Statistical Methods for Mapping Complex Traits

Lorraine Allchin

Lincoln College

University of Oxford

DPhil in Genomic Medicine and Statistics

January 2014

Dedication

This thesis has been a long and interesting project that would not have been possible without the support and encouragement of my long suffering parents, Gil and Sandra Allchin, and boyfriend William Alway. I hope you know how much you mean to me, this is for you.

Acknowledgements

Thanks goes to my supervisors Chris Holmes and Richard Mott for their support and guidance on this project.

Abstract

The first section of this thesis addresses the problem of simultaneously identifying multiple loci that are associated with a trait, using a Bayesian Markov Chain Monte Carlo method. It is applicable to both case/control and quantitative data. I present simulations comparing the methods to standard frequentist methods in human case/control and mouse QTL datasets, and show that in the case/control simulations the standard frequentist method out performs my model for all but the highest effect simulations and that for the mouse QTL simulations my method performs as well as the frequentist method in some cases and worse in others. I also present analysis of real data and simulations applying my method to a simulated epistasis data set.

The next section was inspired by the challenges involved in applying a Markov Chain Monte Carlo method to genetic data. It is an investigation into the performance and benefits of the Matlab parallel computing toolbox, specifically its implementation of the Cuda programing language to Matlab's higher level language. Cuda is a language which allows computational calculations to be carried out on the computer's graphics processing unit (GPU) rather than its central processing unit (CPU). The appeal of this tool box is its ease of use as few code adaptions are needed.

The final project of this thesis was to develop an HMM for reconstructing the founders of sparsely sequenced inbred populations. The motivation here, that whilst sequencing costs are rapidly decreasing, it is still prohibitively expensive to fully sequence a large number of individuals. It was proposed that, for populations descended from a known number of founders, it would be possible to sequence these individuals with a very low coverage, use a hidden Markov model (HMM) to represent the chromosomes as mosaics of the founders, then use these states to impute the missing data. For this I developed a Viterbi algorithm with a transition probability matrix based on recombination rate which changes for each observed state.

Contents

Chapter 1

Introduction

Genetic mapping is the process by which the association between genes and phenotypes is calculated. The first genetic mapping studies used genetic linkage. The earliest published linkage study was in 1913 by Sturtevant [1] who found it was possible to cross Drosophila so that a linear series of six linkage groups (chromosomes) could be constructed.

Linkage studies in laboratory species involve the repeated crossing of subjects with known differing phenotypes. A Mendelian or single gene trait is a phenotype which is associated with, and controlled by, a single gene. It is named after Gregor Mendel who carried out experimental crosses of pea plants in the 1860s [2] [3].

Due to the process of meiosis that occurs during the formation of gamete cells, crossovers or recombination occur between the chromosome pairs. It is possible to infer whether a gene responsible for a complex trait is located near to, or linked to, a Mendelian trait. As recombination is a rare event, the inheritance of a complex trait along with a Mendelian trait implies a close genetic distance between the two causal genes. This, however, requires multiple crosses to ensure the causal genes are located on the same chromosome [4] [48].

A complex trait is a phenotype which is influenced by the interaction of many different genes and the environment. Complex traits may be dichotomous (e.g. disease) or quantitative (e.g. bodyweight). This complication means that in many cases much of the variance remains unexplained and it is difficult to ascertain whether or not all of the causal genes have been identified [2]. These phenotypes can also be

heavily influenced by environmental effects, which further increases the difficulty in identifying the main causal variants.

The analysis and theory of quantitative genetics mainly started to develop in the early part of the twentieth century with the rediscovery of Mendelian genetics, but, what was at the time an independent theory, of continuously varying phenotypes began with Francis Galton [31]. A significant amount of modern statistical analysis can be traced back to his approaches [30], [32]. An important figure in the development of frequentist statistics, Karl Pearson, was inspired by Galtons work although he disagreed with some of the interpretations [30], [33].

Quantitative trait analysis and Mendelian genetics remained mainly separate until the end of the first decade of the twentieth century and the development of multifactor hypothesis. This began with the discovery that inbreeding can lead to a loss of heterozygosity and variation in complex traits and as such these traits would have a Mendelian component [30], [34]. As more research was carried out, traits which were linked to three or more genes were discovered [35]. His development of a three factor model lead to an observation that from a small number of interacting genetic loci can very quickly lead to a large number of genotypes. Given n Mendelian loci interacting there are 3^n different genotypes. [30]

Also around this time, the effect that the environment can have on quantitative traits was first discovered. Early research was carried out in [36], on an inbred bean line. Here it was shown that the mass of the second generation seeds was mostly independent of that of the first generation. It was Johannsen that invented the genotype and phenotype terminology we use today [30] From these beginnings the study of complex traits has developed into a highly technical field. Leading the study of complex traits today is the association study.

In 1980 [5] it was proposed that sequence polymorphisms, micro satellite regions, could be used to track recombination. These are small areas of the genome which repeat at least once. As these were more widespread than Mendelian traits, this allowed a dramatic increase in the power of small linkage studies and important discoveries soon followed:

- The mapping of Huntingdon's disease in 1983 [6]

- Several breast cancer genes [7] [8] [9]
- Alzheimers disease [10] [11]
- Type 1 diabetes in 1994 [12]

Linkage between a genotyped marker and a causal variant is usually characterized by its logarithm of odds (LOD) score. This was developed in 1955 by Morton [13].

$$\text{LOD} = \log_{10} \frac{(1-\theta)^N \theta^R}{0.5(N+R)} \tag{1.1}$$

where θ is the recombination fraction between the causal locus and the genotyped genetic marker, N is the number of non-recombinant individuals and R is the number of recombinant individuals. The LOD score can be thought of as the log of the ratio ratio between the probability of a causal locus being linked to a marker with recombination fraction θ and being unlinked ($\theta = 0.5$). These developments allowed for over a twenty fold increase in the number of Mendelian diseases linked to a gene between the 1980s and 2008 [4] [14].

Nonetheless, linkage studies are difficult to perform on a large scale. In the laboratory they take a significant amount of time and resources in order to carry out the necessary crosses. It is also extremely difficult to extend a large linkage study to a human population as family groups are small and scientists cannot determine the crosses. They are also unsuitable for the genetic dissection of complex traits, and were generally unsuccessful when applied to non-mendelian traits.

Despite their achievements with Mendelian diseases, linkage studies lack the power to detect many complex traits due to their small sample size and limited power and mapping resolution.

An improvement to linkage studies emerged in the 1990s. Here, the idea was developed that one could identify causal genes and variants by comparing the frequency of these variants in the population at large with those with and without a disease [4]. This is the idea behind an association study.

This was not, however, a new development. Association studies had been used as far back as the 1950s, when correlations between blood group antigen and peptic

ulcers were established [15]. To begin with, association studies were carried out using only specific genes which the researcher speculated to be causal.

In the 1990s genome wide association studies were suggested [38] [39]. There has been much success in identifying new variants for many complex traits[40], such as inflammatory bowel disease [41] and for some traits such as asthma [42]. Genetic links for quantitative traits such as lipid levels [43] have also been found. Whilst it is true that population based studies have a much higher power to detect causal variants for complex traits, much care must be taken to avoid simple collection bias. It has been suggested that GWAS studies should provide the first step in looking for associations with complex traits but that secondary analysis should be carried out [24].

Due to the way in which human society has evolved, many variants are shared by individuals with similar ancestry. This is referred to as population structure and this has led to both false positives and negatives in genome wide association (GWAS) studies [45].

Case control studies, in which the population in consideration is sampled to collect approximately equal numbers with and without the specified trait, are generally more powerful, but care must still be taken to account for population structure. Here it is preferable to have the same population structure in both of the data sets, in order that common population structure variations are easily eliminated from the study. This is a method of avoiding selection bias in the data as data split between two populations in this manner would result in non-comparable data sets [46].

Missing heritability is the gap between how much variation there is in a complex trait and how much researchers have been able to identify and attribute to known associated SNPs or other covariates. Part of this gap can be attributed to a current lack of ability to detect rare genetic variants, although technological advances in increasing computing power should help to increase detection rates [20]. Rare events have generally been given as an explanation for the missing heritability still noticeable for complex traits [44]. Environmental effects are also listed as sources of missing heritability. Many other reasons for this have been suggested, such as even rarer variants which are hard to detect with current genotyping technologies. Most GWAS

studies only consider the effects of each variant individually so they lack power when considering gene interactions.

Complex traits can also be affected by epigenetic effects. Epigenetics looks at changes in gene activity that are inheritable but does not change the genome sequence itself. Methylation of the DNA is an example of an epigenetic effect, and has been shown to be associated with complex traits in Arabidopsis [25].

Another branch of study into complex traits is that of epistasis, this is the study into the interaction of genes and there effect on complex traits. Traits with epistatic effects have been found in both plants [17] and animals [21], but it has been ignored to a large extent in human research [22], [23]. The mathematics of this is expanded upon in a later chapter.

As noted earlier, there can be external environmental effects that are associated with complex traits. A study in 2006 showed significant effects behavioural and physiological tests on mice. They noted however that gene interactions have a more frequent and greater effect on complex traits [26].

Thesis Description

The motivation for the first part of this thesis was to develop a Bayesian method capable of considering multi locus associations of disease. To this end I developed a Markov Chain Monte Carlo method using a Metropolis Hastings update step and a Laplace prior.

The next part was inspired by the challenges involved in applying a Markov Chain Monte Carlo method to genetic data. It is an investigation into the performance and benefits of the Matlab parallel computing toolbox, specifically its implementation of the Cuda programming language to Matlab's higher level language. Cuda is a language which allows computational calculations to be carried out on the computer's graphics processing unit (GPU) rather than its central processing unit (CPU). The appeal of this tool box is its ease of use as few code adaptations are needed.

The final project of this thesis was to develop a hidden Markov model (HMM) for reconstructing the founders of sparsely sequenced inbred populations. The motivation being that, whilst sequencing costs are rapidly decreasing, it is still prohibitively

expensive to fully sequence a large number of individuals. We proposed that, for populations descended from a known number of founders, it would be possible to sequence these individuals with a very low coverage, use an HMM to represent the chromosomes as mosaics of the founders, then use these states to impute the missing data. For this I developed a Viterbi algorithm with a transition probability matrix based on recombination rate which changes for each observed state.

Chapter 2

Bayesian Machine Learning Methods for Genome-wide Genetic Association Data

Introduction

Genome wide association studies have become a commonplace method of analysing genotype data over the last few years [54]. For quantitative trait loci (QTL) mapping the first papers discussing the method, such as [56], emerged in the late 1980s. In 2009 the paper [57] collated the results of 118 different GWAS studies on humans and created a database of over fifty six thousand significant associations between SNPs and phenotypes.

Most methods for analysing genetic association data only test each marker in the data (whether they be single nucleotide polymorphisms (SNPs) or more complex variations) one at a time. However it may be better to test multiple SNPs simultaneously. [52] shows that the significance of a SNP can be easier to assess when combined with other SNPs which are associated with the phenotype thus increasing the power of detecting true associations and decreasing the false positive rate due to reduced residual variation.

In this thesis I have developed a method using Bayesian machine learning techniques that will analyse data using multiple markers. My methods are based on Markov Chain Monte Carlo (MCMC) techniques and contain several different ways in which the number of markers in the model and their relative importance to the

response vector can be updated. They can be applied to dichotomous (case-control) and to quantitative traits. I investigated its performance and compared it with other Bayesian and Frequentist methods.

Literature Review

Case-Control Data

In a case control study the binary phenotype y (case (1) control (0)) is often modelled in a logistic framework. Suppose the phenotype depends on the genotypes of a set of SNPs, then the likelihood is given by

$$L(\beta|y,X) = \Pi_{i=1}^{n} \left(\frac{1}{1+\exp X_i\beta}\right)^{y_i} \left(\frac{\exp X_i\beta}{1+\exp X_i\beta}\right)^{1-y_i} \tag{2.1}$$

Where y_i denotes case control status and $X = x_{ij}$ is the (n,p) matrix containing the SNP data, where n is the number of subjects and p is the total number of markers. The elements of the matrix can take the values 0,1 or 2. These correspond to how many copies of the major allele the subject, i, has at a particular locus, j. This coding implicitly assumes the genetic effect of the SNP is additive. There are other ways of coding the markers that can be used to account for dominant and recessive effects. For example, the elements of the matrix might only take the values 0 or 1, depending on if they have at least one copy of the dominate allele. β_j quantifies the effect of SNP j on the phenotype. Large positive values imply an individual carrying the reference allele is more likely to be a case. The model can also be written in terms of its log odds ratio

$$\log\left(\frac{\Pr(y_i=1|x)}{\Pr(y_i=0|x)}\right) = \beta_0 + \Sigma_{j=1}^{p} x_{ij}\beta_j \tag{2.2}$$

where p is the total number of markers and β_0 is the initial value of β_j.

If SNPs are considered one at a time then the effect of each SNP can be evaluated easily in a frequentist framework. However, it is impractical to fit a model with more SNPs than subjects in a frequentist framework.

In addition to logistic regression, another commonly used frequentist method for case control data is the Cochran-Armitage test [58], [59], [60]. Let

$$\bar{X}_{case} = \frac{\Sigma_{i=1}^{n_{case}} \Sigma_{j=1}^{p} X_{ij}}{n_{case}} \tag{2.3}$$

$$\bar{X}_{control} = \frac{\Sigma_{i=n_{case}+1}^{n} \Sigma_{j=1}^{p} X_{ij}}{n - n_{case}} \tag{2.4}$$

then the test statistic for the Cochran-Armitage test can be written as

$$T_{CA} = \frac{\bar{X}_{case} - \bar{X}_{control}}{\mathrm{Var}(\bar{X}_{case} - \bar{X}_{control})^{\frac{1}{2}}} \tag{2.5}$$

Where X is a contingency matrix for a specific marker, n_{case} is the number of cases, n is the sample size and p is the number of different genotypes in evidence at that marker, e.g. it would 3 if the shown genotypes were AA, Aa or aa. The Cochran-Armitage test and logistic regression are asymptotically the same.

Quantitative Traits

If the trait is quantitative then methods such as maximum likelihood [62] have been used. If the trait is normally distributed then the maximum likelihood at SNP j is

$$\text{Log Likelihood}_j = -\frac{1}{2} n \log 2\pi\sigma - \frac{1}{2} \frac{(\Sigma(y_i - \mu) - \beta\eta)^2}{\sigma} \tag{2.6}$$

where y_i is the quantitative phenotype, η is the linear predictor and is given by

$$\eta_i = a(X_{ij} - 1) \tag{2.7}$$

a is the allelic effect of a SNP. β is given by

$$\beta = \Sigma y \Sigma \eta - \frac{n \Sigma y \eta}{(\Sigma \eta)^2} - n(\Sigma \eta)^2 \tag{2.8}$$

μ by

$$\mu = \frac{\Sigma y - \beta \Sigma \eta}{n} \tag{2.9}$$

and σ by

$$\sigma = \frac{\Sigma(y_i - \beta\eta_i - \mu)^2}{n} \tag{2.10}$$

Shrinkage Methods

It can be beneficial to only consider a subset of markers as it can produce a lower prediction error and can be easier to interpret than a model that contains all possible markers [55]. Shrinkage methods are an extension of subset selection methods that work in a more continuous manner and have a lower variability.

Ridge regression is a least squares regression method, (see Chapter 3 for more details on least squares regression), that puts a penalty on the size of the regression coefficients. So if $y = X\beta + \epsilon + \beta_0$ then

$$\beta_{Ridge} = \arg\ \min_\beta \frac{1}{2}\Sigma(y_i - \beta_0(i) - \Sigma X_{ij}\beta_j)^2 + \gamma\Sigma\beta_j^2 \tag{2.11}$$

where β_0 is a constant, ϵ is an error term, X is a matrix of input data and β is a matrix of coefficients. γ controls the rate of shrinkage[55] and $\gamma\Sigma\beta_j^2 = L_2$ is the ridge penalty. Ridge regression has been used in genetic mapping such as in [67].

Another shrinkage method is the Lasso. Here

$$\beta_{Lasso} = \arg\ \min_\beta \Sigma(y_i - \beta_0(i) - \Sigma X_{ij}\beta_j)^2 + \gamma\Sigma|\beta_j| \tag{2.12}$$

where $\Sigma|\beta_j| = L_1$ is the Lasso penalty and again γ controls the shrinkage. The Lasso has been adapted to Bayesian methods for quantitative trait loci in [53]. "Both methods find the first point where"[55] the β contours meet the constraint region. So the Lasso is more likely to remove markers from the model.

Markov Chain Monte-Carlo

MCMC analysis is a type of Byesian inference. This is a branch of mathematics which seeks to fit models to data. In Bayesian inference we define a sampling model, $P(y|\beta)$, and a prior distribution, $P(\beta)$, which reflects any information we have about the parameters of the model before analysing any specific data set. These two probability distributions can be used to determine the posterior distribution, $P(\beta|y)$, using Bayes Rule:

$$P(\beta|y) = \frac{P(y|\beta)P(\beta)}{\int P(y|\beta)P(\beta)\mathrm{d}\beta}. \tag{2.13}$$

The posterior distribution shows how our knowledge about the distribution of the parameters has been changed by the observed data [55].This use of external information can be seen as an advantage of Bayesian inference over frequentist inference (an example of frequentist inference is maximum likelihood estimation), as it leverages this knowledge to produce more informed models. In other ways, however, this need for prior information can be a disadvantage, as the external information could be incorrect or you may not have any prior knowledge of some of the parameters. This could restrict the use of Bayesian methods, though, in this situation, it may be possible to use a uniform distribution for the prior.

There are two main versions of MCMC algorithms, the Metropolis Hastings algorithm and the Gibbs sampler. The Metropolis Hastings algorithm for MCMC was developed in two parts. It was first developed by Metropolis et al 1953, [72], where the algorithm would only work with symmetric jumping distributions. The method was later generalised by W.K.Hastings in 1970, [73]. The algorithm consists of a random walk that uses an acceptance ratio to determine whether each proposed step is accepted or rejected (see the model section for a description of the acceptance ration in relation to my method). A proposed change to the value of a parameter β_j to a new value, β_j^* is accepted with probability α, where

$$\alpha = \frac{P(\beta_j^*|y)/J_t(\beta_j^*|\beta_j)}{P(\beta_j|y)/J_t(\beta_j|\beta_j^*)} \tag{2.14}$$

where J_t is a jumping distribution. In the Metropolis Hastings algorithm, this distribution does not have to be symmetric - an asymmetric distribution can, in fact, increase the rate of convergence [37]. The acceptance and rejection of proposed changes to the parameters over time leads to convergence to a target stationary distribution. The Metropolis Hastings algorithm has been used in the analysis of QTL data in [29].

The Gibbs sampler was developed by S.Geman and D Geman in 1984, [74]. For a vector of parameters $\beta = (\beta_1, ..., \beta_m)$, a Gibbs sampler at time t changes one parameter by drawing a value β_j^t from the conditional distribution

$$P(\beta_j|\beta_j^{t-1}, y) \tag{2.15}$$

where β_j^{t-1} is the vector of all parameters at time $t-1$ without β_j. In this method all proposed values of β_j are accepted, so an advantage of this sampler is that it provides

a good exploration of the parameter space. This prior is limited, however, to cases where it is possible to sample directly form the conditional posterior distribution [37].

Gibbs sampling has been used in case control analysis in [28].

Multiple SNP Mapping Methods

There are some multivariate methods of GWAS analysis such as [63]. Although forward selection does not have the best of reputations in genetics, it has been put forward as a good model selection algorithm for additive QTLs [27]. There are also many different ways in which Bayesian methods have been applied to analyse GWAS data, such as the software packages SNPTEST and BIMBAM [69], which fits a Bayesian regression model to GWAS data. BIMBAM uses a standard linear regression of the form

$$y_i = \mu - \Sigma_j \mathrm{D}_{ij}\beta_j + \epsilon_i \tag{2.16}$$

where y is the phenotype vector, D is the design matrix, which has two columns for each SNP; one for additive effects and one for dominant effects. μ is the phenotype mean, β_j denotes a pair of regression coefficients, one for the SNP additive effect and one for the dominance effect and ϵ_i is the residual. Priors on the mean and variance of the phenotype (either Jeffreys' or gamma and normal priors), a prior on how many of the SNPs in the data set are quantitative trait nucleotides, a normal prior on the effect sizes of the qtns. This method then uses Bayes factors to denote significance. Although in the paper this method is used on small sections of the genome it can be applied to the whole genome.

For GWAS studies, however, Bayesian methods using Markov Chain Monte Carlo (MCMC) have not yet been widely used. In [71] a binary probit Model using posterior distributions and singular value decomposition of the design matrix is fitted using a Gibbs sampler.

[75] fits multiple SNPs and uses a reversible jump MCMC with a Poisson prior to find the best model. [76] developed a correction algorithm for multiple comparisons for case control multi locus association studies. They use a MCMC sampler to estimate type 1 error that gives a good correspondence with values calculated by an analytical method.

MCMC has also been used to identify population structure in GWAS data sets such as in [77]. Here they use a uniform prior but also show that a Poisson prior would give almost identical results.

[78] uses MCMC multi QTL mapping methods. Here they use a flat prior on variance components. They found that, whilst this method is not as good at identifying single QTLs, it is much better at separating finely spaced QTLs.

In [79] the authors developed a Bayesian Lasso which uses similar MCMC methods and priors as in this project for the related problem of association between genotypes and phenotypes, and the effect that covariates can have on that relationship. The Bayesian Lasso method works by creating a "preconditioned response variable using supervised principle component analysis"[79]. It then uses an MCMC step on a subset of SNPs with a double exponential prior on genetic effects.The linear model is

$$y = X\beta + \epsilon \tag{2.17}$$

where epsilon is the residual error $\epsilon \sim N_n(0, \sigma^2 I_n)$. This is preconditioned by removing SNPs with low minor allele frequencies and repeated rows from the genotype matrix. The penalized regression also takes account of discrete and continuous covariates. The linear regression is of the form

$$\tilde{y_i} = \mu + D_i' d + C_i' c + F_i' f + G_i' g + \epsilon_i \tag{2.18}$$

where μ is the mean, D is the vector of discrete covariates with d the corresponding regression coefficient vector, similarly C and c for the continuous covariate, f is the vector of additive effects with F it's indicator vector and similarly g and G for the dominant effects. They then use a Gibbs step to refine parameter choice.

[70] is a thorough review paper of several different types of Bayesian variable selection. They describe these methods as a way to "estimate the posterior probability of all models within the considered class of models" [70]. They describe variable selection methods as fitting into four categories; indicator model selection, stochastic search variable selection, adaptive shrinkage and model space approach. The method presented in this thesis uses metropolis Hastings MCMC and would fit into the model space approach, whereas BIMBAM with its use of Jeffrey's priors would fit into

adaptive shrinkage. They summarise the performance of various models in these categories in terms of their speed, mixing and separation. The adaptive shrinkage models they look at (Jeffrey's and Laplacian) both have average speed, Jeffrey's has excellent mixing and good separation whilst the Laplacian method has poor mixing and poor separation. They class reversible jump model selection as having fast speed, good separation but mixed performance on mixing. They ran all of the models on three different data sets, and each model performed differently on the varying data. In the summary they concluded that a single Bayesian variable selection method is unlikely to be optimal for all types of data.

The Model

Case Control Data

In this thesis I first investigate a Bayesian version of the logistic model for multiple SNPs where the prior probability that a marker is in the model (that its β parameter is non zero) is given by

$$\begin{aligned} \text{Prior}(\beta_j) &= 0 \text{ with probability } \pi && (2.19) \\ &\quad \frac{1}{2\lambda}\exp(-\frac{|x|}{\lambda}) \text{ with probability } 1-\pi && (2.20) \end{aligned}$$

where π is the prior probability that a marker is in the model and λ is the prior expectation of a β given that it is in the model. Here (2.18) is the probability density function of a Laplace distribution with parameters $(0, \lambda)$. This is a sparsity inducing prior parametrised by π and λ. This means that it is a prior which should lead to few markers being added to the model. Thus the prior distribution of the number of markers in the model is Bin(p,π). Also for comparisons, I used a version of the model with a Normal $(0, \sigma)$ prior on β instead. This is so I can evaluate the performance of the Laplace prior when compared to a more commonly used prior.

A Laplace distribution was chosen for the prior as it has the following features; it penalises the absolute value of the markers, it has a large effect on small values and it has fatter tails than the Normal prior (see Figure 2.1 for comparison).

The likelihoods of the priors cannot be factorised, so an MCMC method is used. I developed two MCMC algorithms; (i) a single update version which updates the

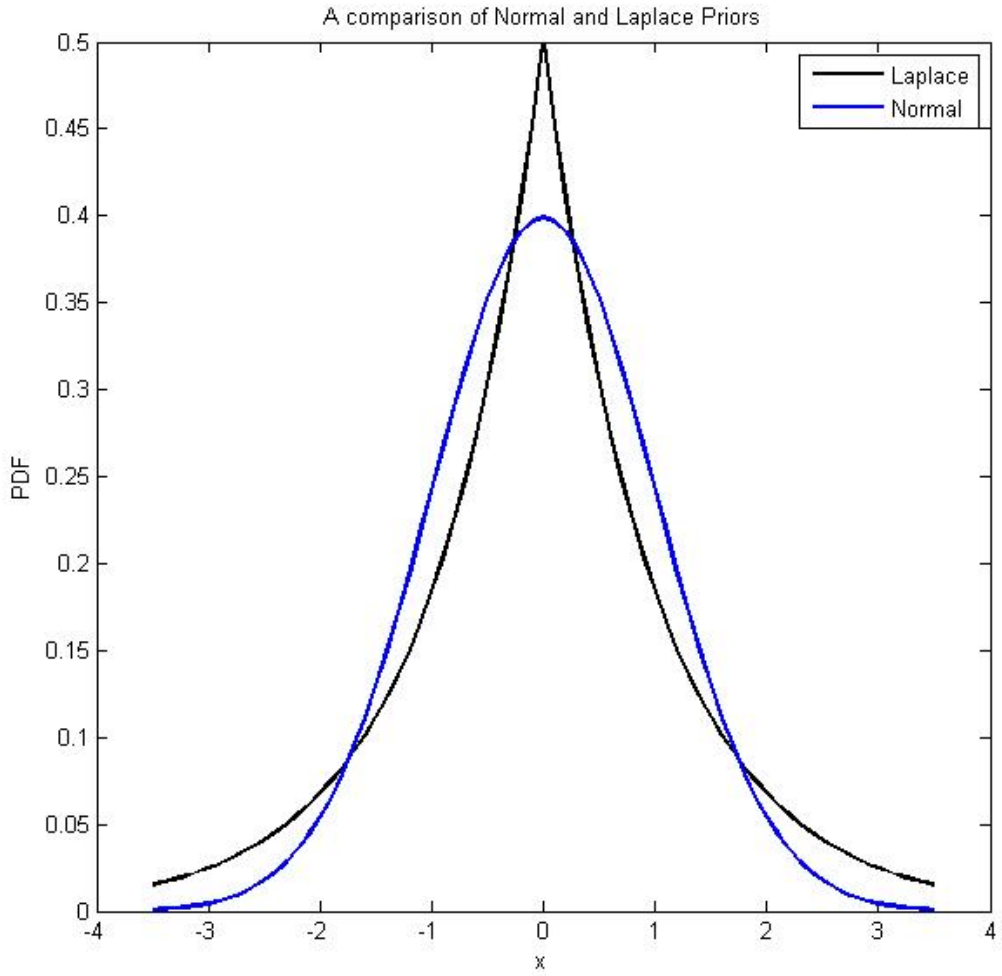

Figure 2.1: A plot comparing the Laplace and Normal distributions

β's one at a time and (ii) a multi update version which optionally updates a marker and one of its immediate neighbours at the same time. Both of these algorithms have Laplace and Normal prior versions. These algorithms use four and five different ways of updating the values of β_j respectively; a flow chart of the multi update method is shown in figure 2.2. The algorithms sweep through all of the markers in the data in genome order. At each marker j the following steps are applied. Firstly, in the multi update version, a random choice between single and bivariate marker updating is made. The single marker update is chosen with a probability of a in the multi update algorithm, the single update model can be considered as having $a = 1$. If the

single marker update is chosen, one of the four routes will then be chosen.

If the current value of $\beta_j = 0$ we propose to add the marker into the model. The new proposed value for β_j is drawn from a N(0,1). If the current $\beta_j \neq 0$ then with probability of b a random draw from a N(0,1) is added to the current value of β_j to produce a new proposed value for β_j.

If $\beta_j \neq 0$ then with probability c we propose to remove marker j from the model by setting the proposed value of β_j to 0. Finally if the current $\beta_j \neq 0$ with probability d we swap marker j out of the model (set $\beta_j = 0$) and instead add in marker k to the model where k is a marker within a 10 marker window around j. This updating step was added to the algorithm to make some allowance for linkage disequilibrium and also to help reduce the likelihood becoming stuck in a local maximum.

If the algorithm takes the bivariate path, there is a random draw from a bivariate normal, this is then added to the value of β_j and one of its neighbours $\beta_{j\pm1}$) (chosen randomly), to form the proposed value of β_j. The probabilities (a, b, c, d) for choosing each update method and the length of the swap window were established by carrying out simulations with varying values to try and optimise the performance of the model.

The updating of the marker value uses the Metropolis Hastings algorithm, which allows for non-symmetric sampling distributions such as occur in this algorithm. The updated value of β_j is accepted with probability α where $\alpha = min(1, R)$, R is calculated by evaluating

$$\begin{aligned} \log\ \mathrm{R} &= A + B + C + D;\ \text{where} && (2.21)\\ A &= \log\ \text{prior distribution ratio} && (2.22)\\ B &= \log\ \text{likelihood ratio} && (2.23)\\ C &= \log\ \text{proposal ratio} && (2.24)\\ D &= \log\ \text{SNP prior ratio} && (2.25) \end{aligned}$$

The contribution of the prior to this ratio consists of two parts A and D. A is calculated using (2.17,2.18) for the current and proposed models. D (the SNP prior ratio) accounts for the affect of changing the number of markers in the model. It is calculated using the following,

$$D = (p^* - p')\log\pi + (p' - p^*)\log(1 - \pi) \qquad (2.26)$$

where p' is the current number of markers in the model and p^* is the proposed number of markers in the model . The log likelihood ratio, B, is calculated by evaluating the likelihood (2.1) for both the proposed and current models. If $\beta^* \sim q(\cdot)$ where β^* is the proposed value for β and q is the jumping distribution, then the proposal ratio, C, is given by

$$\frac{q(\beta|\beta^*)}{q(\beta^*|\beta)}. \tag{2.27}$$

as in this update scheme we let $\beta(j)_{i+1} = \beta(j)_i + w$, $w \sim$N(0,1). Let the probability of a specific $w = R$ the jumping distributions of note are

$$\text{Birth move: } q(\beta|\beta^*) = \text{P(birth move)} * R \tag{2.28}$$

$$\text{Death move: } q(\beta|\beta^*) = 1 * \text{P(death move)}. \tag{2.29}$$

The swap and general update step are symmetrical update steps, so they result in a proposal ratio of 1.

Quantitative Data

The above method is for case control. However quantitative trait data sets need a different approach within the update scheme. The data can be assumed to be normally distributed with unknown variance σ^2 so the likelihood becomes

$$L(\beta, \sigma|y, X) = \Pi_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{(y_i - x_i\beta_i)^2}{2\sigma^2} \tag{2.30}$$

Firstly we need to approximate the distribution from which the phenotypic values have been drawn. We assume that it is normally distributed and we use a Gibbs step to estimate the variance. The resampling of the variance occurs in every iteration of the code and leads to each β currently in the model being resampled with every iteration and so the bivariate update option is omitted. This re sample is calculated as follows

$$
\begin{aligned}
a_1 &= a_0 + \frac{n}{2} && (2.31)\\
\mathrm{V}_1 &= (\mathrm{X}'_\beta \mathrm{X}_\beta + \mathrm{V}^{-1})^{-1} && (2.32)\\
m_1 &= \mathrm{V}_1(X_\beta y + \mathrm{V}_1^{-}1m_0) && (2.33)\\
b_1 &= b_0 + \frac{1}{2}(y'y + m'_0\mathrm{V}_0^{-1}m_0 - m'_1\mathrm{V}_1^{-1}m_1) && (2.34)\\
\sigma &\sim \mathrm{IG}(a_1, b_1) && (2.35)\\
\beta &\sim \mathrm{N}(m_1, V_1) && (2.36)
\end{aligned}
$$

where a_0 and b_0 are constants, X_β is the genotype matrix containing the information for the markers in the model only, V_0 is an identity matrix of size q where q s the number of markers in the model, m_0 is a vector of ones of length q and IG stands for Inverse Gamma distribution. This step is carried out for both Normal and Laplace priors. Another Gibbs step must be carried out when using a Laplace prior. In this case each iteration we redraw λ using the following distribution

$$\lambda \sim \mathrm{IG}(1 + \mathrm{q}, \Sigma_{i=1}^{p}\beta_i) \tag{2.37}$$

The other change is to the log likelihood calculation. Here it is calculated by

$$\log L(\beta, \sigma|y, X) = \frac{n}{2}\log(\sigma) - \frac{1}{2}\Sigma_{i=1}^{n}\frac{(y_i - X_i\beta_i)^2}{\sigma^2} \tag{2.38}$$

Where y is the vector of phenotypic values, derived from (28). The SNP prior and proposal ratios are calculated using the methods described previously.

Method Evaluation

Case/Control Simulations

For the case/control simulations, I used a bootstrapping method with replacement to simulate 1000 cases and 1000 controls for both single and multiple causal SNPs; based on a colon cancer data set for chromosome 18q [68]. This data contains information from 1859 individuals (928 cases, 931 controls) for 183 SNPs. The data I chose for the case control analysis looks at a region of chromosome 18q21 of the human genome. It was created to find loci which increased the risk of colorectal cancer [19]. The original

paper found a signal in this region that contributed to a higher risk for rectal cancer than colon cancer but was significant for both diseases. This data set was chosen as in a previous analysis, carried out by members of my research group, a good signal of association was found [18]. The selection of the cases and controls in the simulation set are carried out using a logistic model where the causal parameter, γ is varied to generate different probabilities of a randomly chosen subject being assigned either case or control status in the simulations.

Linkage disequilibrium (LD) is a measure of the frequency at which alleles for different SNPs occur together. In this thesis I have used the R squared definition of LD which is

$$R^2(f_i, f_j, f_{ij}) = \frac{(f_{ij} - f_i f_j)^2}{f_i(1-f_i)f_j(1-f_j)} \tag{2.39}$$

where f_i is the frequency of allele i and f_{ij} is the frequency of individuals that have allele i at a one locus and allele j at another [65]. A plot of the LD for this data set can be seen in Figure 2.3. Here we can see that the SNPs in this data set were purposely chosen as there is very little LD in the data set. This means that the chosen SNPs do not regularly occur together in the population.

Single Causal SNP Simulations

Here the simulated data are run through the MCMC algorithms under two different priors, (i) a Laplace distribution with λ set so that the two distributions had the same interquartile range (with $\pi = \frac{1}{p}$) and (ii) a normal distribution of mean 0 and variance 10, which is a commonly used prior. The simulations were also run through both the single and multi update algorithms. Each simulation took around two minutes to complete.

Figure 2.4 shows the results for simulating a single causal marker. Here we can see that for the single update option both the Laplace and normal priors perform similarly and they perform slightly worse than the frequentist GLM method (This uses Matlab's inbuilt generalised linear model regression function based on the logistic likelihood with only one SNP in the model. I then rank the SNPs based on their p values). In both multi update cases the MCMC method performs similarly to its single

update counterpart and so it performs slightly worse then the frequentist method at all values of beta. This lowered performance for the MCMC method could be because more iterations are required in the simulations (here the simulations run through 2000 sweeps across the SNPs so a direct comparison between the QTL and CC methods could be made). However this maybe too small of a number for the case control version of the code to converge as can be seen in figure 2.5 where the code is allowed to iterate for 10000 sweeps. In this figure there is a small yet visible improvement of the performance of my MCMC method but it is still outperformed by the frequentist method. The rest of these simulations were carried out with the smaller number of sweeps (2000) to allow for a direct comparison with the quantitative trait simulations.

A brief note about burn in. Burn in for an MCMC method refers to the amount of time/iterations for the algorithm to converge to a steady state. In this model, as a consequence of the number of proposed updates of the β parameters per sweep (at least the number of SNPs in the data set, more if the multi update option is enabled) and the scaling applied to the model, which keeps the total number of markers in the model at any one time to be low, the burn in period is a few hundred sweeps.

Two Causal SNP Simulations

For these simulations the data are simulated with two causal SNPs instead of one. The results are contained in Figure 2.6. Here the frequentist method outperforms the MCMC by a significant margin for all data points. As previously the two priors perform similarly well and the two update schemes are comparable.

Bivariate Update Scheme Changes

At the moment the multi update scheme updates a marker and an immediate neighbour regardless of whether the marker is currently in the model or not. Possible changes are then to update a marker and one chosen at random, only consider the multi update when the marker is already in the model or a combination of both. There is a small improvement for just allowing the multi update step on markers previously added to the model and small dip for random second marker update, whilst the combination of two update steps performs similarly. See figure 2.7 for the simulation results for these update scheme changes.

Update Probability Changes

To investigate the multi update option I varied some of the update choice probabilities to see if that could improve the result, these can be seen in figure 2.8. As shown, a high probability (probability c) for removing a marker from the model (death rate) and low probability (probability d) of choosing the swap step (swap rate) offers a small improvement on the standard updating probabilities, a high swap and low death causes a dip in performance, as does a high bivariate update probability, a. A low bivariate update probability, offers a small level of improvement.

Iteration order

To see what effect the order of updating has, I created a version of the MCMC which permutes the updating order for each sweep. It does this by using the Matlab command randperm(n) where n is the number of markers. The results of this, for both priors, are shown in figure 2.9. This has lead to an improvement in the performance of the method especially at higher values of the bootstrapping parameter.

Dominant effects

As previously mentioned, another way that data can be encoded is, if the SNP has a dominate effect, to have the model matrix consist of 0's and 1's instead. The results for this can be seen in figure 2.10 which use the single update Normal and Laplace prior respectively. For both priors this offers an improvement in performance over heterozygous effects, but the frequentist method is still more effective.

Forward Selection

After this evaluation of the method, I chose to compare my method with a frequentist method that looks at all multiple markers. For this I chose to use Matlab's inbuilt sequential forward selection method. Figure 2.11 shows a comparison between forward selection and a single marker Laplace prior version of my MCMC method. Here you can see a much more favourable result for my MCMC method, at the majority of points it outperforms the sequential forward selection method.

Quantitative Trait Loci Simulations

Here I used genotype data from the outbred mice crlCFW-ALP [99]. This data was chosen as in previous research [16] a QTL has been detected in this region of chromosome 4. This QTL affects the serum level of alkaline phosphatase. This chromosome also contains the gene Akp2 "which encodes the major serum ap isozyme" [16]. The strain was chosen from one of several different outbred strains, so as such is diplotype data, available from [99] which had phenotype data for this region. This is a data set with 71 SNPs and 199 unrelated animals, from which I then simulated the phenotypic value y_i of mouse i as

$$y(i) = a(X(i,j) - 1) + \epsilon \tag{2.40}$$

where i is the subject, j is the causal SNP, ϵ is a random draw from N(0,v) and a is $\sqrt{(}2vn/\Sigma_m X(m,j))$ chosen such that the heritability of the simulated trait was a^2. An LD plot can be seen in Figure 2.12. In this data set we can see that there are two main blocks. The white areas on the plot are caused by SNPs which are the same in all animals.

Normal prior simulations

This first set of simulations are based around four different causal SNPs, 25, 37, 46, 60. The four SNPs where chosen to have different allele frequencies and different linkage profiles. Figure 2.13 shows the minor allele frequency for the data set. The results of these simulations can be seen in figure 2.14. Each simulation took approximately 3 minutes to complete.

However from the results for the causal SNPs we can see that when the minor allele frequency is high my MCMC method outperforms the frequentist but in areas of low minor allele frequency the frequentist performs better. It is not as simple as a direct correlation between allele frequency and performance of my MCMC method. In 2.14it can be seen that the MCMC method produces a better result for simulations based on marker 37 than it does for marker 25 but they both have very low minor allele frequency.

Laplace prior simulations

Next I ran simulations using the Laplace prior. In the first simulations one value of λ was used for all SNPs i.e. the same prior on all SNPs. See figure 2.15 for the results of using one lambda value for all data. Here we can see that for large values of the variance, my MCMC method performs just as well as the frequentist maximum likelihood method, but as the variance falls, the performance of the MCMC falls away from that of the frequentist.

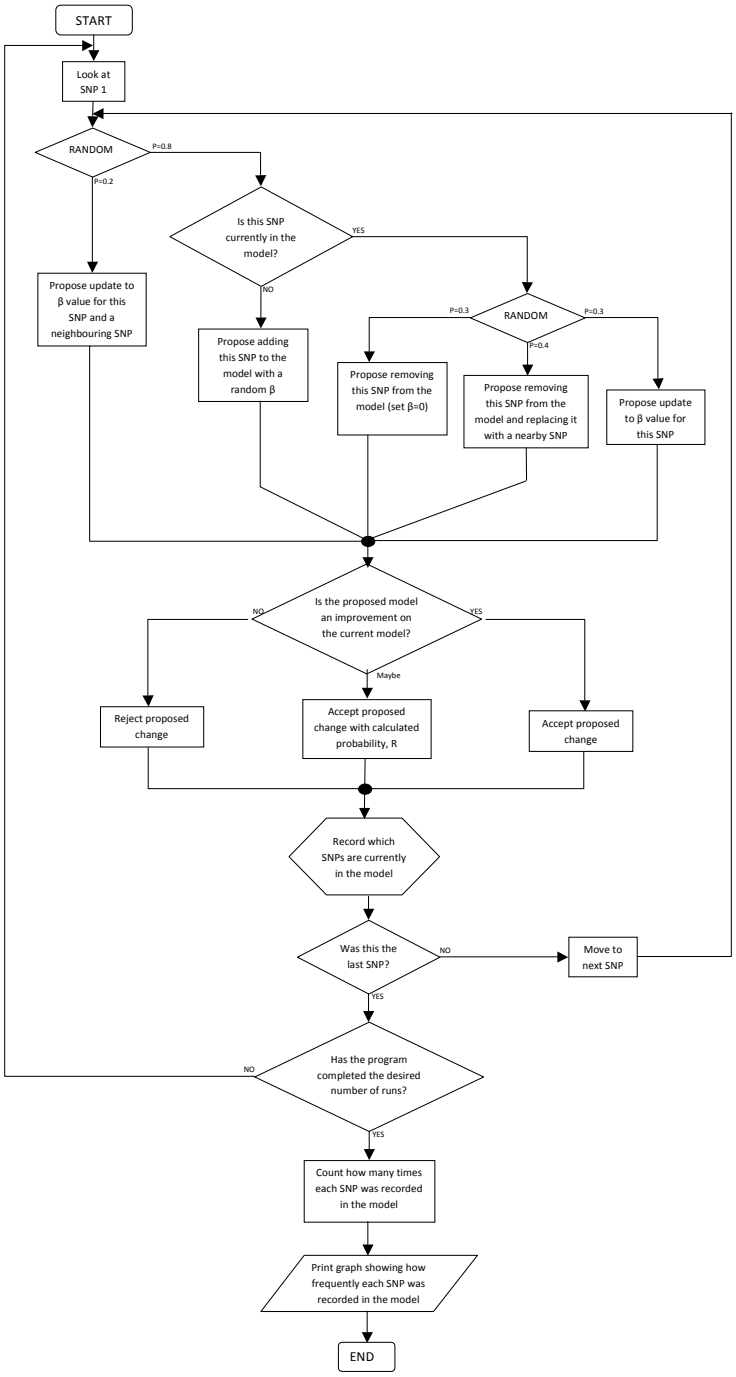

Figure 2.2: Flow Chart

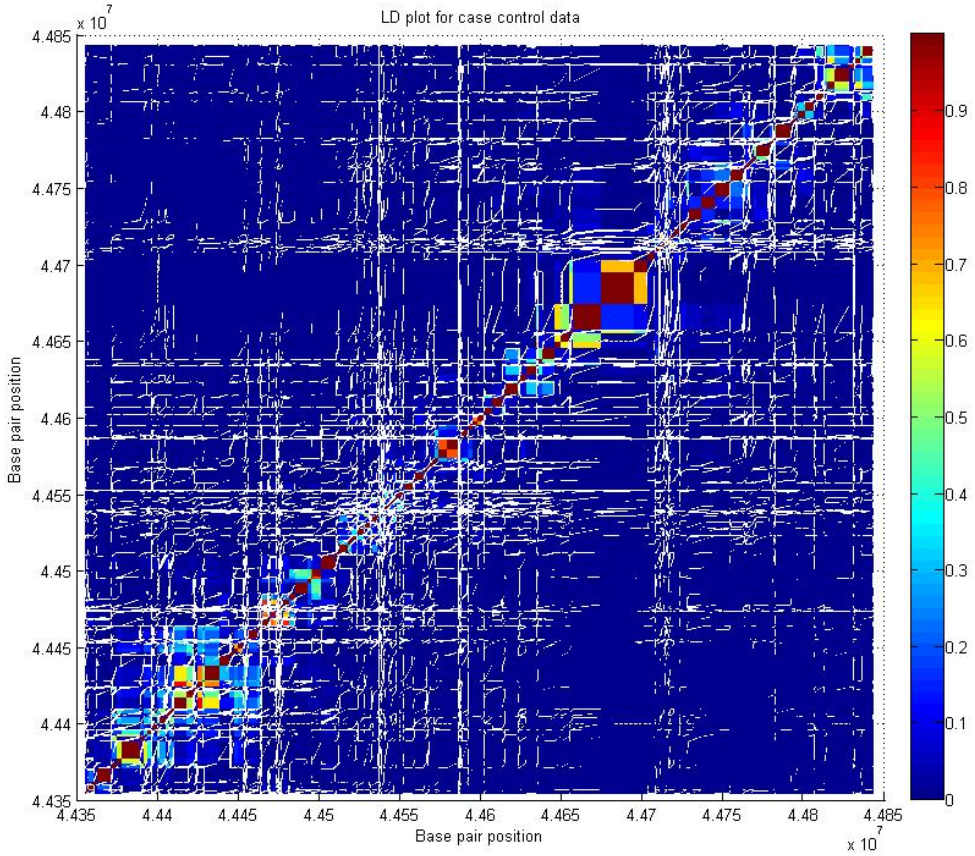

Figure 2.3: A plot showing the Linkage disequilibrium for the case control data

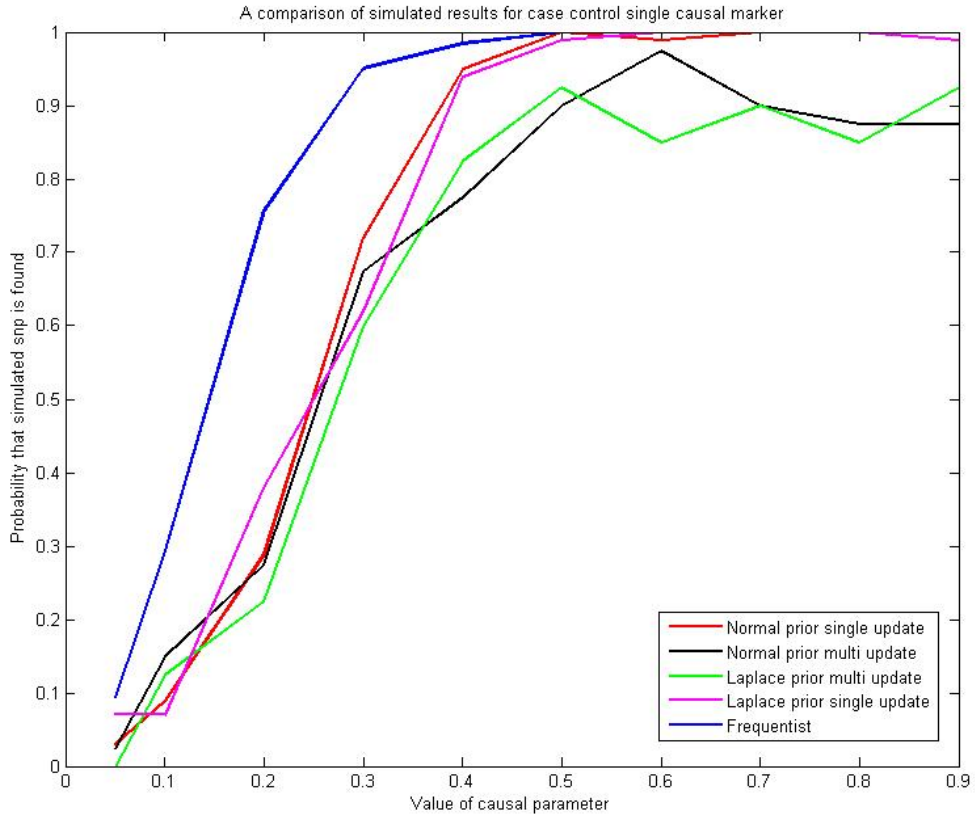

Figure 2.4: Simulation results for case control data single simulated causal SNP. The x axis indicates the value assigned to the causal β parameter. The y axis is the probability that the simulated causal SNP is one of the 10 markers most often included in the model. The blue line indicates the performance of the frequentist GLM method and the other lines the different combinations of update schemes and priors

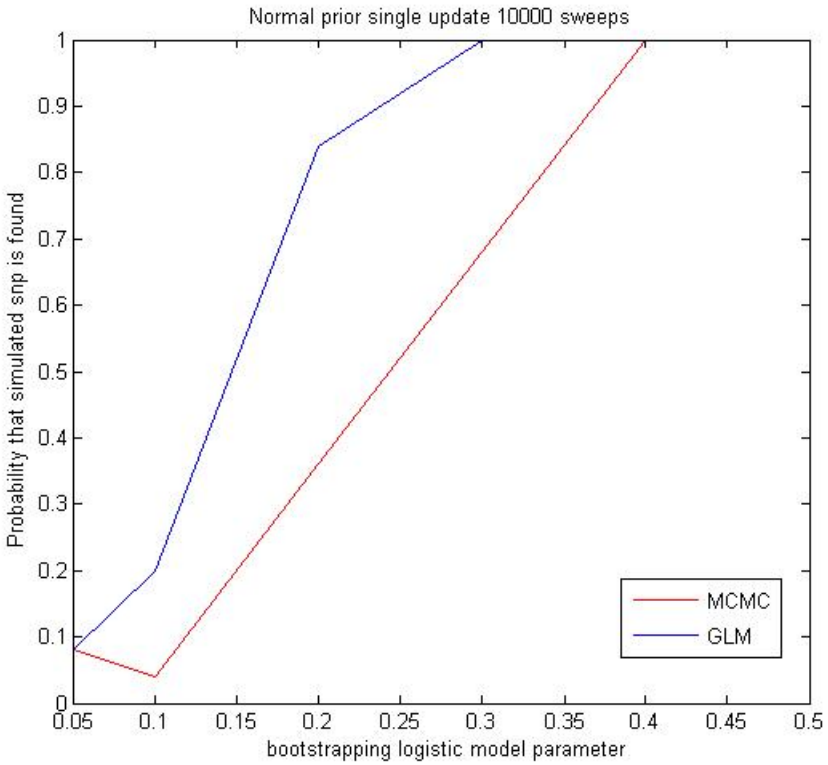

Figure 2.5: Simulation results for case control data with one simulated causal SNP , normal prior, single update, with number of sweeps increased 5 times to 10,000. he x axis indicates the value assigned to the causal β parameter. The y axis is the probability that the simulated causal SNP is one of the 10 markers most often included in the model. The red line indicates the performance of the MCMC model whereas the blue line is the performance of the frequentist GLM method

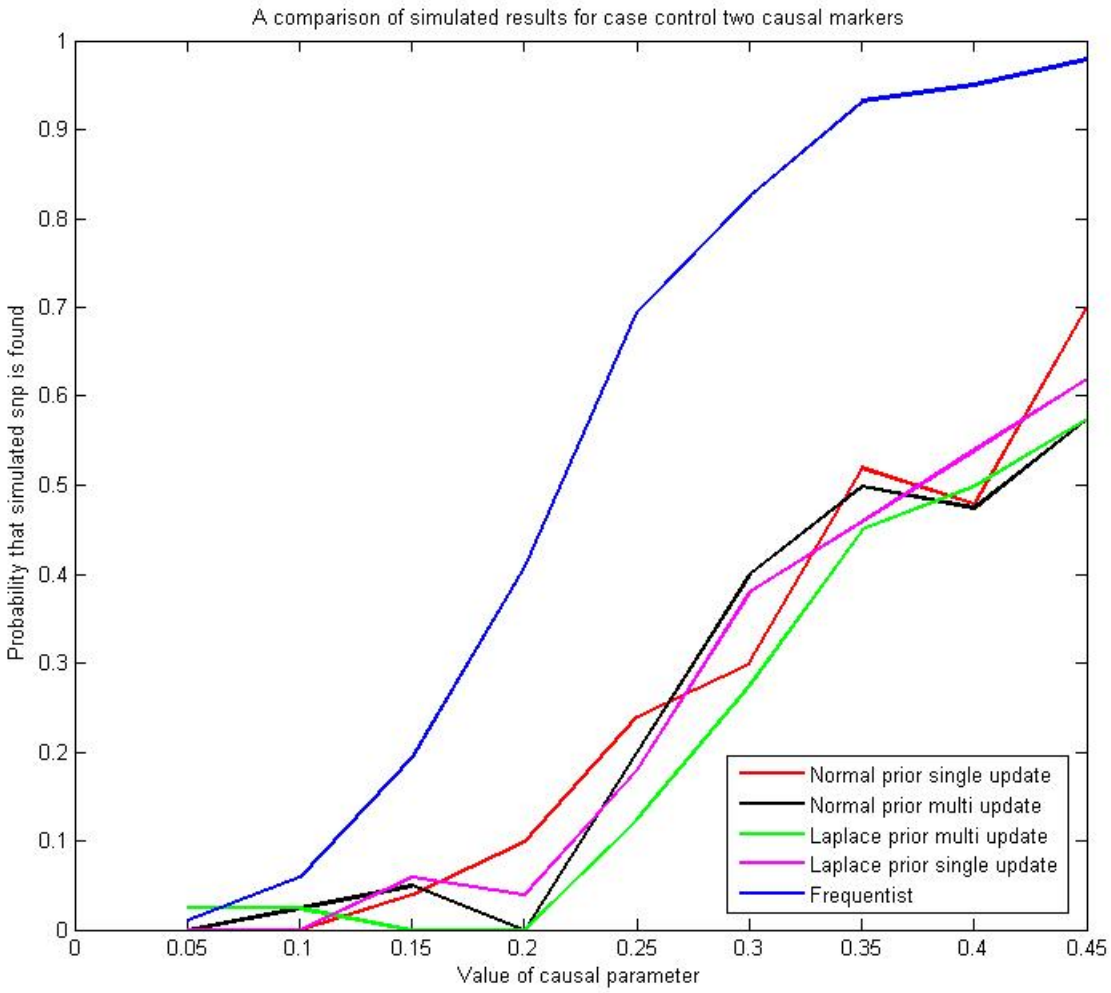

Figure 2.6: Simulation results for case control data two simulated causal SNPs. The x axis indicates the value assigned to the causal β parameter. The y axis is the probability that the simulated causal SNP is one of the 10 markers most often included in the model. The blue line is the performance of the frequentist GLM method and the others the various combinations of update schemes and priors.

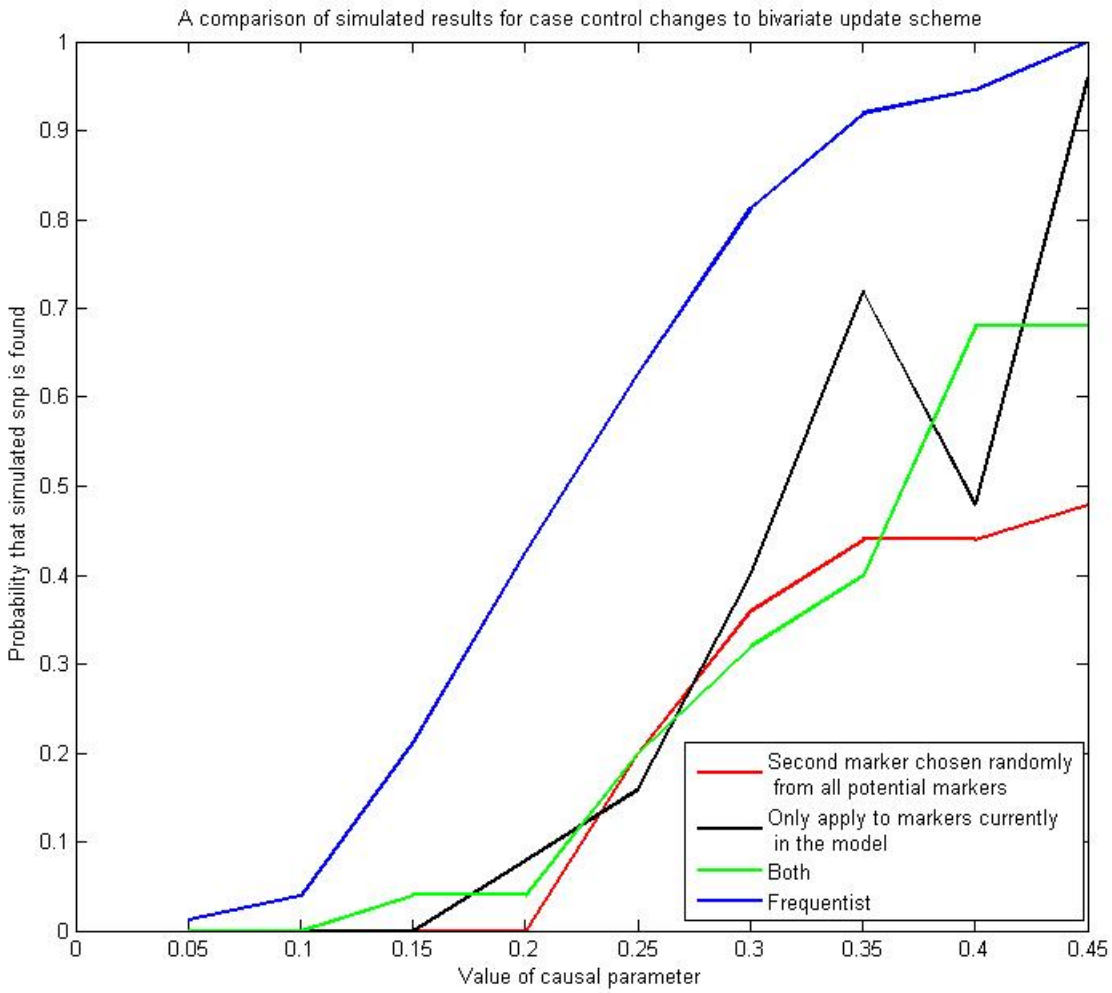

Figure 2.7: Simulation results for case control data two simulated causal SNPs with the multi update changed to only applied to makers currently in the model and updating a second marker at random not necessarily and immediate neighbour. The x axis indicates the value assigned to the causal β parameter. The y axis is the probability that the simulated causal SNP is one of the 10 markers most often included in the model. The blue line is the performance of the frequentist GLM method where as the others show results of changes to the bivariate updating scheme as noted. Normal prior multi update

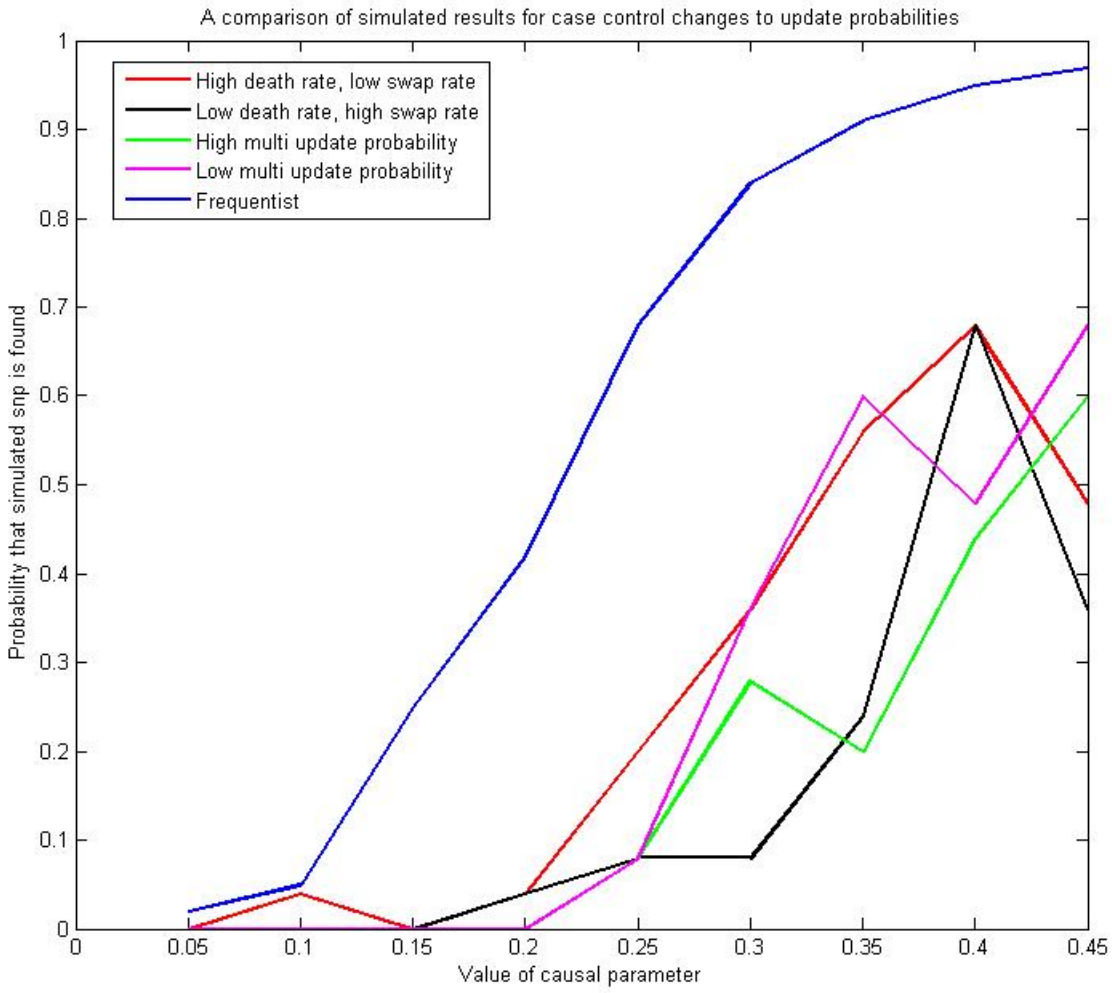

Figure 2.8: Simulation results for case control data two simulated causal SNPs with various changes to update probabilities, as indicated in the legend. The x axis indicates the value assigned to the causal β parameter. The y axis is the probability that the simulated causal SNP is one of the 10 markers most often included in the model. The blue line is the performance of the frequentist GLM method and the others the MCMC with the different update probabilities. Normal prior multi update

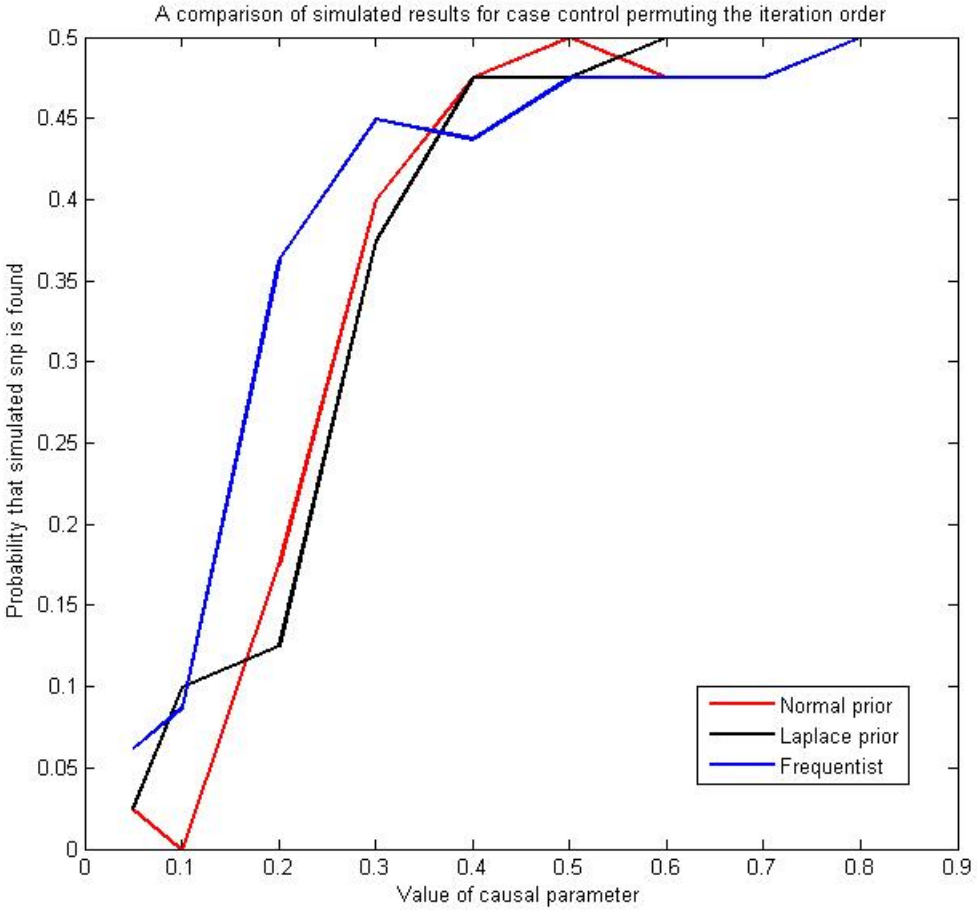

Figure 2.9: Simulation results for permuting the update order of the SNPs. The x axis indicates the value assigned to the causal β parameter. The y axis is the probability that the simulated causal SNP is one of the 10 markers most often included in the model. The red line indicates the performance of the MCMC model with a normal prior, the black line the MCMC model with a Laplace prior and the blue line is the performance of the frequentist GLM method. Single update scheme

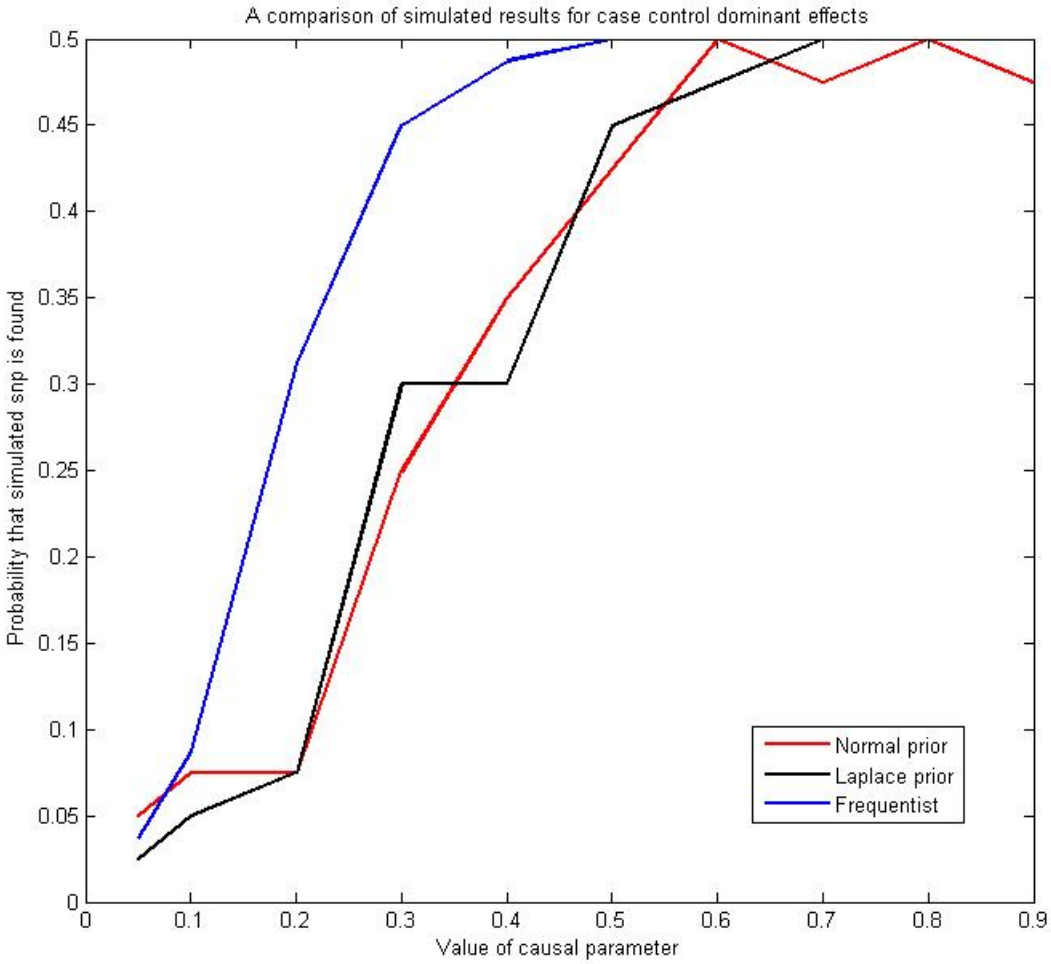

Figure 2.10: Simulation results for a single marker when the data set has been re evaluated to indicate a dominant trait. The x axis indicates the value assigned to the causal β parameter. The y axis is the probability that the simulated causal SNP is one of the 10 markers most often included in the model. The red line indicates the performance of the MCMC model with a normal prior, the black the MCMC model with a Laplace prior and the blue line is the performance of the frequentist GLM method. Single update scheme

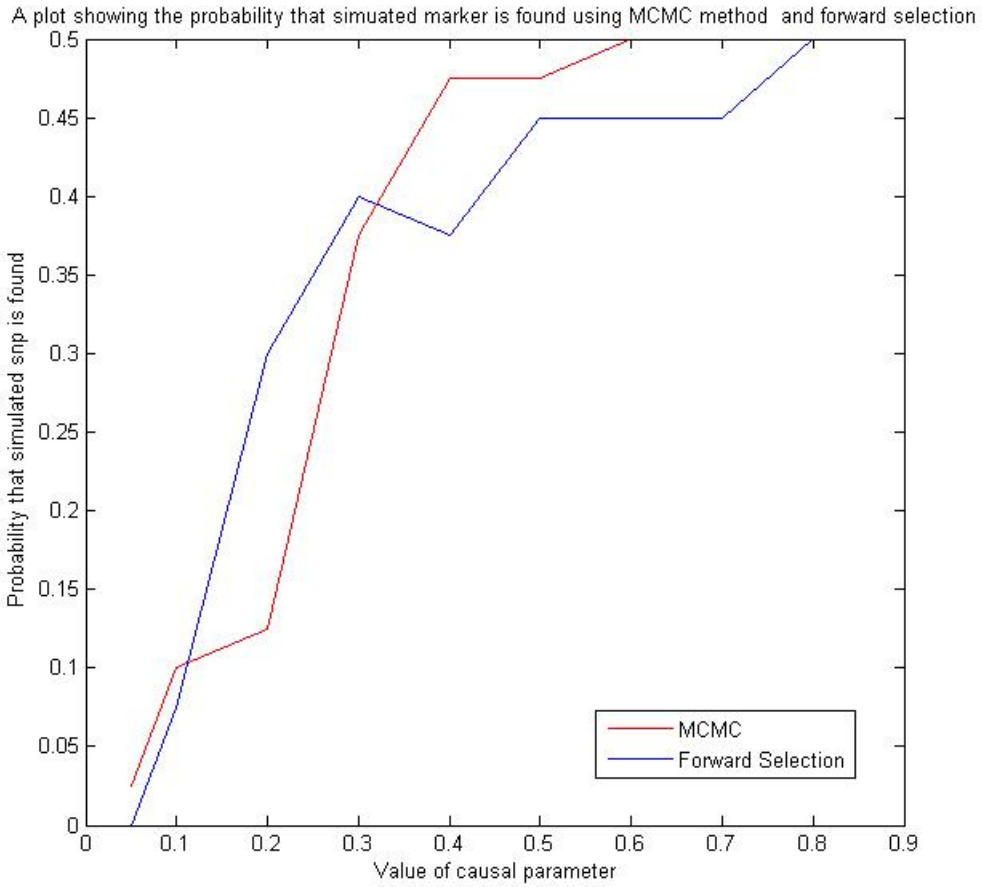

Figure 2.11: Simulation results for a single marker against a forward selection method. The x axis indicates the value assigned to the causal β parameter. The y axis is the probability that the simulated causal SNP is one of the 10 markers most often included in the model. The red line indicates the performance of the MCMC model whereas the blue line is the performance of the forward selection method. Normal prior single update

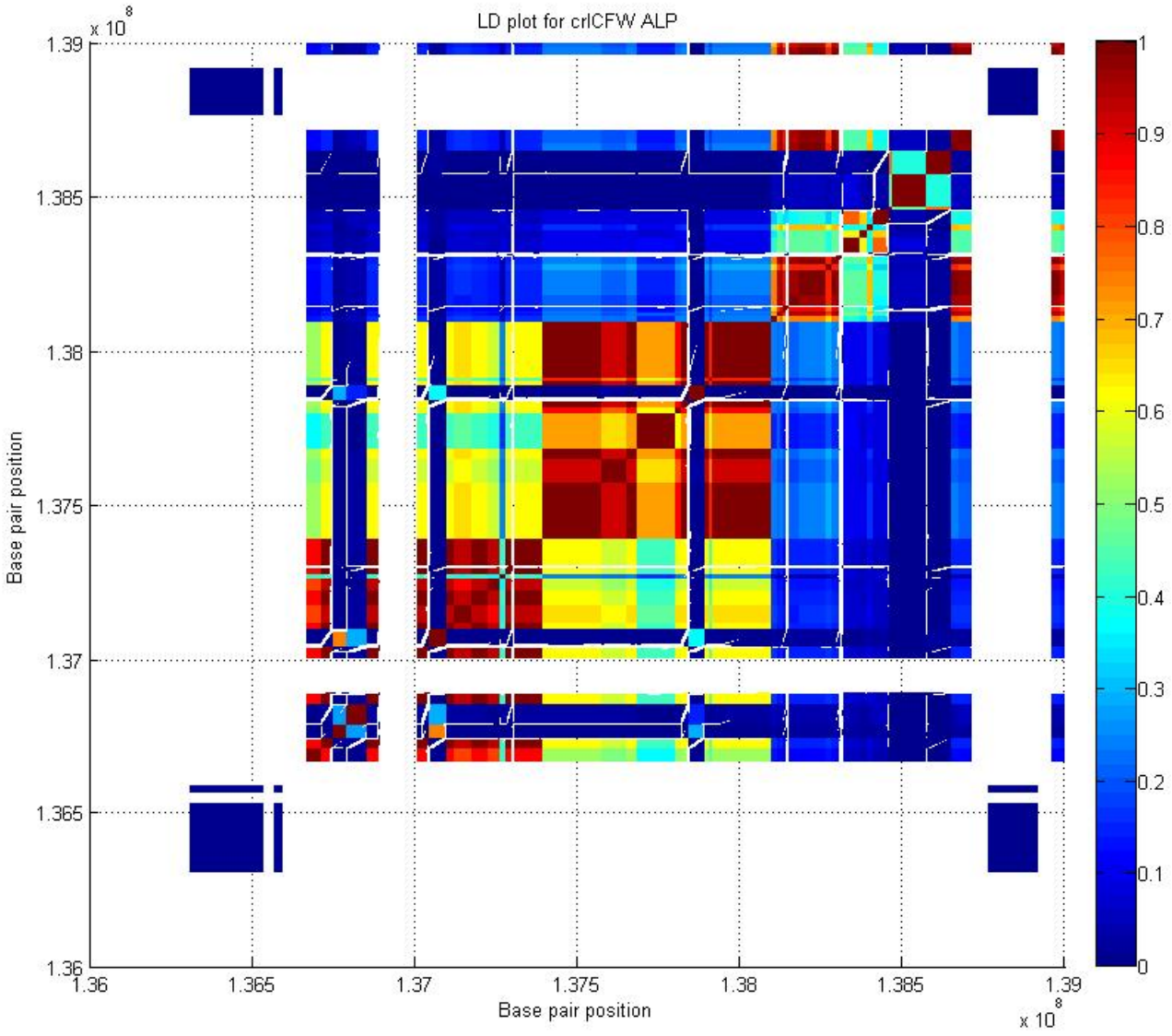

Figure 2.12: A plot showing the Linkage disequilibrium for the quantitative trait data

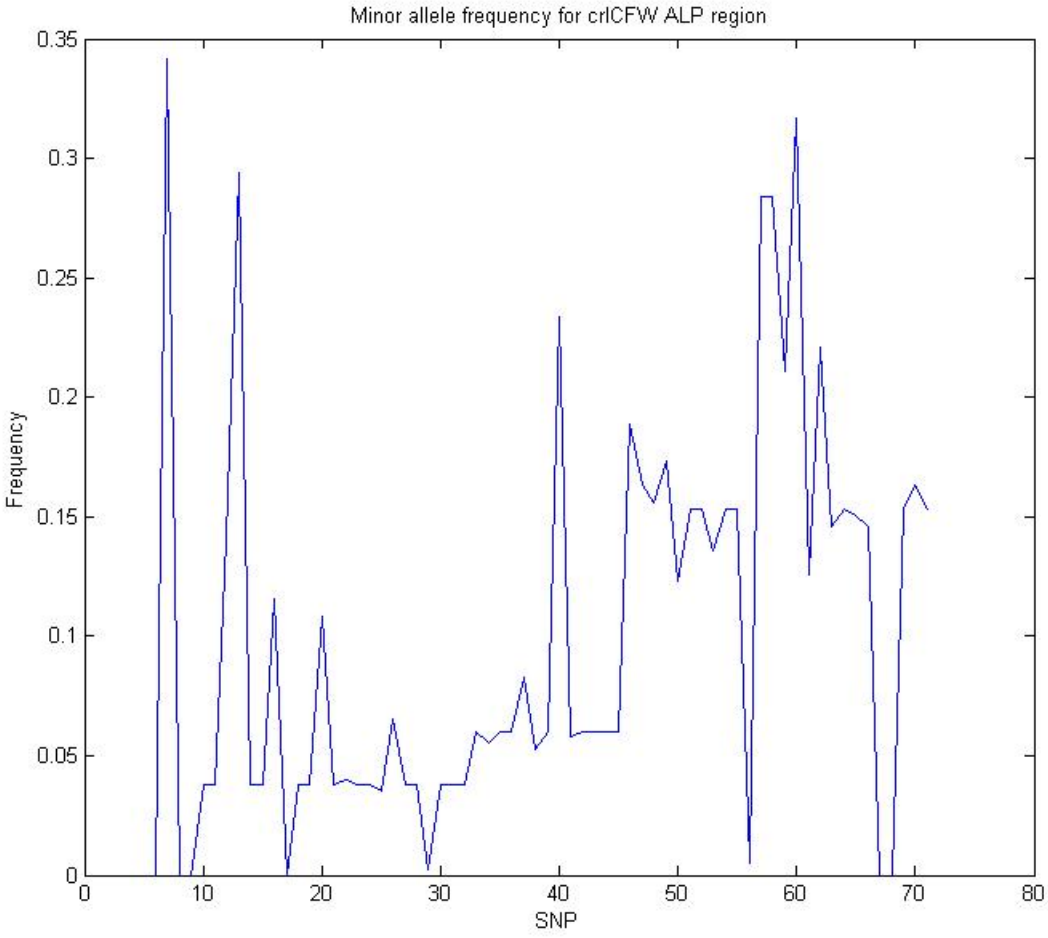

Figure 2.13: Minor allele frequency of the quantitative trait data set

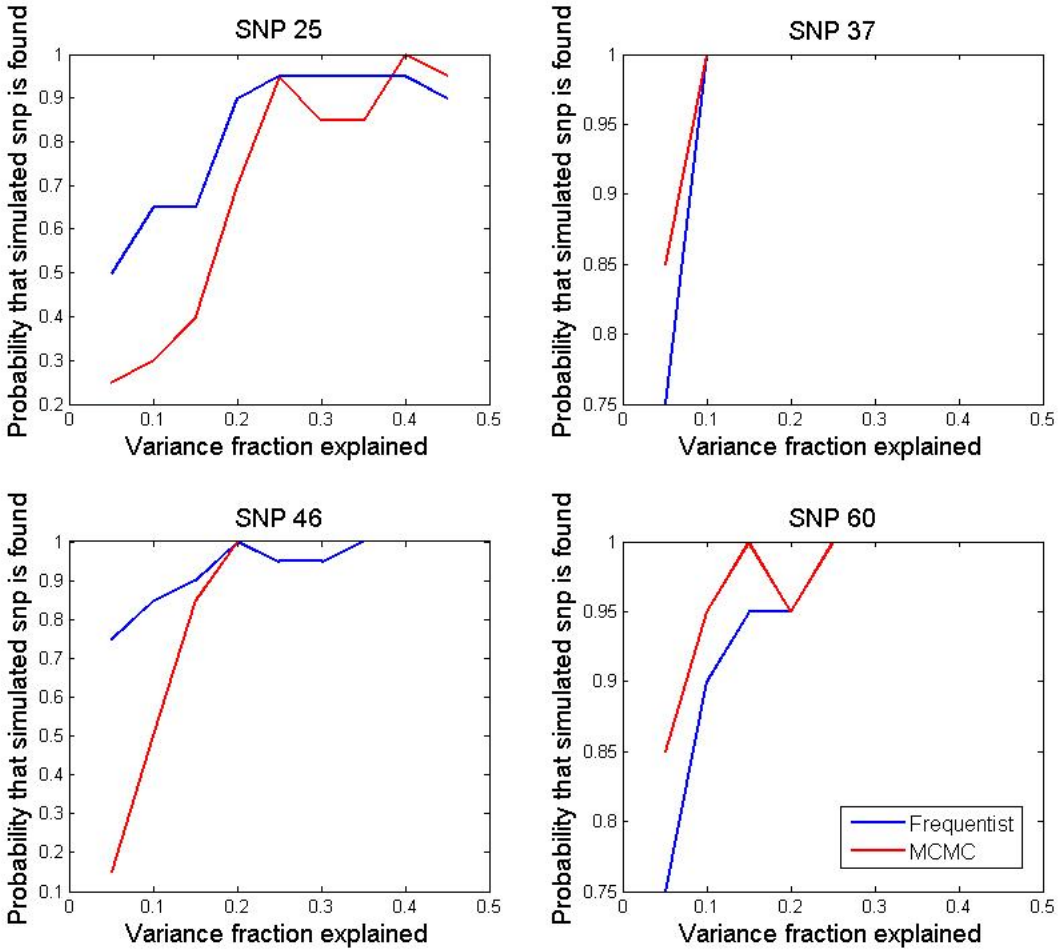

Figure 2.14: Simulation results for qtl with normal prior based on four different causal SNPS as labelled. The X axis shows variance fraction explained by the causal SNP. Y axis the probability that the simulates causal SNP one of the 5 markers most often included in the model. The red lines (labelled MCMC) is the results for the MCMC model and the blue line (labelled likelihood) details the results for the frequentist maximum likelihood method

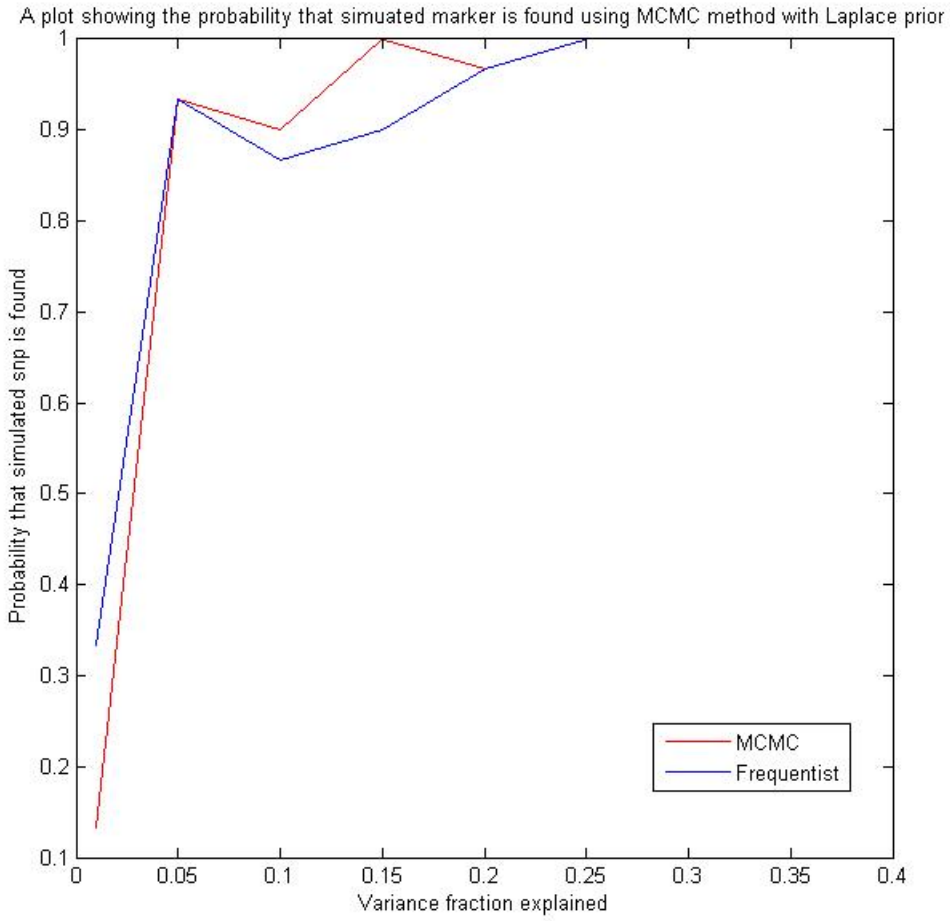

Figure 2.15: Simulation results for qtl X axis shows variance fraction explained by the causal SNP. Y axis the probability that the simulates causal SNP one of the 5 markers most often included in the model. The red line (labelled MCMC) is the results for the MCMC model and the blue line (labelled likelihood) details the results for the frequentist maximum likelihood method

After these simulations it was time to compare my method with another Bayesian method. For this I chose to use BIMBAM. This is a popular method devised by Stevens et al in [69]. BIMBAM is a Bayesian regression method that uses priors on phenotype mean and variance, on SNP effects and which SNPs are quantitative trait nucleotides. They also use priors on effect sizes, regression coefficients, phenotype mean and precision. The BIMBAM method uses Bayes factors to determine association between genotypes and phenotypes. Using the program they devised I ran BIMBAM on the same set of phenotype simulations as used above for the Laplace priors. As can be seen in figure 2.16 BIMBAM and my MCMC perform similarly for SNP 37 but for marker 25 it outperforms my method but does not perform as well as the frequentist method.

Iteration order

As with the case control data I wanted to see what effect permuting the updating order of the SNPs would produce. The results for this can be seen in figure 2.17. Neither method performs particularly well, and though at low variance levels the gap between the methods widens, this adaptation of the method has lead to an improvement in performance.

Dominant effects

Again, I also investigated the effect of recoding the genotype matrix for dominance effects using the same method as stated in the case/control section. The results for this can be seen in figure 2.18, where the Laplace version of the method was used. As you can see, whilst neither method is very good at identifying the correct marker in this recoded problem, both methods behave in a similar fashion, which is an improvement for the MCMC method.

Epistasis

Epistasis is an interaction between genes, the term was used initially in 1909 in [81] [82] where the author uses it describe the presence of one allele preventing another from expressing. A simple example of epistasis is mentioned in [82]. Imagine two

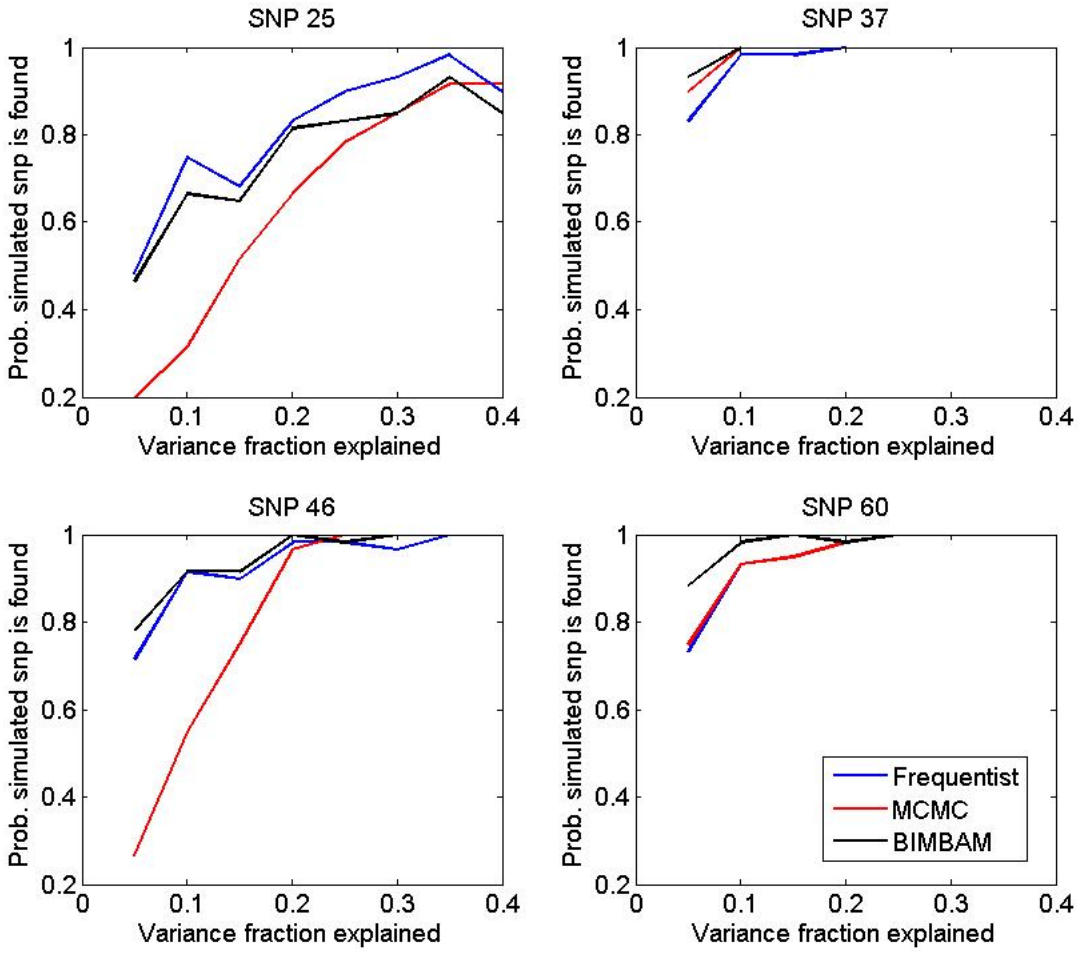

Figure 2.16: Simulation results for qtl X axis shows variance fraction explained by various different causal SNPs as labelled. Y axis the probability that the simulates causal SNP one of the 5 markers most often included in the model. The red lines (labelled MCMC) is the results for the MCMC model and the blue line (labelled likelihood) details the results for the frequentist maximum likelihood method. The black line (labelled BIMBAM) show the results for the BIMBAM method.

areas of the genome which determine the fur colour of mice; one which can be B or b and another G or g. If the genes had an epistatic interaction the outcomes would be as shown in table 2.1

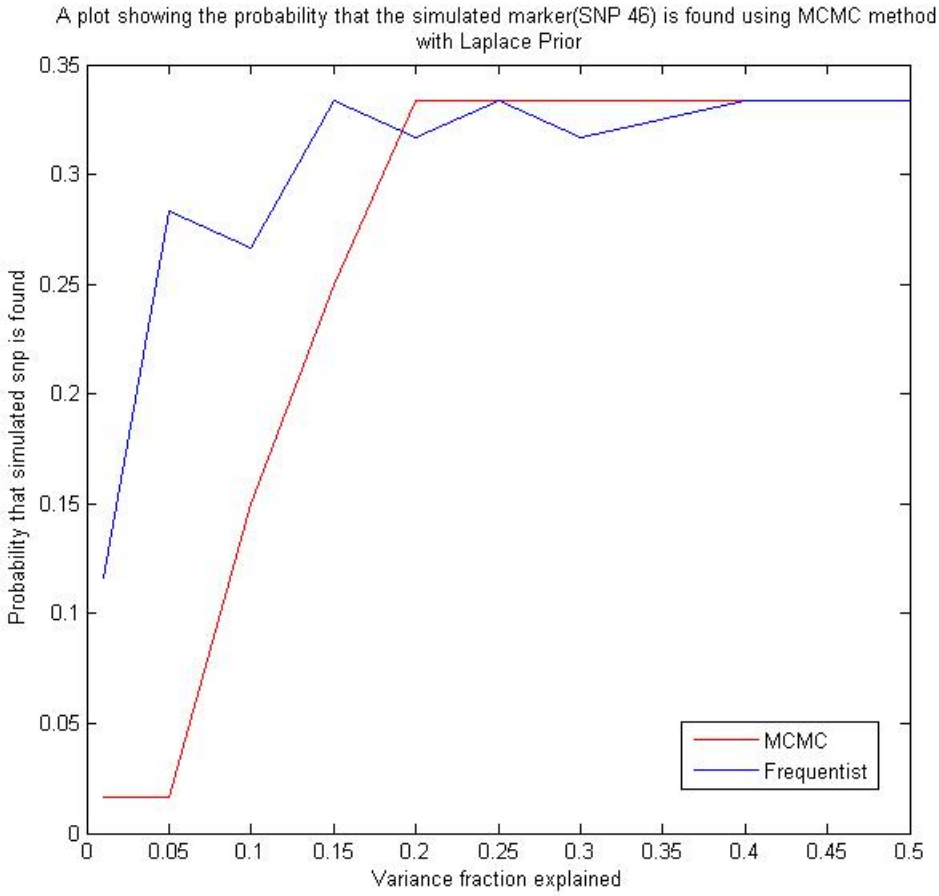

Figure 2.17: Simulation results for qtl X axis shows variance fraction explained by the causal SNP, with permuting the iteration order. Y axis the probability that the simulates causal SNP one of the 5 markers most often included in the model. The red line (labelled MCMC) is the results for the MCMC model and the blue line (labelled likelihood) details the results for the frequentist maximum likelihood method.

Genotype	gg	Gg	GG
bb	white	grey	grey
Bb	black	grey	grey
BB	black	grey	grey

Table 2.1: Table showing a simple example of epistasis, reproduced form [82], where the elements of the table signify fur colour

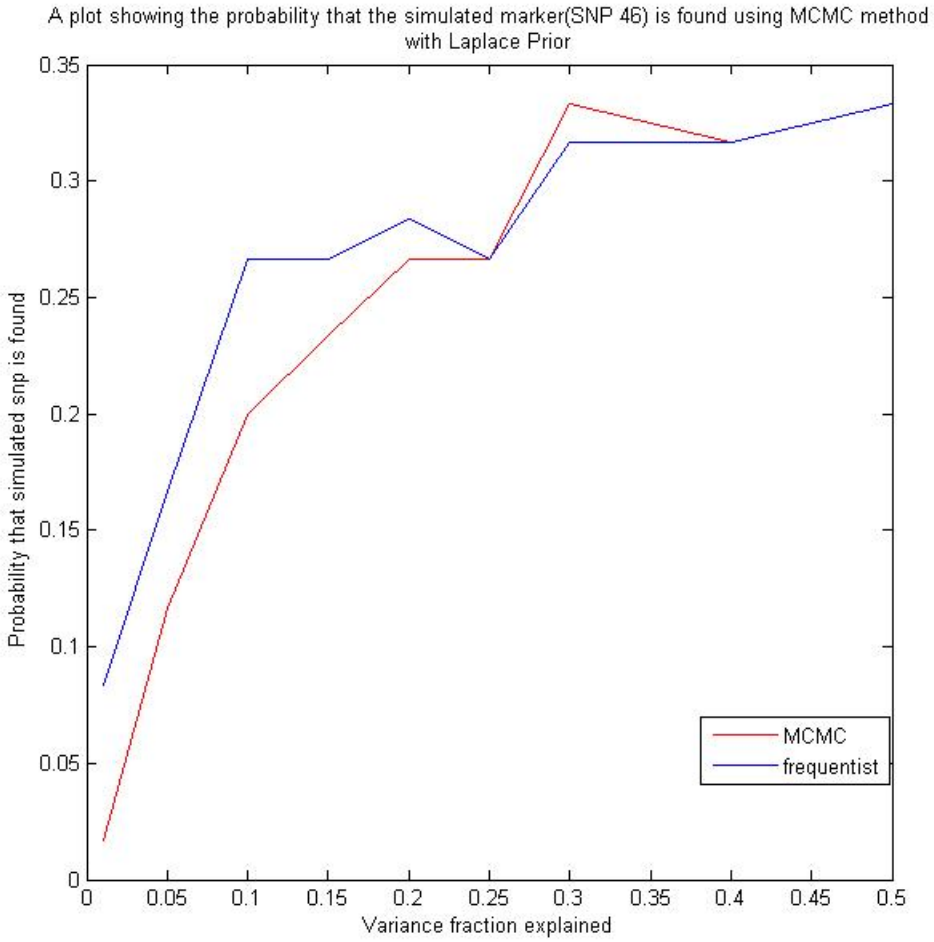

Figure 2.18: Simulation results for qtl X axis shows variance fraction explained by the causal SNP for dominant effects. Y axis the probability that the simulates causal SNP one of the 5 markers most often included in the model. The red line (labelled MCMC) is the results for the MCMC model and the blue line (labelled likelihood) details the results for the frequentist maximum likelihood method

A mathematical definition of epistasis is

$$y = \mu + a_1x_1 + b_1z_1 + a_2x_2 + b_2z_2 + c_{aa}x_1x_2 + c_{ab}x_1z_2 + c_{ba}x_2z_1 + c_{bb}z_1z_2 \quad (2.41)$$

Where x and z are variables representing additive and dominant effects,μ is the mean, a_i are the coefficients of the additive effects, and b_i are the coefficients of the dominant effects for each loci. c_{aa} is the coefficient of additive additive effects, c_{dd} is the coefficient of dominant dominant effects and c_{ad} and cda are the coefficients of the cross dominant additive effects. This definition is taken from [82] and works for

quantitative phenotypes.

Much research has been devoted to looking at epistatic effects such as in [83], where the role of epistasis in recombination and double strand break repair is investigated, or in [84], it is shown that epistasis plays a part in causing autoimmune diseases in some strains of mice. It is hypothesised in [85] that epistasis plays a major role in a persons susceptibility to many common diseases. However, it is argued in [86] that epistasis is ignored by many studies into complex traits.

Using an Arabidopsis data set, I created a new genotype matrix $\tilde{X}$ which is coded for epistasis by

$$\begin{aligned} \tilde{X}_{n,k} = &\quad 0 \text{ if } X_{n,i} = X_{n,j} = 1 && (2.42)\\ = &\quad 2 \text{ if } X_{n,i} = X_{n,j} = 2 && (2.43)\\ = &\quad 1 \text{ otherwise} && (2.44) \end{aligned}$$

where $k = 1..m$ and m is the number of distinct pairs of the SNPs i and j. This data was chosen as there is a known phenotype, time to bud, which has shown evidence of epistasis in Arabidopsis [17]. The data used as the basis for simulations in this section comes from the MAGIC lines as described in [97]. As the MAGIC lines are inbred this is haplotype data. I then ran some simulations, where I simulated an epistatic qualitative trait, the results of which can be seen in figure 2.19. As you can see neither method is particularly good an detecting epistatic effects, but my MCMC method performs only half as well as the frequentist method.

Real Data

Simulations are good for evaluating methods, but methods should also be run on real data as it contains variations and noise which can be difficult to model in simulated data. For this comparison between methods I have looked at two real data sets, the human case control and mouse QTL data sets that I used as a basis for my simulations. To see the performance of the data, I have plotted the probability of each SNP being included in the model after the burn in period. For comparison I have used the maximum likelihood and GLM methods as used above in all the relevant simulations and both MCMC methods use the Laplace prior. My method took about

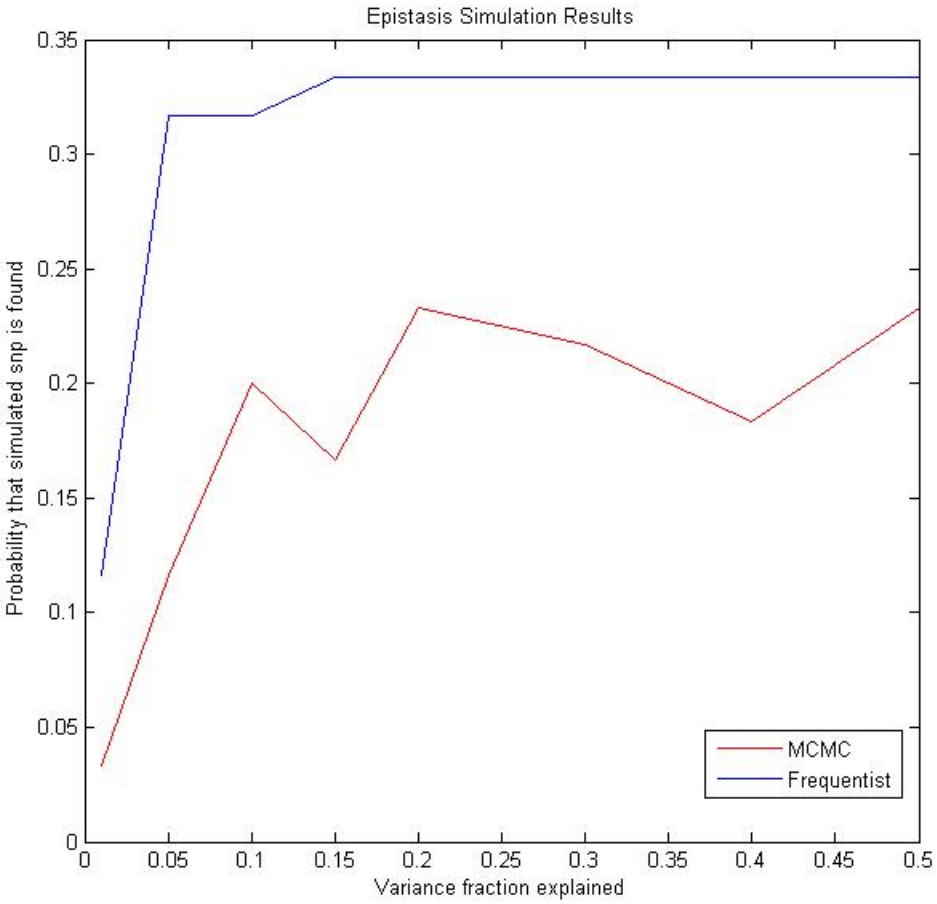

Figure 2.19: Simulation results for epistasis, X axis shows the total variance fraction explained by the causal pair of SNPs. Y axis the probability that the simulates causal SNP one of the 5 markers most often included in the model. The red line (labelled MCMC) is the results for the MCMC model and the blue line (labelled likelihood) details the results for the frequentist maximum likelihood method

3 minutes to identify a causal SNP from real case control data, and about 5 minutes to reach a result when given real QTL data. The results for the case/control data can be seen in figure 2.20, the QTL data results can be seen in 2.21. From the results for the case control data, we can see that the MCMC method identifies the same causal SNP (134) as the frequentist method, but that it does so it a much cleaner fashion. This is slightly surprising considering the performance of the method in the simulation tests.

However in the QTL case, we have a very different story. The MCMC analysis

has not matched any of the most highly associated SNPs found by the frequentist model, it has instead locked onto markers neighboring those SNPs, except in the case of the most highly correlated. Whilst the noise of real data was beneficial to the case/control analysis it has significantly hampered the QTL analysis. This is again is not quite the outcome I would have predicted although the simulations did indicate that it was never going to perform particularly well. Maybe the case/control MCMC method is more robust to the noise of real data.

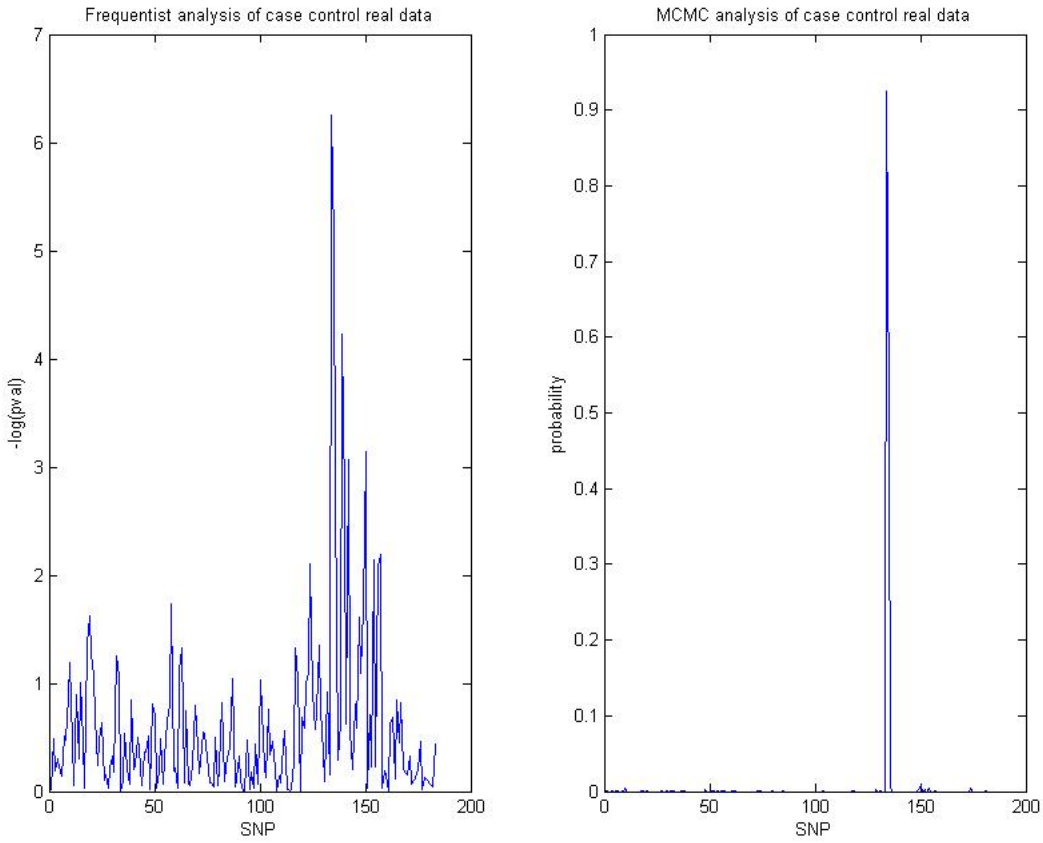

Figure 2.20: Frequentist and MCMC analysis of real case control data

Conclusions

One significant disadvantage of my MCMC method to the standard frequentist or even BIMBAM is computational time, it takes significantly longer to run my methods than

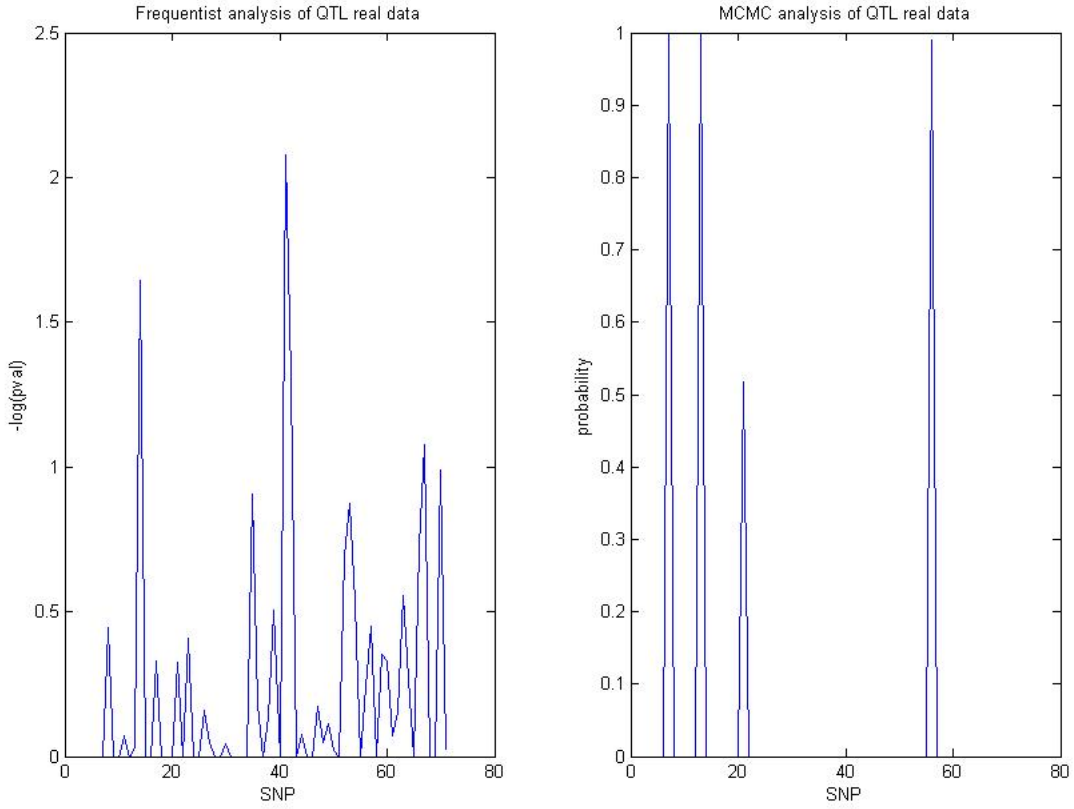

Figure 2.21: Frequentist and MCMC analysis of real QTL data

to use the more established methods. Also this method can only be practically used on a small candidate region due to the long computational time. A possible solution to this is to use graphics processors (GPUs) to run the algorithm. GPUs have the benefit that they are designed to carry out repetitive tasks very quickly which sounds ideal for implementing MCMC methods. This is discussed in the next chapter.

After many months of refining and evaluations the best results we can draw from my MCMC methods is that, overall, they perform reasonably and find the correct answer in a majority of cases. One thing to note, when analysing the performance of my method in the previous simulations, is that my method is much more sensitive to the random noise generated in the simulation processes than the frequentist methods. This is apparent in the much larger deviations from a smooth curve in the results plots for the MCMC method.

When looking at real data we can see that my methods perform exceptionally well for the case control data and terribly for the QTL data. Considering the overall performance of the methods in the simulations the QTL result is slightly surprising but it is the case/control analysis of real data which was truly surprising and may warrant further investigation in the future.

Chapter 3

Computing on the Graphic Processing Unit

Introduction

The Graphics Processing Unit (GPU) was invented by Nvidia in 1999 [87]. It is a computer chip which allows computers to rapidly build images. It does this by having many cores (in the order of a thousand for the latest desktop GPUs) so that operations are massively parallelised. A recent advance in computing is to run code which traditionally is carried out on the CPU on the GPU instead. This has been made possible by Nvidia's development of the Cuda programming language. Whilst this is an adaptation of C it still requires an investment of time to learn how to code in the new language. Mathworks has created the parallel computing toolbox, as an addition to standard Matlab, which requires very little time spent learning new commands to be able to take advantage of the benefits of GPU computing. The main commands to learn are GPUarray(), which sends a matrix to the GPU, and gather(), which returns an object to the CPU. Other commands work as normal providing that all objects involved are located on the GPU.

To test the abilities of the parallel computing toolbox we purchased two servers. Each server uses two Nvidia GeForce GTX 680 graphics cards. These cards each have 2GB of RAM and 1536 cores. These GPUs are standard high street available cards, albeit high end ones. All operations were compared to the CPU on these servers which has dual 8-core Sandy Bridge processors with hyperthreading enabled, so they can work as a 32 core CPU server.

Literature Review

In [88] Armstrong et al used GPU computing to improve the speed of Multifactor Dimensionality Reduction which is used in epistasis analysis. They found a reduction factor of 150 over a C++ implementation (running on a Beowulf cluster) of the algorithm and a reduction factor of 160 over a Java implementation (on an 8 core machine). GPU optimization has also been used in high throughput sequencing alignment [89]. Here the authors used the MUMmerGPU algorithm and they found a 10 fold reduction in computing time, this includes the fact that very large data sets must be sectioned into more pieces as GPU processors have smaller amounts of available RAM (the GTX680 card has 2GB of RAM). Although GPU coding does offer considerable time reductions for computation, similar reductions can be found by using parallelisation on CPU cores. In [90], using SNPrank (an algorithm that ranks SNPs "on their importance in the context of a phenotype-specific interaction network"[90]), it was found that the multithreaded CPU was 14 times faster than the single CPU version, whereas the GPU version was 1.1 times faster than the multithreaded CPU.

GPU computing has also been used in QTL detection [91]. Here, a GPU version of QTLMap was developed (in which the existing version was already optimised for multicore threading) and the authors found a reduction in computing time of up to 75 times. In genotype imputation there was found to be a 10-15 times reduction in computational time for their HMM depending on computer hardware [92]. In [93] the authors developed a haplotyping and genotype imputation method called MendelGPU. They compared it to other highly used methods; impute2 and beagle. For haplotype phasing and genotype imputation, their GPU method was 6 times faster than beagle and 30 times faster than impute2. For genotype imputation for a low pass sequencing study the speed up factors were 104 and 144 respectively. They also found that the maximum memory used was significantly lower for the GPU method. In gene regulatory networks [94], a speed up factor of 145 times was achieved. In [95] GPUs were used as parallel cores to carry out population and sequential MCMC and a speed up factor of between 30 and 500 was found.

Basic Operations

One of the most important factors to know when comparing GPU Matlab and CPU Matlab is how the two versions compare on standard arithmetic operations. These are important as they are the basis of all more advanced operations. Some notes are necessary before the results. It is much faster to create the matrix directly on the GPU than generate vectors and matrices on the CPU and then send them to the GPU. For example, it takes 1.5seconds to create a $(10000, 10000)$ matrix on the CPU and then send it to the GPU, but it takes only 0.1 seconds to create it directly on the GPU and 0.5 seconds to return it to RAM memory. For fairness the CPU created objects are then sent to the GPU so that the operations are carried out on identical objects. The timing tests were carried out using one of the GPU cards, and each command was repeated 20 times. For this initial set of timing tests the time is broken up with the creation of the matrix included in the time for CPU and the sending of the matrix to the GPU included in the time for GPU. This addresses the issue of giving a realistic idea for a computational time when running an individual operation on an existing data set, not a random creation. The timing was carried out using the tic toc function.

Command	Size	Average CPU Time	Average GPU Time	Fold Difference
for loop adding: $a = a + 1$	$100,000$	0.0004	6.18	15000 fold increase
Adding two vectors	$(10000, 1)$	0.0003	0.04	130 fold increase
Inner product of two vectors	$(1, 10000)$ $(10000, 1)$	0.0007	0.004	5.7 fold increase
Outer product of two vectors	$(10000, 1)$ $(1, 10000)$	0.11	0.06	1.8 fold decrease
Adding two matrices	$(10000, 10000)$ $(10000, 10000)$	2.68	0.72	3.7 fold decrease
Multiplying two matrices	$(10000, 20000)$ $(20000, 10000)$	17.49	1.17	15 fold decrease
Transposing a matrix	$(10000, 20000)$	3.56	0.97	3.7 fold decrease
Inverting a matrix	$(8000, 8000)$	6.27	12.17	1.9 fold increase
Multiplying a vector and a matrix	$(1, 20000)$ $(20000, 10000)$	2.61	0.51	5.1 fold decrease

Table 3.1: Table to compare the times taken to perform various basic operations carried out on the CPU and GPU.

Looking at the results, in table 3.1 we can see a pattern in which most commands are faster on the GPU. Any command which ends in a single value is slower on the GPU, this will be due to the way in which GPU processors function. As mentioned previously, a GPU consists of hundreds of cores, so it can do many operations in parallel. It is also important to note that the maximum size of matrix which a 2GB GPU card can hold is $(26039, 20000)$, however with a matrix this size it is impossible to carry out any calculations as this all but fills the cards memory.

For the second set of timing tests, see table 3.2, the creation of the objects and the sending of objects to the GPU was removed from the timing set. This was to give an understanding of the core computational time when adding further calculations to an existing script.

Command	Size	Average CPU Time	Average GPU Time	Fold Difference
for loop adding: $a = a + 1$	$100,000$	0.0004	6.575	17000 fold increase
Adding two vectors	$(10000, 1)$	0.00005	0.04	800 fold increase
Inner product of two vectors	$(1, 10000)$ $(10000, 1)$	0.0001	0.002	20 fold increase
Outer product of two vectors	$(10000, 1)$ $(1, 10000)$	0.109	0.056	1.9 fold decrease
Adding two matrices	$(10000, 10000)$ $(10000, 10000)$	0.162	0.087	1.9 fold decrease
Multiplying two matrices	$(10000, 20000)$ $(20000, 10000)$	12.77	0.053	240 fold decrease
Transposing a matrix	$(10000, 20000)$	1.04	0.164	6.3 fold decrease
Inverting a matrix	$(8000, 8000)$	5.71	12.09	2.1 fold increase
Multiplying a vector and a matrix	$(1, 20000)$ $(20000, 10000)$	0.089	0.007	13 fold decrease

Table 3.2: Table to compare the times taken to perform various basis operations carried out on the CPU and GPU, excluding the transfer of the inputs from the timing.

Whilst there is no change as to which operations cause increases and decreases in computational time, the extent of the time differences are worthy of note. For the "for loop" addition, whilst only moving a single digit object which requires very little memory, we still get a further 2000 fold increase in time, but this can easily be ascribed to general variation in computation due to the magnitude of the numbers involved. This variation in result could also be observed between the two tests for matrix multiplication. What is more noteworthy is the change in computation time when comparing the addition of two vectors which is an increase of 670 on the previous test and also the change for the inner product of two vectors which increased over 3 times. Another important change is that the benefit of using a GPU for adding two matrices decreased.

Complex Operations

Whilst the standard operations are the building blocks of mathematics it is also useful to see how these combine and affect the computational time of more complex operations. Here I present the result of investigation into 5 further operations of varying complexity.

Least Squares Regression

To begin, let us look at least squares regression. For this we take two inputs, a vector y and a matrix X, then the following calculations are made

$$X_t = X' \tag{3.1}$$

$$V = (X_t X)^{-1} \tag{3.2}$$

$$\beta = V X_t y \tag{3.3}$$

Computation times for this sequence of operations was compared using a $(5000, 5000)$ matrix and a $(5000, 1)$ vector. Running the entire programme on the CPU takes 13.94 seconds whilst on the GPU it takes 7.25 seconds. By using the information from the previous table, it might be expected that to minimise computation time we should use the CPU for the inversion step. Following this and using a mixture of CPU and GPU operations the computation time became 16.19 seconds. This is a surprising result. However, further investigation found that the percentage increase in time for the inversion of a matrix on the GPU is smaller for a $(5000, 5000)$ matrix than the $(8000, 8000)$ used for the original timing tests. Also the small decrease in time for the CPU inversion is not enough to offset the increases caused by moving the matrices back and forth between the CPU and GPU in this case. It should also be noted that the Matlab function "inv" can be unstable, especially when there are more matrices and vectors being held in the RAM memory, either on the GPU or that accessible to the CPU, than just that matrix being inverted. This could be another factor in the somewhat surprising result in this test.

Maximum Likelihood

Another operation we can look at is the maximum likelihood method. This is a standard frequentist method for finding causal SNPs in QTL analysis and it is the frequentist method I used in the previous chapter. To calculate the maximum likelihood the following computational steps are taken

$$\beta = \frac{\Sigma y \Sigma t - n\Sigma(yt)}{(\Sigma(t))^2 - n\Sigma t^2} \tag{3.4}$$

$$\mu = \frac{1}{n}\Sigma(y - \beta\sigma t) \tag{3.5}$$

$$\sigma = \frac{1}{n}\Sigma(y_i - \beta\mu t_i) \tag{3.6}$$

$$\text{log likelihood} = -\frac{1}{2}n\log 2\pi\sigma - \frac{1}{2}\Sigma\frac{(y_i - \mu - \beta t_i)^2}{\sigma} \tag{3.7}$$

where y is the phenotype vector and t is the linear predictor. Using the CPU and just timing the above four lines, (not the creation of the linear predictor) the CPU timing is 0.0002 seconds whereas when the same four lines are run on the GPU it takes 0.0028 seconds. This result is not surprising as from the timing tests we know that any calculation which results in a single value answer is slower on the GPU. The time difference between CPU and GPU and the overall magnitude of the computational time suggest that splitting the work between CPU and GPU would take longer due to the time to move objects between the two processors.

Forward Selection

A third operation we can look at is forward selection. This is a method of model selection which adds in the marker which explains the highest proportion of variance in the model matrix automatically until a desired end point is reached. Here forward selection with a logistic framework was considered, which could be used for looking at case control data such as that used in the previous chapter of this thesis. The beta

values were calculated using the following Newton Raphson step

$$\begin{aligned} \beta_i &= \beta_{i-1} + (X_k' V X_k)^{-1} X_k'(y-p) && (3.8)\\ p(j) &= (1 + exp(-(X_k)_j \beta_{i-1})^{-1} && (3.9)\\ V(j,j) &= p(j)(1-p(j)) && (3.10)\\ V(j,m) &= 0 \text{ otherwise} && (3.11) \end{aligned}$$

where X_k is a matrix containing those columns of the model matrix X which correspond to the markers included in the model, and y is the response vector of 0's and 1's which equate to case or control status. To evaluate the fit of the models the Akaike information criterion AICc was used, which is

$$AICc = \frac{2k(k+1)}{n-k-1} + 2k - 2log(L) \tag{3.12}$$

where k is the number of markers in the model, n is the total number of markers in the data set and L is the likelihood, which in the logistic case is calculated by

$$L = \Pi_{i=1}^{n} \left(\frac{1}{1+\exp X_i\beta} \right)^{y_i} \left(\frac{\exp X_i\beta}{1+\exp X_i\beta} \right)^{1-y_i}. \tag{3.13}$$

A new model is chosen by the addition of a marker that minimises the $AICc$. The difference between the $AICc$ of the old model and the new model is considered via

$$exp\frac{(AIC_{k-1} - AIC_k)}{2}. \tag{3.14}$$

The process is then iterated until the models converge or a desired number of explanatory markers has been reached. For these tests I used a data system with 200 patients and 20 markers which ran until convergence or until 5 markers were in the model. When the procedure was run on the CPU the computational time was 0.652 seconds whilst on the GPU this took 310 seconds; a dramatic increase. A mixture of the two processors took 0.966 seconds. Again this method shows that the strengths of the GPU lie in matrix manipulation not vectors or single value objects. An examination of the time difference between the purely CPU implementation and the mixed procedure shows that a significant proportion of the differences in time between the procedures is devoted to sending the objects between the processors. As the objects increase in size this difference in performance will decrease. Another way to decrease

computation time is to consider parallel computing. This is most easily visualised as running the iterations of a "for loop" at the same time on the different cores of the CPU. When this is applied to the above forward selection method the average computational time was 0.282 seconds which is approximately half the time seen when using the CPU as a single core.

Single Value Decomposition

The next operation we can look at is single value decomposition. This is an operation for an m by n matrix, X, such that

$$X = U\Sigma V \tag{3.15}$$

where U is an m by m matrix, V is an n by n matrix and Σ is a diagonal matrix whose entries are the singular values of X. The matrices U and V are orthogonal if they are real valued and unitary matrices if they are complex. Unlike the previous operations considered in this chapter, Matlab's own inbuilt single value decomposition function is capable of handling GPU objects, so we will use that to see how Matlab's own functions process GPU objects. I investigated this function over many different sizes of the starting matrix X. For small matrices, I began with $(100, 100)$, there was a significant difference; 0.006 seconds average for the CPU compared to 0.471 seconds for the GPU. As the size of the matrix increased the percentage difference between the two processors narrowed, for a $(2000, 1000)$ matrix the average times were 0.661 seconds and 1.296 seconds respectively, for $(5000, 5000)$ they were 31.65 seconds and 36.47 seconds. Continuing to increase the size of the matrix lead to a reversal of the performance so that the GPU became the faster processor. For a $(8000, 5000)$ matrix, the CPU average was 60.65 seconds whereas the GPU average was 52.2 seconds. This trend continued for the largest sized matrix I tested, a $(8000, 8000)$ matrix. Here the CPU average was 140 seconds and the GPU average was 115 seconds. These results compare favourably to the information I discovered in the initial testing, that GPU processors are best at handling large matrices.

Markov Chain Monte Carlo

Lastly we get to the motivation for this investigation into the use of graphics processors in statistical computation; the MCMC method as described in the previous chapter. For comparisons I evaluated both the case control and the quantitative trait versions of the MCMC as they use different update procedures which would put different stresses on the processors. Let us begin by looking at the case control version. In this method we have updates to at most two β values per iteration and so other updates are single values and vectors used throughout the method. For evaluation purposes I used the human colon cancer case control data set from the previous chapter, which contains 183 markers, and used that to simulate 1000 cases and 1000 controls. To begin to test the suitability of the method I looked at a single run of the MCMC. That involves updating the β value of each marker once. This run took an average of 0.151 seconds on the CPU and 2.197 seconds on the GPU. This is a dramatic increase, over 1300 %. However I was optimistic that a significant proportion of this could be attributed to the movement of objects between processors. So for the next series of tests the computation was extended to 100 runs. This however caused a deepening of the problem. The CPU took 3.597 seconds whilst the GPU took 288 seconds. This is due to the amount of vector calculation inherent to my update scheme.

Next we should consider the quantitative trait version of the MCMC. This was run with phenotype data from outbread mice in the ALP region. The data set contains 71 markers and 199 individuals. Following the same procedure as above we started with just 1 run. For the CPU this averaged 0.221 seconds but for the GPU the average was 5.47 seconds. A much poorer result than for the case control version. So the next step was to extend the operation to 100 runs despite an unfavourable result being expected. For the CPU version it took 8.651 seconds but the GPU took over 10 minutes. This was due to the fact that the quantitative trait MCMC uses several commands from the Matlab statistics toolbox to make calculating the Gibbs steps and β values simpler. However these operations have not yet been updated to allow the use of gpuArray objects. This required frequent movement of objects between the CPU and GPU to occur, leading to the dramatic slow down of the calculation.

Unfortunately due to the way that my MCMC is designed it is not possible to use parallel "for loops", as each successive iteration of the method depends on the previous one.

Summary Table

Here for easy comparison of the results of the statistical methods tests I present a table (table 3.3) which summaries the results based on averages for each method with the largest initial inputs.

Method	Average CPU Time	Average GPU Time	Fold difference
Least Squares Regression	13.94	7.25	1.9 fold decrease
Maximum Likelihood Calculation	0.0002	0.0028	14 fold increase
Forward Selection	0.652	310	500 fold increase
Single Value Decomposition	140	115	1.2 fold decrease
Markov Chain Monte Carlo	3.597	288	80 fold increase

Table 3.3: Table to compare the results of various complex operations carried out on the CPU and GPU

Conclusions

This project has been an interesting investigation into the relative performance of CPU and GPU processors and their uses in statistical programming. From this project we can conclude that the GPU processor's strengths lie with matrix calculations, however it has significant weaknesses with vectors and single valued objects. From a statistical methods view point, least squares regression and single value decomposition are good uses of the inherent strengths of a graphics processor but forward selection, when approximation to roots is involved, or Markov Chain Monte Carlo methods which cannot be parallelised are methods for which the use of GPUs should

be avoided. In the future when all Matlab functions are compatible with gpuArrays then MCMC should be revisited. There is also a future in combinations of CPU and GPU processing, however the time to move objects between RAM and GPU memory at the moment is too prohibitive to make full use of graphics processors' strengths. Another problem with current GPU technology is the low levels of available memory when compared to what can be fitted in RAM for CPU use. My laptop, for example, has 12Gb of RAM for CPU use and 2Gb for the GPU. So while computational use of GPUs would be most beneficial for the very large matrices which can occur in statistical genetics, the technology is currently lacking on a personal level. Currently there exist industrial graphic processors with 8Gb of memory (Nvidia's Tesla processors). These processors also have faster communication and calculation speeds, but are too expensive for general use. For now, I believe that the future of speed optimisation for statistical computing lies in parallelisation of methods between CPU cores; current home processors have 4 cores per processor but up to 8 available with hyperthreading. This would allow theoretically up to an 8 times reduction in computational time whilst still allowing the use of large data sets. The GPU toolbox for Matlab also provides parallel computing optimisations and this would be an area I would like to explore more fully given more time.

Chapter 4

Genotyping by Multiplex Pooled Sequencing

Introduction

In this section I am looking at multiplex pooled sequencing data sets. The aim here is to develop a method to genotype outbred populations quickly and cheaply, similar to the method discussed in [96]. The method used in [96] is based on a restriction digest of the genome; they sequence the same small sections of the genome at high coverage, whereas we sequence randomly from the genome. We have two different populations which we will be using, one is the MAGIC [98] lines of Arabidopsis and another is an outbred mouse population [99]. A necessary feature of these populations is that they are descended from a number of known inbred founders [97]. Genotyping is then carried out as follows: the individual DNA samples are bar coded and pooled together, then they are Illumina sequenced at about 0.1x coverage. From this we have a set of sequence data, at one allele only, of each individual which has information for a small proportion of the genome. We align this set of data to a reference genome, allowing us to remove any SNPs which are from repetitive regions and discard any called positions which are not known to be SNPs. For each sequenced individual we then have a unique subset of alleles. I will create a hidden Markov model (HMM) to try to reconstruct the genome for each individual by using the founders of the lines as the hidden states for each SNP. From this we should be able to map the break points between each section of DNA from the different founders and then use imputation to

create the complete genome. We can then use these highly certain reconstructions to find associations.

Literature Review

HMMs are a form of dynamic Bayesian network where the model is assumed to follow a Markov process. There are several different types of HMM, the most commonly used is simple HMM where there is one set of hidden states and one set of observed data, however there are also nested HMMs such as those used in [101], where for one set of data there are two different sets of hidden states. In our data sets the phase and founder are independent hidden information for each observed SNP. Pair HMMs are another type [100]. These have two sets of observed data for the same set of hidden states. HMMs were developed in the 1960s by L.E.Baum [102]. There are several different algorithms which can be used to identify the hidden states of an HMM. The Viterbi algorithm, first proposed in 1967 in [103], identifies the most likely state at each data point, whereas the forward and backward algorithm calculates the posterior marginals of all the hidden state variables. These algorithms are described in the model section of this chapter.

HMMs and simpler dynamic programming algorithms have been used in many areas of statistical genetics. One example is the Happy algorithm [104]. This was first developed as a dynamic programming algorithm to assign probabilities of an allele coming from a specific founder in heterogeneous stock mice. This algorithm allowed the authors to finely map small effect QTLs.

HMMs have also been used to identify copy number variants. These are small sections of the genome that repeat, possibly multiple times in small areas. Papers such as [105] and [106] have used HMMs to analyse this type of data.

CpG islands are areas of the genome where the base pair sequence CG occurs with higher frequency than in the rest of the genome [107]. These areas are thought to play a role in methylation and regulation. HMMs have been used to identify these areas [100], [108].

Another area that has used HMMs in genetic research is sequence alignment. In [109] an HMM is used to produce multiple sequence alignments from “unaligned

protein or DNA sequences"[109]. HMMs can also then be used to turn multiple sequence alignments into a system that can be used with databases to find remotely homologous sequences. [110] is a review paper of these methods.

The Model

A Hidden Markov Model needs two input data matrices, an Emission probability matrix and a Transition probability matrix. An Emission probability matrix signifies how likely it is that the observed value at a point came from a specific state. In this case it signifies how likely it is that a specific SNP observation came from one of the founders. Now as SNPs are single value points and can only take one value in a specific line excluding mutations, this could be considered a binary probability either 1 if a founder has the same allele at that point in the genome or 0 if it does not. However laboratory methods for determining the value of a specific SNP are not completely accurate. They use repetition of coverage of the area to provide a higher probability of a correct value. In the data provided for this project we do not have a depth of coverage to rely on to ensure an error free data set. Also errors in sequencing can occur from the data reads being aligned to the wrong section of the genome, creating SNP values which would not normally be seen. To account for this we set an error rate for the data. So for this project the Emission probabilities are

$$E(i,t) = \begin{cases} \gamma & \text{Foundermatrix}(i,t) = 1 \quad (4.1) \\ \epsilon & \text{Foundermatrix}(i,t) = 0 \quad (4.2) \\ \delta & t = F + 1 \quad (4.3) \end{cases}$$

where Foundermatrix is the matrix of allele values at the genetic locations in the data set, F the number of founders and i is the genetic location. The values of γ, δ and ϵ are scaled for each genetic location such that the sum of each type comes to a specified value, e.g. $\sum_{t,F_{it}=1} \gamma = 0.95$. You will notice that another founder has been added to the matrix. This is to account for an unknown founder or a novel SNP. This unknown founder was added as, for some populations, it is possible that not all of the founders may be known or it may be suspected that a subject of a different linage

was accidentally included in the breeding scheme or it could be that one of the inbred founders of a line died out before a high coverage sequence of its genome could be taken. As mentioned, a novel SNP would arise from mutations unique to a specific subject or one of its ancestors. Due to the way the founder matrix was created a novel SNP at a new location would be excluded from this analysis but a previously unseen base pair at a specific location would be included. As an aside, there are other variations in the genome than SNPs, including insertions, the addition of more base pairs between two previously known locations, deletions, which are the removal of sections from the genome, and copy number variations (CNVs), a repeated section of the genome. All of these events can be longer than one base pair and are excluded from this analysis.

A transition matrix of an HMM denotes the probability of a change in state, e.g. the probability that the observed value at time t comes from state A when at t-1 the observed value corresponded to state B. In these data sets the transition matrix denotes the probability of a recombination event. When a cell undergoes meiosis each chromosome forms bonds with its pair. The single chromosome in the daughter cell contains information from each one of the chromosomes in the corresponding pair of the mother cell. Recombination is a relatively rare event. It usually occurs in one or two locations per meiosis event. However over time these crossovers build up so we should see more than two founders per chromosome. The likelihood of a recombination event increases with the genetic distance between two SNPs, so assuming that recombination is equally likely across the whole chromosome, the recombination rate between two SNPs, with base pair positions A and B where B is greater than A, can be approximated using

$$\phi = \frac{(B - A)mr}{G} \tag{4.4}$$

where G is the length of the genome in base pairs, m is the length of the genome in centimorgans and r is the number of generations which, for the magic lines, is 10. For Arabidopsis G would be 135mbp and the number of centimorgans m, is 500. The length of a centimorgan varies between species as it is defined to be the number of base pairs such that the probability of a recombination event is 0.01. The number of generations is included as a scaling factor because, as mentioned

previously, the number of identifiable recombination events increases over time. The Transition probabilities for the HMM are calculated using recombination rates so that the probability of staying in the same founder is the most likely but it decreases as the distance between SNPs increases. The transition probabilities are then given by

$$\mathrm{T}_i(t,s) = \begin{cases} \frac{1-e^{-\phi}}{F+1} & t \neq s \quad (4.5) \\ e^{-\phi} + \frac{1-e^{-\phi}}{F+1} & t = s \quad (4.6) \end{cases}$$

where F is the number of founders.

The calculation of probabilities of a specific observation coming from a specific founder can be calculated in many different ways. This HMM then uses the Viterbi algorithm. The Viterbi algorithm calculates the most likely path through the data by using the following method

$$\begin{aligned} V(0,0) &= 1 && (4.7) \\ V(0,k) &= 0 \text{ for all } k > 0 && (4.8) \\ \text{for } i &= 1:n && (4.9) \\ V(i,l) &= E(i,l)\max_k(V(i-1,k)T(k,l)) && (4.10) \\ \text{pointer}(i,l) &= \text{argmax}_k(V(i-1,k)T(k,l)) && (4.11) \\ \text{Termination: } P(x,\theta) &= \max_k(V(n,k)T(k,0) && (4.12) \\ \theta_n &= \text{argmax}_k(V(n,k)T(k,0)) && (4.13) \\ \text{path determination for } j &= n:1,\ \theta_{j-1} = pointer(j,\theta_j) && (4.14) \end{aligned}$$

where n is the number of observations [100].

Another method of calculating the probabilities is to use the forward and backward algorithms to calculate the probability of each founder at each data point. The

forward algorithm is [100]

$$\begin{aligned} F(0,0) &= 1 && (4.15)\\ F(0,k) &= 0 \text{ for all } k>0 && (4.16)\\ \text{for } i &= 1:n && (4.17)\\ F(i,l) &= E(i,l)\Sigma_k F(i-1,k)T(k,l) && (4.18)\\ \text{Termination: } P(x) &= \Sigma_k(F(n,k)T(k,0)) && (4.19) \end{aligned}$$

and the backward is [100]

$$\begin{aligned} B(n,k) &= T(k,0) \text{ for all } k && (4.20)\\ \text{for } i &= n-1:1 && (4.21)\\ B(i,l) &= \Sigma_k E(i+1,k)B(i+1,k)T(k,l) && (4.22)\\ \text{Termination: } P(x) &= \Sigma_k(B(1,k)T(0,k)E(1,k)) && (4.23) \end{aligned}$$

However, as the founders are identical at many of the sequenced positions, I feel that the forward and backward algorithms are not suitable for this problem as the probabilities will be very similar and so it may be harder to identify a correct path.

This HMM scheme applies to homozygous, inbred, data. These are populations where both chromosomes in a pair are virtually identical. However populations in general are heterozygous, the chromosomes in a pair differ, and that means that this data can actually be thought of as two separate interwoven data sets. This feature of genetic data is referred to as phasing. Due to the way the experiment was carried out we do not know the phasing of the observed SNP values.

Model Evaluation

To evaluate the model I ran a series of series of simulations. In each simulation I chose the number of total observations and how many blocks I wanted to divide the observations into. Then each block is randomly assigned to one of the known founders. For each simulation I randomly chose a number of base pair positions for chromosome 1 equal to the number of total observations, these positions were then

sorted and divided appropriately between the founder blocks. I then used this set of information to generate a set of of observations to use in my HMM algorithm.

For each pair of total observations and block size, I ran 100 different simulations. To check the performance of my algorithm I calculated the overall percentage correctly assigned to each block for the entire genome. The results of this can be seen in figure 4.1. As we can see in this idealised situation across all block sizes (the numbered columns of the bar chart), the algorithm performs very well and there is a clear relationship between increasing the block size and an improvement in correct assignment to the block. This improvement in performance tapers off between block sizes of 500 and 1000.

Although the algorithm has performed exceptionally well in the first 5 sets of simulations, the data that the HMM is analysing is very far from real data as it does not contain any errors. The current state of sequencing technology produces errors in sequence reads. So for my final set of simulations I created a set of 1000 observations split between 5 blocks but applied an error rate of 5%. The results for these simulations can be seen in the bar labelled error in figure 4.1. Here we can see that the performance has dipped by a couple of percentage points compared to the error free, 200 observation block length, simulations. This reduction in performance is mostly due to occasions when the errors cluster, causing the algorithm to assign an extra founder block.

Real Data

The next step for this project is to take what I have learned from the simulations and apply it to real data. The first run through of real data analysis failed to produce much of what could be said to be founder blocks. This suggested that either the Emission or Transition probabilities needed rescaling. I chose to scale the Transition probabilities by altering ϕ, o I returned to creating more simulations to check that reducing ϕ did not have an adverse effect on the success rate when identifying founders for the simulated idealised and error containing datasets. Reducing ϕ by a factor of $\frac{1}{1000}$ produced the best balance between simulation results and number of founder blocks identified in the real data. In tables 4.1-4.5 you can see the reconstructions

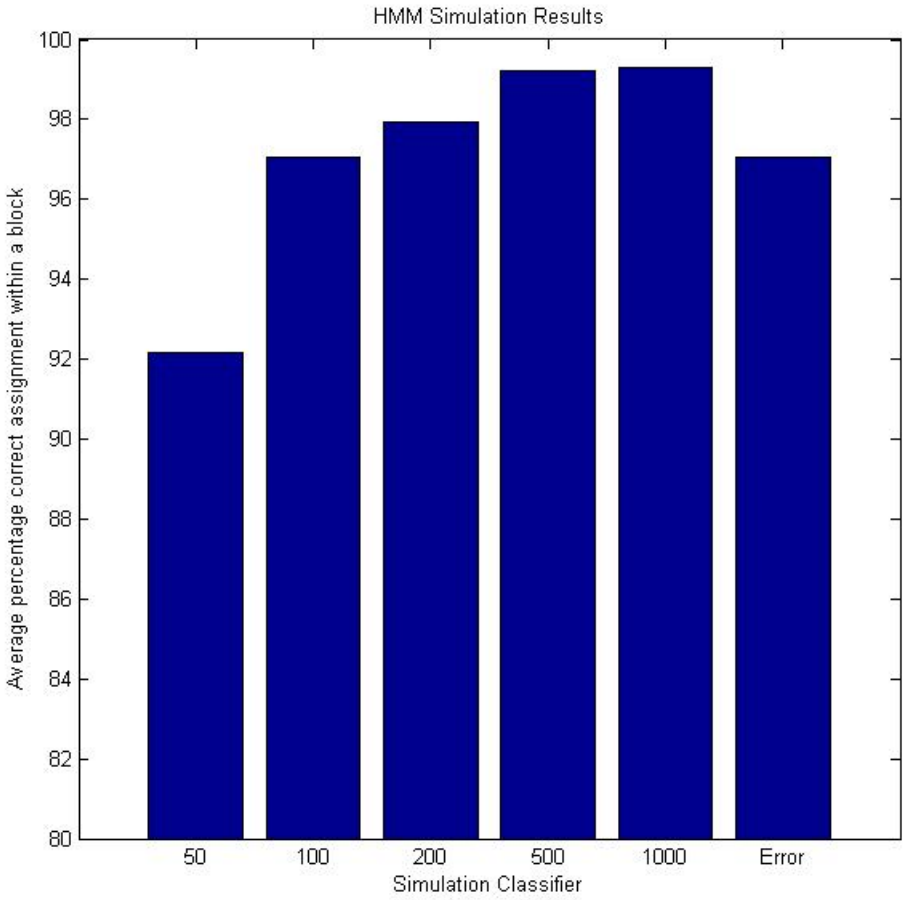

Figure 4.1: A bar chart showing the average percentage correct assignments to a founder block over a range of different block sizes

of chromosome 1 for 5 different plants. From these tables we can see that there are patterns of repeated founders, as you would expect from this type of cross. The pattern is most obvious in table 4.4 and table 4.5.

However when the error rates of these founder blocks are analysed, it averages 0.4, which is much too high. Unfortunately none of my efforts in calibrating the combination of emission and transition probabilities produced significant improvement, so the only conclusion that can be drawn is that this model is not applicable for real data of this form.

Start bp	End bp	Block length	Founder	Markers in block
10495	2740267	2729772	ct-1	89
2808676	7278237	4469561	ler-0	178
7384379	11712445	4328066	rsch-4	349
11722416	12762202	1039786	hi-0	182
12811779	13352119	540340	mt-0	140
13369626	14114985	745359	wil-2	214
14133003	14594206	461203	no-0	39
15140903	16639017	1498114	can-0	169
16661870	17729413	1067543	wu-0	268
17768356	19381614	1613258	oy-0	177
19409937	22152490	2742553	edi-0	180
22202845	22872106	669261	sf-2	111
22880632	25518863	2638231	wil-2	435
25564689	27337879	1773190	sf-2	161
27365127	30403752	3038625	no-0	149

Table 4.1: A table showing the reconstruction of chromosome 1 for Magic 10

Start bp	End bp	Block length	Founder	Markers in block
57121	4789429	4732308	wu-0	170
4842906	6913268	2070362	oy-0	100
6948380	9573365	2624985	edi-0	156
9706336	16406872	6700536	bur-0	1075
16446134	16903172	457038	kn-0	150
16952065	18115622	1163557	rsch-4	331
18120437	20270219	2149782	ler-0	247
20354685	22663467	2308782	edi-0	262
22749885	23381381	631496	zu-0	170
23401783	24019265	617482	can-0	166
24040932	24449658	408726	mt-0	124
24463552	25420431	956879	rsch-4	165
25435574	26299831	864257	bur-0	119
26360138	30383101	4022963	kn-0	266

Table 4.2: A table showing the reconstruction of chromosome 1 for Magic 814

Start bp	End bp	Block length	Founder	Markers in block
11	6877473	6877462	sf-2	311
6905943	8936302	2030359	mt-0	133
9081412	10081717	1000305	ct-1	76
10140202	10523428	383226	wu-0	43
10598833	12630237	2031404	wil-2	392
12643094	12762202	119108	tsu-0	46
12856640	13232681	376041	can-0	132
13258707	13803993	545286	rsch-4	220
13813714	13975638	161924	ws-0	79
14014764	16569020	2554256	ct-1	255
16579108	17492035	912927	rsch-4	338
17506960	17794522	287562	edi-0	67
17800313	19141222	1340909	col-0	196
19148720	20773134	1624414	edi-0	171
20794575	21763683	969108	wil-2	71
21853484	23258661	1405177	tsu-0	320
23282261	26359810	3077549	wu-0	564
26411809	27836960	1425151	bur-0	114
27897933	30377077	2479144	wil-2	134

Table 4.3: A table showing the reconstruction of chromosome 1 for Magic 423

Start bp	End bp	Block length	Founder	Markers in block
10495	937853	927358	sf-2	42
976088	3645500	2669412	can-0	88
3778301	8446631	4668330	kn-0	253
8497688	10407535	1909847	sf-2	134
10422959	12920073	2497114	col-0	406
12939670	13625001	685331	sf-2	174
13656486	16475822	2819336	zu-0	310
16481902	17029461	547559	tsu-0	199
17058647	17386933	328286	col-0	92
17407677	19834120	2426443	ct-1	377
19861465	21351708	1490243	kn-0	119
21375562	23623548	2247986	tsu-0	423
23630689	24857659	1226970	rsch-4	315
24866083	30416936	5550853	ws-0	458

Table 4.4: A table showing the reconstruction of chromosome 1 for Magic 296

Start bp	End bp	Block length	Founder	Markers in block
706	617398	616692	zu-0	75
639845	2334497	1694652	ws-0	86
2356492	5509270	3152778	bur-0	255
5558788	7706127	2147339	wu-0	220
7741206	8963926	1222720	edi-0	168
9069815	10395740	1325925	ct-1	183
10418538	12224204	1805666	col-0	539
12245703	13500018	1254315	ct-1	707
13504580	13897617	393037	hi-0	242
13912706	15758552	1845846	oy-0	205
15967197	16269831	302634	rsch-4	127
16314489	16799212	484723	wu-0	271
16801304	19253426	2452122	rsch-4	946
19267304	19796111	528807	edi-0	116
19812801	22529264	2716463	zu-0	527
22536083	22636262	100179	mt-0	69
22649406	22949146	299740	sf-2	153
22970054	23597896	627842	zu-0	229
23609738	23848979	239241	wil-2	142
23863884	25351416	1487532	oy-0	557
25353287	26985122	1631835	wil-2	360
27020983	27781259	760276	can-0	107
27830755	30408907	2578152	wil-2	241

Table 4.5: A table showing the reconstruction of chromosome 1 for Magic 51

Conclusion

From the above, it is clear that use of the Viterbi algorithm using a scaled recombination rate to calculate an evolving Transition probability is not a viable possibility for pooled sequence data from an inbred population. To extend this project, applying the algorithm to further data sets is an obvious choice as this would give greater insight into why the model is successful for my simulated data, but fails when applied to real data. However, with more available time, I would compare this method to a different type of HMM algorithm, one based on loss functions, to try to gain further understanding of this model's failings.

The original aim for this chapter was for this method to be extended to cover the reconstruction of the mice datasets. This data poses additional difficulty as the sequenced mice were not inbred and so have a diploid genome. As such, applying my HMM to this data would require allowing for the unknown phase of the data. A possible way to do this would be to treat the data as two interwoven data sets. This could be analysed by creating a nested HMM - a HMM with two levels of hidden information, one for the founder and one for the unknown phase. The transition probabilities for such an HMM could be based on a binomial distribution and the emission probabilities for the phase would all be equal.

Another extension, following on from an improvement in reconstruction quality, would be to use the reconstructions to search for associations.

Chapter 5

Conclusion

The Bayesian Machine Learning Methods for Genome-wide Association Data chapter took many twists and turns throughout the life of this project culminating with the surprising result of the real data analysis of the case/control data agreeing so strongly with the frequentist analysis. On the other hand the result for the QTL analysis was disappointing, if not surprising.

There are many ways in which this work could be furthered and improved. Firstly, given more time, I would recode the method presented here into a more traditional language such as C++. This is due to the fact that, in its current Matlab implementation, it is only viable for use on small data sets, which is not in keeping with the current state for statistical analysis of genetic data, for the obvious reasons that the genome is very large and there are many SNPs which could be associated with any given phenotype. This recoding would be beneficial as it would remove the computational overhead of running Matlab and, also, Matlab suffers with reading in large data files. This recoding would then allow a more thorough investigation into epistasis and give the method access to a wider range of data sets.

In my analysis in this chapter, I compared the function of my method using two different priors, a normal distribution and a Laplace distribution. This could be extended to include a wide range of prior distributions. One distribution which is of interest to me is the Horseshoe prior as detailed in [111]. This prior is appealing as it was developed especially to apply to sparse problems.

As stated at the end of the Computing on the Graphic Processing Unit chapter, Matlab's current GPU methods do not fit well with my MCMC analysis due to the

latters need to be run sequentially. It is clear from the literature review and my own investigations that there is a significant future in statistical research for the use of both Matlab's in built GPU toolbox and GPU enhancements of other coding languages. For an example of languages developed for the scientific community, Mathematica now has GPU utilizing capabilities [112]. In more traditional languages, commercial libraries for Microsoft's .net implementation of C# are being developed, for example, as detailed in [113], CenterSpace has created a GPU Accelerated version of their NMath library.

As far as extensions and further work are concerned for this chapter, there are many ways forward. On a basic level further benchmarking of Matlab's GPU toolbox using a wider range of statistical methods, is an obvious extension. Another way this project could be carried forward is by investigating different languages' implementations of GPU enhancements. This would provide the further benefit of eliminating the computing overheads of running Matlab. This would be the most useful extension when investigating improvements to my MCMC algorithm, as, from the literature review, it is clear that using a GPU enabled version of a traditional programming language has significant advantages over Matlab's implementation.

Lastly, this investigation was run using consumer level graphics cards, which suffer from limited memory. Nvidia produce industrial grade cards which have higher memory, so either waiting for consumer cards to improve or purchasing a higher specification memory card would provide greater power to the investigation, allowing me to use larger matrices and further playing to the strengths of the GPU over the CPU. This may improve the performance of the GPU in the tests.

For the simulations and real data analysis of the pooled data in the Genotyping by Multiplex Pooled Sequencing chapter, we can see that further investigation is needed to see if there is a future in statistical research for using transition probabilities, based on recombination rates, that evolve with the observed states. In simulations they give nice consistent results but when applied to real data the model does not achieve the same level of success.

As with the other chapters there are many ways in which this chapter can been extended and developed further. Firstly, as well as the data for Arabidopsis, my

group also sequenced a selection of mice which were descended from a number of inbred lines. This data presents more challenges than the Arabidopsis data as it has large areas of heterogeneity. This causes a further level of uncertainty in the data as the phase of each observed SNP is also unknown. This could be dealt with by treating the phasing as a second level of hidden states and developing a nested HMM. For this second level of states, the emission probabilities would be equal, whilst the Transition matrix could be based on a binomial distribution.

Another extension could be linked to evaluating the quality of a given Viterbi path. This could be done by looking at the entropy of each section which assigned to an individual founder. The entropy of a section containing x locations can be calculated using the following formula.

$$\text{Entropy}(x) = -\sum_{x} p(x) log(p(x)) \tag{5.1}$$

A lower entropy value for a section coincides with the path through that section being more likely. To calculate these probabilities we would need to use the forward and backward algorithms, so that the probability of being in any state L at any time i can be calculated in the following manner

$$P(i, L) = \Sigma_J \Sigma_K F(i-1, J) T_{i-1}(J, L) B(i+1, K) T_i(K, L) E(i, L) \tag{5.2}$$

where J, K represent the individual states, and F, B, T and E are as described in the Genotyping by Multiplex Pooled Sequencing chapter.

Entropy has been used in [114] to help to stop an HMM from over fitting. Finally the most immediately useful extension to this chapter would be to make use of the reconstructed genomes in association tests. This would allow us to evaluate the performance of the reconstructions and to make best use of the phenotype data that has been gathered for the plants sequenced using this method.

Bibliography

[1] The linear arrangement of the six sex linked factors in drosophila, as shown by their mode of association, A H Sturtevant, Journal of Experimental Zoology, Volume 14, Issue 1 14-43

[2] Introduction to Genetic Analysis 9th edition, A Griffiths et al, W H Freeman and Company, 2008

[3] Experiments in Plant Hybridization, Gregor Mendel 1865

[4] Genetic Mapping in Human Disease, Science, David Altshuler, Mark J. Dalyand Eric S. Lander 2008 November 7; 322(5903): 881-888

[5] Construction of a genetic linkage map in man using restriction fragment length polymorphisms, D Botstein, RL White, M Skolnick, RW Davis,Am J Hum Genet. 1980 May;32(3):314-31

[6] A polymorphic DNA marker genetically linked to Huntington's disease, James F. Gusella et al, 1983, Nature 306, 234 - 238

[7] Linkage of early-onset familial breast cancer to chromosome 17q21, JM Hall, MK Lee, B Newman et al, Science, 250 (1990), pp. 1684-1689

[8] Identification of the breast cancer susceptibility gene BRCA2, R Wooster, G Bignell, J Lancaster et al, Nature, 378 (1995), pp. 789-92

[9] Genetic linkage analysis in familial breast and ovarian cancer: results from 214 families. The Breast Cancer Linkage Consortium, D F Easton et al, Am J Hum Genet. 1993 April; 52(4): 678-701

[10] Genetic linkage studies in Alzheimer's disease families, M.A. Pericak-Vance et al, Experimental Neurology, Volume 102, Issue 3, December 1988, Pages 271-279

[11] Linkage studies in familial Alzheimer disease: Evidence for chromosome 19 linkage, M. A. Pericak-Vance et al, Am J Hum Genet. 1991 June; 48(6): 1034-1050.

[12] A genome-wide search for human type 1 diabetes susceptibility genes, JL Davies, Y Kawaguchi, ST Bennett et al, Nature, 371 (1994), pp. 130-36

[13] Sequential tests for the detection of linkage, NE Morton, Am J Hum Genet, 7 (1955), pp. 277-318

[14] www.ncbi.nlm.nih.gov/sites/entrez?db=omim

[15] The blood groups in relation to peptic ulceration and carcinoma of colon, rectum, breast and bronchus, an association between the ABO groups and peptic ulceration, Ian Aird et al, BMJ August 1954, 315-321

[16] Serum alkaline phosphatase activity is regulated by a chromosomal region containing the alkaline phosphatase 2 gene (Akp2) in C57BL/6J and DBA/2J mice, Foreman JE1,Blizard DA,Gerhard G,Mack HA,Lang DH,Van Nimwegen KL,Vogler GP,Stout JT,Shihabi ZK,Griffith JW,Lakoski JM,McClearn GE,Vandenbergh DJ, Physiol Genomics.2005 Nov 17;23(3):295-303. Epub 2005 Sep 13.

[17] Epistatic interaction between Arabidopsis FRI and FLC flowering time genes generates a latitudinal cline in a life history trait Ana L. Caicedo, John R. Stinchcombe, Kenneth M. Olsen, Johanna Schmitt, and Michael D. Purugganan 15670-15675 PNAS, November 2, 2004, vol. 101 no. 44

[18] Valdar, W., Sabourin, J., Nobel, A. and Holmes, C. C. (2012), Reprioritizing Genetic Associations in Hit Regions Using LASSO-Based Resample Model Averaging. Genet. Epidemiol., 36:451-462. doi:10.1002/gepi.21639

[19] Genome-wide association scan identifies a colorectal cancer susceptibility locus on 11q23 and replicates risk loci at 8q24 and 18q21. Tenesa

A,Farrington SM,Prendergast JG,Porteous ME,Walker M,Haq N,Barnetson RA,Theodoratou E,Cetnarskyj R,Cartwright N,Semple C,Clark AJ,Reid FJ,Smith LA,Kavoussanakis K,Koessler T,Pharoah PD,Buch S,Schafmayer C,Tepel J,Schreiber S,Vlzke H,Schmidt CO,Hampe J,Chang-Claude J,Hoffmeister M,Brenner H,Wilkening S,Canzian F,Capella G,Moreno V,Deary IJ,Starr JM,Tomlinson IP,Kemp Z,Howarth K,Carvajal-Carmona L,Webb E,Broderick P,Vijayakrishnan J,Houlston RS,Rennert G,Ballinger D,Rozek L,Gruber SB,Matsuda K,Kidokoro T,Nakamura Y,Zanke BW,Greenwood CM,Rangrej J,Kustra R,Montpetit A,Hudson TJ,Gallinger S,Campbell H,Dunlop MG Nat Genet. 2008 May;40(5):631-7.

[20] Human genetic variation and its contribution to complex traits Kelly A. Frazer1, Sarah S. Murray1, Nicholas J. Schork1and Eric J. Topol Nature Reviews Genetics10,241-251(April 2009)

[21] AN EPISTATIC GENETIC BASIS FOR FLUCTUATING ASYMMETRY OF MANDIBLE SIZE IN MICE Larry J. Leamy1,Eric J. Routman2and James M. Cheverud3, Evolution, Volume 56,Issue 3,pages 642-653,March 2002

[22] Data and Theory Point to Mainly Additive Genetic Variance for Complex Traits Hill WG,Goddard ME,Visscher PM(2008)Data and Theory Point to Mainly Additive Genetic Variance for Complex Traits.PLoS Genet4(2):e1000008.doi: 10.1371/journal.pgen.1000008

[23] The future of model organisms in human disease research Timothy J. Aitman, Charles Boone, Gary A. Churchill, Michael O. Hengartner, Trudy F. C. Mackayand Derek L. Stemple Nature Reviews Genetics12,575-582(August 2011)

[24] Prioritizing GWAS Results: A Review of Statistical Methods and Recommendations for Their Application Rita M. Cantor, Kenneth Lange, Janet S. Sinsheimer, AJHG, Volume 86, Issue 1, 8 January 2010, Pages 6-22

[25] Assessing the Impact of Transgenerational Epigenetic Variation on Complex Traits Johannes F,Porcher E,Teixeira FK,Saliba-Colombani V,Simon M,et

al.(2009)Assessing the Impact of Transgenerational Epigenetic Variation on Complex Traits.PLoS Genet5(6):e1000530.doi: 10.1371/journal.pgen.1000530

[26] Genetic and Environmental Effects on Complex Traits in Mice William Valdar,Leah C. Solberg, Dominique Gauguier, William O. Cookson,J. Nicholas, P. Rawlins,Richard Mottand Jonathan Flint GeneticsOctober 2006vol. 174no. 2959-98

[27] A model selection approach for the identification of quantitative trait loci in experimental crosses, Karl W. Bromanand Terence P. Speed, J. R.Statist.Soc.B (2002) 64, Part 4, pp. 641-656

[28] Burton, P. R., Tiller, K. J., Gurrin, L. C., Cookson, W. O., Musk, A. W., and Palmer, L. J. (1999). Genetic variance components analysis for binary phenotypes using generalized linear mixed models (GLMMs) and Gibbs sampling.Genetic epidemiology,17(2), 118-140.

[29] A Bayesian method for simultaneously detecting Mendelian and imprinted quantitative trait loci in experimental crosses of outbred species Hayashi T, Awata T. Genetics.2008 Jan;178(1):527-38

[30] Genetics and Analysis of Quantitative Traits, M Lynch and B Walsh, Sinauer Associates 1998

[31] Hereditary Genius, Francis Galton reprinted 1962, Meridian Books

[32] Francis Galton, count and measure, measure and count, 1993 J F Crow, Genetics 135, 1-4

[33] Pearson, Karl. "Mathematical Contributions to the Theory of Evolution. XII.– On a Generalised Theory of Alternative Inheritance, with Special Reference to Mendel's Laws.[Abstract]." Proceedings of the Royal Society of London (1903): 505-509.

[34] The composition of a field of maize.. G H Shull, Rpt AM. Breed. Assoc 4, 296-301

[35] Kreuzungsuntersuchungen an hafer und weizen. H Nilsson-Ehle, Lunds Univ, Arsskrift, n.s, series 2 vol 5, no 2, 1-122.

[36] Johannsen, Wilhelm L. ber Erblichkeit in Populationen und in reinen Linien. Jena: Fischer, 1903.

[37] Bayesian Data analysis, second edition, A Gelman et al, Chapman and Hall, 2004

[38] Variations on a Theme: Cataloging Human DNA Sequence Variation, Francis S. Collins, Mark S. Guyer, Aravinda Chakravarti, Science 28 November 1997: Vol. 278 no. 5343 pp. 1580-1581

[39] The Future of Genetic Studies of Complex Human Diseases, Neil Risch, Kathleen Merikangas, Science 13 September 1996: Vol. 273 no. 5281 pp. 1516-1517

[40] Genome-wide association studies for complex traits: consensus, uncertainty and challenges, Mark I. McCarthy et al, 2008, Nature Reviews Genetics 9, 356-369

[41] Sequence variants in the autophagy gene IRGM and multiple other replicating loci contribute to Crohn's disease susceptibility, M Parkes et al. Nature Genetics 39, 830-32 (2007).

[42] Genetic variants regulating ORMDL3 expression contribute to the risk of childhood asthma, M F Moffatt et al, Nature 448, 470-73 (2007).

[43] Newly identified loci that influence lipid concentrations and risk of coronary artery disease, C J Willeret al. Nature Genetics 40, 161-69 (2008)

[44] Finding the missing heritability of complex diseases, Teri A. Manolio et al, 2009, Nature 461, 747-753

[45] Genome-wide association studies for common diseases and complex traits, Joel N. Hirschhorn and Mark J. Daly, Nature Reviews Genetics 6, 95-108 (2005)

[46] Genome-wide association studies: theoretical and practical concerns, William Y. S. Wang er al, 2005, Nature Reviews Genetics 6, 109-118

[47] Genotype imputation for genome-wide association studies, Jonathan Marchini and Bryan Howie, 2010, Nature Reviews Genetics 11, 499-511

[48] Genetic linkage studies, M Dawn Teare and Jennifer H Barrett, 2005, The Lancet, Volume 366, Issue 9490, Pages 1036-1044

[49] Problems of reporting genetic associations with complex outcomes, Helen M Colhoun, Paul M McKeigue, George Davey Smith, 2003, The Lancet, Volume 361, Issue 9360, Pages 865-872

[50] Comprehensive literature review and statistical considerations for GWAS meta-analysis, Ferdouse Begum et al, Nucleic Acids Research, 2012, Vol. 40, No. 9 3777-3784

[51] Association scan of 14,500 nonsynonymous SNPs in four diseases identifies autoimmunity variants, Paul R Burton et al, 2007, Nature Genetics 39, 1329 - 1337

[52] Hoggart CJ, Whittaker JC, De Iorio M, Balding DJ, 2008 Simultaneous Analysis of All SNPs in Genome-Wide and Re-Sequencing Association Studies. PLoS Genet 4(7): e1000130. doi:10.1371/journal.pgen.1000130

[53] Bayesian LASSO for Quantitative Trait Loci Mapping, Nengjun Yi and Shizhong Xu,Genetics June 2008 vol. 179 no. 2 1045-1055

[54] Joel N. Hirschhorn and Mark J. Daly, 2005, Genome-wide association studies for common diseases and complex traits; Nature Reviews Genetics 6, 95-108 (February 2005)

[55] The Elements of Statistical Learning: Data Mining, Inference and Prediction, T Hastie et al, Second Edition, Springer, 2009.

[56] E. S. Lander and D. Botstein; Mapping Mendelian Factors Underlying Quantitative Traits Using RFLP Linkage Maps; Genetics January 1, 1989 121:185-199

[57] Andrew D Johnson and Christopher J O'Donnell; An Open Access Database of Genome-wide Association Results; BMC Medical Genetics 2009, 10:6

[58] Cochran, WG (1954). Some methods for strengthening the common chi-square tests. Biometrics Vol. 10, No. 4 (Dec., 1954), pp. 417-451

[59] Armitage, P Tests for Linear Trends in Proportions and Frequencies, Biometrics Vol. 11, No. 3 (Sep., 1955), pp. 375-386

[60] Spencer et al 2009 Designing Genome-Wide Association Studies: Sample Size, Power, Imputation, and the Choice of Genotyping Chip. PLoS Genet 5(5): e1000477. doi:10.1371/journal.pgen.1000477

[61] Qing Lu , Yeunjoo Song , Xuefeng Wang , Sungho Won , Yuehua Cui and Robert C Elston, The effect of multiple genetic variants in predicting the risk of type 2 diabetes, BMC Proceedings 2009, 3(Suppl 7):S49

[62] J. I. Weller; Maximum Likelihood Techniques for the Mapping and Analysis of Quantitative Trait Loci with the Aid of Genetic Markers; Biometrics Vol. 42, No. 3 (Sep., 1986), pp. 627-640

[63] Liu Y-Z, Pei Y-F, Liu J-F, Yang F, Guo Y, et al. 2009 Powerful Bivariate Genome-Wide Association Analyses Suggest the SOX6 Gene Influencing Both Obesity and Osteoporosis Phenotypes in Males. PLoS ONE 4(8): e6827. doi:10.1371/journal.pone.0006827

[64] William Valdar, Leah C Solberg, Dominique Gauguier, Stephanie Burnett, Paul Klenerman, William O Cookson, Martin S Taylor, J Nicholas P Rawlins, Richard Mott and Jonathan Flint, Genome-wide genetic association of complex traits in heterogeneous stock mice, Nature Genetics - 38, 879 - 887 (2006)

[65] Mathematical properties of the r2 measure of linkage disequilibrium, Jenna M. VanLiere and Noah A. Rosenberg, Theor Popul Biol. 2008 August; 74(1): 130-137.

[66] Kendall's Advanced Theory of Statistics, Volume 2B Bayesian Inference, Second Edition, Anthony O'Hagen and Jonathon Forster, Arnold publishers 2004

[67] Insights into Colon Cancer Etiology via a Regularized Approach to Gene Set Analysis of GWAS Data, Lin S. Chen, Carolyn M. Hutter, John D. Potter, Yan Liu, Ross L. Prentice, Ulrike Peters, Li Hsu, The American Journal of Human Genetics, Volume 86, Issue 6, 11 June 2010, Pages 860-871

[68] Genome-wide association scan identifies a colorectal cancer susceptibility locus on 11q23 and replicates risk loci at 8q24 and 18q21, A Tenesa et al, Nat Genet. 2008 May;40(5):631-7. doi: 10.1038/ng.133. Epub 2008 Mar 30

[69] Servin B, Stephens M, 2007 Imputation-Based Analysis of Association Studies: Candidate Regions and Quantitative Traits. PLoS Genet 3(7): e114. doi:10.1371/journal.pgen.0030114

[70] A review of Bayesian variable selection methods: what, how and which, R. B. O'Hara and M. J. Sillanpaa, Bayesian Anal. Volume 4, Number 1 (2009), 85-117.

[71] Soonil Kwon , Jinrui Cui , Shannon L Rhodes , Donald Tsiang , Jerome I Rotter and Xiuqing Guo; Application of Bayesian classification with singular value decomposition method in genome-wide association studies; BMC Proceedings 2009, 3(Suppl 7):S9

[72] Metropolis et al Equation of state calculations by fast computing machines,1953 Journal of Chemical Physics 21 (6): 1087-1092

[73] W. K. Hastings; Monte Carlo sampling methods using Markov chains and their applications; Biometrika (1970) 57(1): 97-109 doi:10.1093/biomet/57.1.97

[74] Geman, S.; Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (6): 721-741

[75] Lee SH, van der Werf JHJ, Hayes BJ, Goddard ME, Visscher PM, 2008 Predicting Unobserved Phenotypes for Complex Traits from Whole-Genome SNP Data. PLoS Genet 4(10): e1000231. doi:10.1371/journal.pgen.1000231

[76] Kazuharu Misawa, Shoogo Fujii, Toshimasa Yamazaki, Atsushi Takahashi, Junichi Takasaki et al, New correction algorithms for multiple comparisons in case-control multilocus association studies based on haplotypes and diplotype configurations, Journal of Human Genetics 53, 789-801 doi:10.1007/s10038-008-0312-0

[77] Jonathan K. Pritchard, Matthew Stephens, and Peter Donnelly; Inference of Population Structure Using Multilocus Genotype Data; Genetics June 1, 2000 155:945-959

[78] Uleberg and Meuwissen; Fine mapping of multiple QTL using combined linkage and linkage disequilibrium mapping A comparison of single QTL and multi QTL methods, 2007, GENETICS SELECTION EVOLUTION Volume 39, Number 3, 285, DOI: 10.1186/1297-9686-39-3-285

[79] Jiahan Li, Kiranmoy Das, Guifang Fu, Runze Li, and Rongling Wu; The Bayesian Lasso for genome-wide association studies; Bioinformatics (2011) 27(4): 516-523

[80] A Hierarchical Bayesian Framework for Constructing Sparsity-inducing Priors, Anthony Lee, Francois Caron, Arnaud Doucet, Chris Holmes, arXiv preprint arXiv:1009.1914.

[81] Mendel's Principles of Heredity, W Bateson, 1909, Cambridge University Press, Cambridge

[82] Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans, Heather J. Cordell, Hum. Mol. Genet. (2002) 11 (20): 2463-2468

[83] Role of RAD52 Epistasis Group Genes in Homologous Recombination and Double-Strand Break Repair, Lorraine S. Symington, Microbiol. Mol. Biol. Rev. December 2002 vol. 66 no. 4 630-670

[84] Spontaneous Autoimmune Disease in FcγRIIB-Deficient Mice Results from Strain-Specific Epistasis Immunity, Silvia Bolland, Jeffrey V Ravetch, Volume 13, Issue 2, 1 August 2000, Pages 277-285

[85] The Ubiquitous Nature of Epistasis in Determining Susceptibility to Common Human Diseases, Moore J.H., Hum Hered 2003;56:73-82

[86] Epistasis: too often neglected in complex trait studies?, rjan Carlborg and Chris S. Haley, Nature Reviews Genetics 5, 618-625 (August 2004)

[87] www.nvidia.com

[88] Accelerating epistasis analysis in human genetics with consumer graphics hardware, Nicholas A Sinnott-Armstrong, Casey S Greene, Fabio Cancare and Jason H Moore, BMC Res Notes. 2009 Jul 24;2:149.

[89] High-throughput sequence alignment using Graphics Processing Units, Michael C Schatz, Cole Trapnell, Arthur L Delcher and Amitabh Varshney, BMC Bioinformatics 2007, 8:474

[90] Real-world comparison of CPU and GPU implementations of SNPrank: a network analysis tool for GWAS, Nicholas A. Davis, Ahwan Pandey, and B. A. McKinney, Bioinformatics (2011) 27(2): 284-285

[91] GPU accelerated QTL detection, G. Chapuis, O. Filangi, J.M. Elsen , D. Lavenier , P. Le Roy (submitted to Journal of computational biology May 25, 2012)

[92] Acceleration Genotype Imputation for Large Dataset on GPU, Hu Xiaohan, Procedia Environmental Sciences Volume 8, 2011, Pages 457-463

[93] Mendel-GPU: Haplotyping and genotype imputation on Graphics Processing Units, Gary K. Chen, Kai Wang, Alex H. Stram, Eric M. Sobel, and Kenneth Lange, Bioinformatics first published online September 5, 2012 doi:10.1093/bioinformatics/bts536

[94] A GPU-Based Implementation of Differential Evolution for Solving the Gene Regulatory Network Model Inference Problem. Luis E. Ramrez-Chavez, Carlos A. Coello Coello and Eduardo Rodriguez-Tello, Proceedings of the WPABA 2011, Galveston Island, TX, USA, pp. 21-30, Published by the Universidad Complutense de Madrid, October 2011

[95] On the Utility of Graphics Cards to Perform Massively Parallel Simulation of Advanced Monte Carlo Methods, Anthony Lee, Christopher Yau, Michael B. Giles, Arnaud Doucet and Christopher C. Holmes, Journal of Computational and Graphical Statistics, Volume 19, Issue 4, 2010, pages 769-789

[96] Andolfatto et al, Multiplexed shotgun genotyping for rapid and efficient genetic mapping, Genome Res. 2011. 21: 610-617

[97] Kover PX, Valdar W, Trakalo J, Scarcelli N, Ehrenreich IM, et al. 2009 A Multiparent Advanced Generation Inter-Cross to Fine-Map Quantitative Traits in Arabidopsis thaliana. PLoS Genet 5(7): e1000551. doi:10.1371/journal.pgen.1000551

[98] Gan et al Multiple reference genomes and transcriptomes for Arabidopsis thaliana, 2011 Nature 477, 419-423

[99] Yalcin B, Nicod J, Bhomra A, Davidson S, Cleak J, et al. 2010 Commercially Available Outbred Mice for Genome-Wide Association Studies. PLoS Genet 6(9): e1001085. doi:10.1371/journal.pgen.1001085

[100] Biological sequence analysis, R Durbin et al, 13th printing, 2009

[101] Effect of genetic divergence in identifying ancestral origin using HAPAA, A Sundquist et al, Genome Res. 18, 676-682 (2008).

[102] Statistical Inference for Probabilistic Functions of Finite State Markov Chains, L E Baum and T Petrie, 1966, The Annals of Mathematical Statistics 37 (6): 1554-1563

[103] Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, A J Viterbi, Information Theory, IEEE Transactions on , vol.13, no.2, pp.260,269, April 1967

[104] A method for fine mapping quantitative trait loci in outbred animal stocks, R Mott et al, Proc. Natl. Acad. Sci. USA 97, 12649-12654 (2000)

[105] PennCNV: An integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data, Kai Wang et al, Genome Res. 2007. 17: 1665-1674

[106] Systematic assessment of copy number variant detection via genome-wide SNP genotyping, Gregory M Cooper et al, Nature Genetics 40, 1199 - 1203 (2008)

[107] CpG islands as gene markers in the vertebrate nucleus, Adrian P. Bird, Trends in Genetics, Volume 3, 1987, Pages 342-347

[108] A species-generalized probabilistic model-based definition of CpG islands, Rafael A. Irizarry, Hao Wu, Andrew P. Feinberg, Mammalian Genome, October 2009, Volume 20, Issue 9-10, pp 674-680

[109] Multiple alignment using hidden Markov models, Sean R. Eddy, ISMB-95 Proceedings, 1995, 114-120

[110] Profile hidden Markov models, S R Eddy, Bioinformatics (1998) 14 (9): 755-763

[111] The horseshoe estimator for sparse signals, Carlos M. Carvalho, Nicolas G. Polson and James G. Scott, Biometrika (2010), 97, 2, pp. 465-480

[112] http://reference.wolfram.com/mathematica/guide/GPUComputing.html

[113] http://www.centerspace.net/doc/NMath/whitepapers/
NMath.Premium.Benchmarks.pdf

[114] Towards a Maximum Entropy Method for Estimating HMM Parameters, Walder, Christian J., Kootsookos, Peter J. and Lovell, Brian C. (2003). . In: Lovell, Brian C. and Maeder, Anthony J, Proceedings of the 2003 APRS Workshop on Digital Image Computing. Workshop on Digital Image Computing, Brisbane, (45-49). February 7, 2003.