 AA AC CC
2        0 -1.3604013 AA AA AC
3        0  0.5672941 AA AC AC
4        0 -0.2220068 CC AA AA
5        0  1.8051683 AA AC AC
```

In this case, the alleles at all three SNPs are now represented with A's and C's rather than 0's and 1's. Additionally, the genotype at each SNP is given in one column.

To illustrate how `hapassoc` handles missing genotype data, we introduce missing genotypic information for SNPs M1 and M2, as shown below. We modify the first subject's genotype at M1 so that it is completely missing. We modify the second subject's genotype at M2 so that only one allele, an 'A', is known. Since the data frame `hypoDataGeno` is in genotypic format, M2 is a factor. Therefore, before modifying the second subject's M2 genotype, we must add a level 'A' to M2:

```
> hypoDatGeno[1,"M1"]<-NA #First subject missing genotype at M1
> #Add a level 'A' to M2
> levels(hypoDatGeno[,"M2"])<-c(levels(hypoDatGeno[,"M2"]),"A")
> hypoDatGeno[2,"M2"]<-"A" #Modify second subject's genotype
```

The data in `hypoDatGeno` is now pre-processed by `pre.hapassoc`. For illustration, the optional parameter `maxMissingGenos` has been changed from its default value of one to two.

```
> example2.haplos<-pre.hapassoc(hypoDatGeno,3,maxMissingGenos=2,allelic=F)

Haplotypes will be based on the following SNPs (genotypic format):
 M1, M2, M3
Remaining variables are:
 affected, attr
```

```
> example2.haplos$nonHaploDM[142:148,]

    affected      attr
142        0 -1.045788
143        0 -1.045788
144        0 -1.045788
145        0 -1.045788
146        0 -1.360401
147        0 -1.360401
148        0 -1.360401
```

When there are no more than `maxMissingGenos` SNPs with missing genotype data, rows corresponding to haplotype configurations compatible with the missing genotype data for a subject are added to the end of the augmented data matrices `haploDM` and `nonhaploDM` returned by `pre.hapassoc`. For example, rows 142 to 145 of `nonHaploDM` correspond to the first subject in `hypoDatGeno` and rows 146 to 148 of `nonHaploDM` correspond to the second subject in `hypoDatGeno`. When there are more missing genotypes than are allowed by the user, as in the following command, a warning message is given to indicate that some individuals have been removed from the dataset.

```
> pre.hapassoc(hypoDatGeno,3,maxMissingGenos=0,allelic=F,verbose=FALSE)

Warning message:
2 subjects with missing data in more than 0 genotype(s) removed
 in: handleMissings(SNPdat, nonSNPdat, numSNPs, maxMissingGenos)
```

With SNP data in genotypic format, the haplotypes are now denoted by combinations of the A and C alleles at each SNP, as can be seen in the display of the initial haplotype frequency estimates.

```
> example2.haplos$init

      hAAA       hAAC       hACA       hACC       hCAA       hCAC       hCCA
0.24859595 0.26034040 0.23949970 0.09008737 0.10150449 0.02672430 0.01039986
      hCCC
0.02284793
```

To illustrate linear regression, the continuous non-genetic covariate `attr` will be used as a continuous trait in the call to `hapassoc`:

```
> example2.regr<-hapassoc(attr~hAAA+hACA+hACC+hCAA+pooled,
+                         example2.haplos,family=gaussian())
> summary(example2.regr)

Call:
hapassoc(form = attr ~ hAAA + hACA + hACC + hCAA + pooled, haplos.list = example2.haplos,
```

```
    family = gaussian())

Number of subjects used in analysis: 100

Coefficients:
                Estimate Std. Error      zscore  Pr(>|z|)
(Intercept) -0.198910820  0.3145337 -0.63239909 0.5271261
hAAA        -0.003337329  0.2422522 -0.01377626 0.9890085
hACA        -0.005089539  0.2213682 -0.02299129 0.9816572
hACC         0.423914584  0.4979658  0.85129263 0.3946068
hCAA         0.300230129  0.2938526  1.02170329 0.3069214
pooled       0.019130817  0.3311376  0.05777302 0.9539294

Estimated haplotype frequencies:
          Estimate Std. Error
f.hAAA 0.248400660 0.03590094
f.hAAC 0.257565353 0.03806208
f.hACA 0.241325932 0.03550967
f.hACC 0.090867626 0.02673329
f.hCAA 0.101119710 0.02355280
f.hCAC 0.030216413 0.01611531
f.hCCA 0.009153699 0.00911681
f.hCCC 0.021350608 0.01213278

(Dispersion parameter for gaussian family taken to be 1.009584127756)

Log-likelihood: -410.627503879034
```

The `dispersion` reported by the summary function is the maximum likelihood estimate of the error variance in the linear model.

5. Summary

hapassoc is an R package for haplotype-trait associations in the presence of uncertain haplotype phase, for dichotomous or continuous traits. It uses an EM algorithm called the method of weights to handle the missing haplotype-phase information. Non-genetic attributes, environmental covariates and haplotype-environment interactions can be included in generalized linear models of trait associations. Assumptions of Hardy-Weinberg proportions and independence of genetic and non-genetic covariates are made. In addition to additive effects, dominant and recessive effects of the haplotype risk factors can also be specified in **hapassoc**, in contrast to similar haplotype-trait association software. Missing genotype data is also handled by **hapassoc** so that those individuals with a limited amount of missing genotype data are not removed during the analysis.

hapassoc relies on many of the R base functions, and for that reason it uses much of the same syntax. For someone familiar with the R `glm` and `lm` functions, using **hapassoc** should be relatively straightforward, as the specification of model formulae is identical, the output of

the `summary` method is similar and the `anova` method is similar when used to compare two nested models. Currently `anova.hapassoc` differs from the `anova` methods for `lm` and `glm` in that `anova.hapassoc` requires exactly two models as input. By contrast, `anova.lm` and `anova.glm` take either one fitted model, in which case a sequence of possible sub-models are compared, or an arbitrary number of nested models. Work to make `anova.hapassoc` behave more like `anova.lm` and `anova.glm` is underway and will be included in a future version of **hapassoc**. As an alternative to likelihood ratio tests, users may construct the appropriate Wald statistics in the usual way, with the estimated coefficients and variance-covariance matrix returned by the `hapassoc` function.

Acknowledgements

We would like to thank Matthew Pratola for optimizing the R code and initiating its conversion to an R package, and Sigal Blay for code optimization and package maintenence. We would also like to thank the **hapassoc** users for their valuable feedback in bug reports. This research was supported by Natural Sciences and Engineering Research Council of Canada grants 227972-00 and 213124-03, by Juvenile Diabetes Research Foundation International grant 1-2001-873, by Canadian Institutes of Health Research grants NPG-64869, ATF-66667 and GEI-53960, and in part by the Mathematics of Information Technology and Complex Systems, Canadian National Centres of Excellence. JG is a Scholar of the BC Michael Smith Foundation for Health Research.

References

Burkett K (2002). *Logistic Regression with Missing Haplotypes*. Master's thesis, Statistics and Actuarial Science: Simon Fraser University.

Burkett K, McNeney B, Graham J (2004). "A Note on Inference of Trait Associations with SNP Haplotypes and Other Attributes in Generalized Linear Models." *Human Heredity*, **57**, 200–206.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**, 1–38. With discussion.

Horton NJ, Laird NM (1999). "Maximum Likelihood Analysis of Generalized Linear Models with Missing Covariates." *Statistical Methods in Medical Research*, **8**, 37–50.

Ibrahim JG (1990). "Incomplete Data in Generalized Linear Models." *Journal of the American Statistical Association*, **85**, 765–769.

Lake SL, Lyon H, Tantisira K, Silverman EK, Weiss ST, Laird NM, Schaid DJ (2003). "Estimation and Tests of Haplotype-Environment Interaction when Linkage Phase is Ambiguous." *Human Heredity*, **55**, 56–65.

Lipsitz SR, Ibrahim JG (1996). "A Conditional Model for Incomplete Covariates in Parametric Regression Models." *Biometrika*, **83**, 916–922.

Louis TA (1982). "Finding the Observed Information Matrix when Using the EM Algorithm." *Journal of the Royal Statistical Society B*, **44**, 226–233.

McCullagh P, Nelder JA (1983). *Generalized Linear Models*. Chapman and Hall, London.

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA (2002). "Score Tests for Association Between Traits and Haplotypes when Linkage Phase is Ambiguous." *American Journal of Human Genetics*, **70**, 425–434.

Appendix: Details on the computation of the standard errors

>From standard likelihood theory (e.g. McCullagh and Nelder 1983), the variance-covariance matrix of the maximum likelihood estimators $\widehat{\theta}$ can be approximated by the inverse of the Fisher information matrix

$$\mathcal{I}(\theta_0) = \mathsf{E}\left[I(\theta_0)\right] \equiv \mathsf{E}\left[-\frac{\partial^2}{\partial\theta\partial\theta^T} l(\theta_0)\right]$$

where θ_0 is the "true" value of θ and $l(\theta)$ is the observed-data log-likelihood. The usual estimate of $\mathcal{I}(\theta_0)$ is the observed information $I(\widehat{\theta})$.

Under independence of genetic and non-genetic covariates, the observed-data likelihood $L(\theta)$ factors into a term involving β, ϕ and γ_g and another term involving γ_e:

$$\begin{aligned}
L(\theta) = \prod_i \mathsf{P}(y_i, x_{obs,i} \mid \theta) &= \prod_i \sum_j \mathsf{P}(y_i, x_i^j \mid \theta) \\
&= \prod_i \sum_j \mathsf{P}(y_i \mid x_i^j, \beta, \phi)\mathsf{P}(x_{gi}^j \mid \gamma_g)\mathsf{P}(x_{ei} \mid \gamma_e) \\
&= \prod_i \mathsf{P}(x_{ei} \mid \gamma_e) \sum_j \mathsf{P}(y_i \mid x_i^j, \beta, \phi)\mathsf{P}(x_{gi}^j \mid \gamma_g) \\
&= \left[\prod_i \sum_j \mathsf{P}(y_i \mid x_i^j, \beta, \phi)\mathsf{P}(x_{gi}^j \mid \gamma_g)\right] \times \prod_i \mathsf{P}(x_{ei} \mid \gamma_e)
\end{aligned}$$

The observed information $I(\theta)$ is therefore block-diagonal:

$$I(\theta) = \begin{bmatrix} I(\beta, \phi, \gamma_g) & 0 \\ 0 & I(\gamma_e) \end{bmatrix}, \qquad \text{and so} \qquad I^{-1}(\theta) = \begin{bmatrix} I^{-1}(\beta, \phi, \gamma_g) & 0 \\ 0 & I^{-1}(\gamma_e) \end{bmatrix}.$$

Thus, only $I(\beta, \phi, \gamma_g)$ is required for inference of β.

Information matrix for β, ϕ and γ_g

Louis (1982) derived an expression for the observed information in missing data problems in terms of complete-data score vectors and information matrices. Let $S_c(\beta, \phi, \gamma_g)$ and $S_{ci}(\beta, \phi, \gamma_g)$ be the complete-data score functions for all subjects and for the ith subject, respectively. Define $I_c(\beta, \phi, \gamma_g)$ and $I_{ci}(\beta, \phi, \gamma_g)$ to be the corresponding observed information matrices. Further, let $S_{ci}^j(\beta, \phi, \gamma_g)$ and $I_{ci}^j(\beta, \phi, \gamma_g)$ be, respectively, the complete-data score function and observed information for the ith subject with covariate vector x_i^j. Under

the linear model, recall $\eta_i^j = {x_i^j}^T\beta$ and $\mu_i^j = g^{-1}(\eta_i^j) = b'(\nu_i^j)$. Define $d_{ij} = y_i\nu_i^j - b(\nu_i^j)$ and let $c'(y,\phi)$ and $c''(y,\phi)$ denote the first and second derivatives of $c(y,\phi)$ with respect to ϕ. Then $S_{ci}^j(\beta,\phi,\gamma_g)^T = \left[S_{ci}^j(\beta)^T, S_{ci}^j(\phi), S_{ci}^j(\gamma_g)^T\right]$ where

$$\begin{aligned} S_{ci}^j(\beta) &= \frac{\partial}{\partial\beta}\log P(y_i \mid x_i^j,\beta,\phi) = \frac{m_i(y_i-\mu_i^j)}{\phi}\, x_i^j, \\ S_{ci}^j(\phi) &= \frac{\partial}{\partial\phi}\log P(y_i \mid x_i^j,\beta,\phi) = -\frac{m_i d_{ij}}{\phi^2} + c'(y_i,\phi) \\ \text{and}\quad S_{ci}^j(\gamma_g) &= \frac{\partial}{\partial\gamma_g}\log P(x_i^j \mid \gamma_g). \end{aligned}$$

The observed information $I_{ci}^j(\beta,\phi,\gamma_g)$ has three submatrices along its diagonal. In the top left-hand corner,

$$I_{ci}^j(\beta,\beta) = -\frac{\partial^2}{\partial\beta\partial\beta^T}\log P(y_i \mid x_i^j,\beta,\phi) = \frac{m_i\, x_i^j\, {x_i^j}^T}{\phi\, g'(\mu_i^j)},$$

where $g'(\mu_i^j)$ is the derivative of the link function with respect to μ, evaluated at μ_i^j. The next diagonal submatrix is

$$I_{ci}^j(\phi,\phi) = -\frac{\partial^2}{\partial\phi\partial\phi}\log P(y_i \mid x_i^j,\beta,\phi) = -\frac{2m_i d_{ij}}{\phi^3} - c''(y_i,\phi).$$

Finally, in the bottom right-hand corner

$$I_{ci}^j(\gamma_g,\gamma_g) = -\frac{\partial^2}{\partial\gamma_g\partial\gamma_g^T}\log P(x_i^j \mid \gamma_g).$$

The three off-diagonal submatrices are $I_{ci}^j(\beta,\gamma_g) = I_{ci}^j(\phi,\gamma_g) = 0$ and

$$I_{ci}^j(\beta,\phi) = -\frac{\partial^2}{\partial\beta\partial\phi}\log P(y_i \mid x_i^j,\beta,\phi).$$

The expressions for $S_{ci}^j(\gamma_g)$ and $I_{ci}^j(\gamma_g,\gamma_g)$ depend on the distribution of genetic covariates. For example, suppose that the jth possible haplotype configuration consistent with the observed SNP genotypes of the ith subject has $x_{gi}^j = (n_1,\ldots,n_{H+1})$, where $n_h = 0,1,2$ codes the number of copies of haplotype h and there are $H+1$ haplotypes in the population. Then, under HWP,

$$\mathsf{P}(x_{gi}^j|\gamma_g) \propto {\gamma_{g1}}^{n_1} \times {\gamma_{g2}}^{n_2} \times \cdots \times {\gamma_{gH}}^{n_H} \times \left(1 - \sum_{h=1}^{H}\gamma_{gh}\right)^{n_{H+1}},$$

where γ_{gh} is the population frequency of haplotype h. The score vector is

$$S_{ci}^j(\gamma_g) = \left(\frac{n_1}{\gamma_{g1}}, \;\cdots\; , \frac{n_H}{\gamma_{gH}}\right)^T - \frac{n_{H+1}}{1-\sum_{h=1}^{H}\gamma_{gh}}\;(1,\cdots,1)^T$$

$$\text{and} \quad I_{ci}^{j}(\gamma_g, \gamma_g) = -\text{diag}\left(\frac{n_1}{\gamma_{g1}^2}, \cdots, \frac{n_H}{\gamma_{gH}^2}\right) - \frac{n_{H+1}}{\left(1 - \sum_{h=1}^{H} \gamma_{gh}\right)^2} J,$$

where J is an $H \times H$ matrix of ones.

Louis (1982) writes the observed information as

$$I(\beta, \phi, \gamma_g) = \text{E}[I_c(\beta, \phi, \gamma_g)|x_{obs}, y] - \text{Var}[S_c(\beta, \phi, \gamma_g)|x_{obs}, y].$$

Under independence of subjects, it follows that

$$I(\beta, \phi, \gamma_g) = \sum_i \mathsf{E}[I_{ci}(\beta, \phi, \gamma_g)|x_{obs,i}, y_i] - \sum_i \mathsf{VAR}[S_{ci}(\beta, \phi, \gamma_g)|x_{obs,i}, y_i]$$

where

$$\mathsf{E}[I_{ci}(\beta, \phi, \gamma_g)|x_{obs,i}, y_i] = \sum_j w_{ij}(\beta, \phi, \gamma_g) I_{ci}^{j}(\beta, \phi, \gamma_g) \tag{1}$$

and

$$\mathsf{VAR}[S_{ci}(\beta, \phi, \gamma_g)|x_{obs,i}, y_i] = \sum_j w_{ij}(\beta, \phi, \gamma_g) S_{ci}^{j}(\beta, \phi, \gamma_g) S_{ci}^{j}(\beta, \phi, \gamma_g)^T - \left\{\sum_k w_{ik}(\beta, \phi, \gamma_g) S_{ci}^{k}(\beta, \phi, \gamma_g)\right\} \left\{\sum_l w_{il}(\beta, \phi, \gamma_g) S_{ci}^{l}(\beta, \phi, \gamma_g)^T\right\}$$

(Lipsitz and Ibrahim 1996; Burkett 2002)

It can be shown that the expected value of $\sum_j w_{ij} I_{ci}^{j}(\beta, \phi)$ is a vector of zeroes. Thus, to estimate the Fisher information, we may replace the $I_{ci}^{j}(\beta, \phi)$ submatrix of $I_{ci}^{j}(\beta, \phi, \gamma_g)$ in equation (1) by a vector of zeros and take $I_{ci}^{j}(\beta, \phi, \gamma_g)$ to be a block-diagonal matrix.

Affiliation:

Kelly Burkett
Department of Statistics and Actuarial Science
Simon Fraser University
8888 University Drive
Burnaby BC, Canada,V5A 1S6
E-mail: kburkett@sfu.ca
URL: http://stat-db.stat.sfu.ca:8080/statgen

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 16, Issue 2 *Submitted:* 2005-10-20
May 2006 *Accepted:* 2006-04-26