MONTE CARLO MARKOV CHAIN EXACT INFERENCE FOR BINOMIAL REGRESSION MODELS

by

David Zamar 2006

B.Sc. in Computer Science, University of British Columbia, 2004

A Project submitted in partial fulfillment

of the requirements for the degree of

Master of Science

in the School

of

Statistics and Actuarial Science

SIMON FRASER UNIVERSITY

Summer 2006

APPROVAL

Name: David Zamar 2006

Degree: Master of Science

Title of Project: Monte Carlo Markov Chain Exact Inference for Binomial Regression Models

Examining Committee: Dr. Charmaine Dean
Chair

Dr. Jinko Graham
Senior Supervisor
Simon Fraser University

Dr. Brad McNeney
Senior Supervisor
Simon Fraser University

Dr. Tim Swartz
External Examiner
Simon Fraser University

Date Approved: _____

Abstract

Current methods for conducting exact inference for logistic regression are not capable of handling large data sets due to memory constraints caused by storing large networks. We provide and implement an algorithm which is capable of conducting (approximate) exact inference for large data sets. Various application fields, such as genetic epidemiology, in which logistic regression models are fit to larger data sets that are sparse or unbalanced may benefit from this work. We illustrate our method by applying it to a diabetes data set which could not be analyzed using existing methods implemented in software packages such as LogXact and SAS. We include a listing of our code along with documented instructions and examples of all user methods. The code will be submitted to the Comprehensive R Archive Network as a freely-available R package after further testing.

Keywords: conditional inference, exact test, logistic regression, Markov chain Monte Carlo, Metroplis-Hastings algorithm

Acknowledgements

The following thesis, although an individual work, was made possible by the insights and direction of several people. I will most likely not be able to include everyone or thank each one of them enough.

I would like to begin by thanking Jinko Graham and Brad McNeney, my educational supervisors, who guided me, helped me reach for high standards and provided timely and instructive comments throughout every stage of the thesis process.

My research was encouraged and made possible by the Graduate Committee, the Department of Statistics and Actuarial Science, and IRMACS at Simon Fraser University. Thanks also to Tim Swartz for his careful reading of my thesis and useful comments.

My fellow graduate students were also helpful and kind. Special thanks to Linnea, Kelly, Jean, Zhijian, Sigal, Matthew and Gurbakhshash.

I am most of all very grateful to my family who have always provided me with encouragement, support and unconditional love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Why Exact Inference?

Asymptotic inference is often unreliable when modelling rare events, or dealing with data sets that are small, or data sets that are imbalanced or sparse. Typically, statistical inference for logistic regression models involves large sample approximations based on the likelihood function. Unfortunately, asymptotic methods become inadequate when sample sizes are small or the data are sparse or skewed. With sparse data, one or more of the parameter estimates may lie on or near the boundary of the parameter space. In other words, the estimated probabilities are close to 0 (or 1) and resulting logit coefficients are effectively minus (or plus) infinity. Conceptually there is insufficient information in the data to distinguish whether the true parameter value is in the interior or on the boundary of the parameter space. This uncertainty is reflected in the large standard errors for the parameter estimates. When the true parameter values lie on the boundary of the parameter space, the large-sample theory is invalid; when they lie near the boundary, the large-sample theory is unreliable unless the sample size is very large. If prior distributions for the parameters can be specified, a Bayesian perspective may be adopted allowing exact inference to be based on the posterior distributions of the parameters given the observed data. Cox (1970) proposed a frequentist alternative to large-sample inference which utilizes the exact distribution of the sufficient statistic for the parameters of interest, conditional on the sufficient statistics for the other "nuisance" parameters in the model. Exact logistic regression refers to exact

conditional inference for binomial data modelled by a logistic regression and is reliable no matter how small or imbalanced the data set. As sample size grows and/or the data become better balanced and less sparse, the solution obtained using the conventional large sample approach will coincide with the solution obtained by the exact approach.

1.2 Binomial Response Model

In logistic regression, the binary response variable is modelled as a binomial random variable with the logit link function. Let Y_i be the response of the i^{th} subject with

$$Y_i \sim binomial(m_i, p_i)$$

and

$$logit\,(p_i) = \mathbf{w}_i^T\beta + \mathbf{z}_i^T\gamma, \quad i = 1, \ldots, n.$$

where, β is a vector of nuisance parameters corresponding to the first p explanatory variables $(w_{i1,}w_{i2}, \ldots, w_{ip})^T$ and γ is a vector of parameters corresponding to the last q explanatory variables $(z_{i1,}z_{i2}, \ldots, z_{iq})^T$on subject i. We are not interested in making inferences on β; however, including the $\mathbf{w}_i$'s in the model reduces noise and provides better inference on the regression parameters, γ, of interest. Ultimately, we are interested in studying the relationship between $\mathbf{p} = (p_1, ..., p_n)$ and $\mathbf{z}$. For example, we might want to compare

$$H_0 : logit\,(\mathbf{p}) = \mathbf{W}\beta \qquad (\gamma = \mathbf{0},\ \beta\ arbitrary)$$

vs.

$$H_1 : logit\,(\mathbf{p}) = \mathbf{W}\beta + \mathbf{Z}\gamma \qquad (\gamma \neq \mathbf{0},\ \beta\ arbitrary),$$

where $\mathbf{W} = \begin{pmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_n^T \end{pmatrix}$ and $\mathbf{Z} = \begin{pmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_n^T \end{pmatrix}$ are $n \times p$ and $q \times p$ dimensional matrices respectively.

1.3 Sufficient Statistics for β and γ

Exact conditional inference is based on the distribution of the sufficient statistic $\mathbf{T}$ for the parameters of interest, γ, given a sufficient statistic $\mathbf{S}$ for the nuisance parameters β. Equivalently, inference is based on the conditional distribution of $\mathbf{Y}$ given $\mathbf{S}$. The conditional distribution of $\mathbf{Y}$ given $\mathbf{S}$ does not depend on β since we are conditioning on the sufficient statistic for β. The joint density of $\mathbf{Y}$ is

$$\begin{aligned} f(\mathbf{y}|\beta,\gamma) &= \prod_{i=1}^{n}\binom{m_i}{y_i}\exp\left\{y_i\log(p_i)+(m_i-y_i)\log(1-p_i)\right\} \\ &= \left[\prod_{i=1}^{n}\binom{m_i}{y_i}\right]\exp\left\{\sum_i y_i\log\left(\frac{p_i}{1-p_i}\right)+\sum_i m_i\log(1-p_i)\right\} \\ &= \left[\prod_{i=1}^{n}\binom{m_i}{y_i}\right]\exp\left\{\sum_i y_i\log\left(\frac{p_i(\beta,\gamma)}{1-p_i(\beta,\gamma)}\right)\right\}\exp\left\{\sum_i m_i\log\left[1-p_i(\beta,\gamma)\right]\right\} \\ &= \left[\prod_{i=1}^{n}\binom{m_i}{y_i}\right]\exp\left\{\sum_i y_i\mathbf{w}_i^T\beta+\sum_i y_i\mathbf{z}_i^T\gamma\right\}\exp\left\{h(\beta,\gamma)\right\}. \end{aligned}$$

where

$$h(\beta,\gamma)=\sum_i m_i\log\left[1-p_i(\beta,\gamma)\right].$$

Notice that,

$$\sum_i y_i\mathbf{w}_i^T\beta=(y_1,\ldots,y_n)\begin{pmatrix}\mathbf{w}_1^T\beta\\ \vdots\\ \mathbf{w}_n^T\beta\end{pmatrix}=(y_1,\ldots,y_n)\begin{pmatrix}\mathbf{w}_1^T\\ \vdots\\ \mathbf{w}_n^T\end{pmatrix}\beta=\mathbf{y}^T\mathbf{W}\beta=\beta^T\mathbf{W}^T\mathbf{y}$$

and similarly,

$$\sum_i y_i\mathbf{z}_i^T\gamma=\gamma^T\mathbf{Z}^T\mathbf{y}.$$

By the factorization theorem, $\mathbf{T} = \mathbf{Z}^T\mathbf{y}$ is sufficient for γ and $\mathbf{S} = \mathbf{W}^T\mathbf{y}$ is sufficient for β. Suppose $\mathbf{S} = \mathbf{W}^T\mathbf{y} = \mathbf{s}$. Let $C(\mathbf{s}) = \{\mathbf{y}^* : S(\mathbf{y}^*) = \mathbf{s}\} = \{\mathbf{y}^* : \mathbf{W}^T\mathbf{y}^* = \mathbf{s}\}$. Then,

$$
\begin{aligned}
f(\mathbf{y}|\mathbf{S}=\mathbf{s}) &= \frac{\left[\prod_{i=1}^{n}\binom{m_i}{y_i}\right]\exp\left\{\beta^T\mathbf{W}^T\mathbf{y}+\gamma^T\mathbf{Z}^T\mathbf{y}\right\}\exp\{h(\beta,\gamma)\}}{\exp\{h(\beta,\gamma)\}\sum_{\mathbf{y}^*\in C(s)}\left[\prod_{i=1}^{n}\binom{m_i}{y_i^*}\right]\exp\{\beta^T\mathbf{W}^T\mathbf{y}^*+\gamma^T\mathbf{Z}^T\mathbf{y}^*\}} \\
&= \frac{\left[\prod_{i=1}^{n}\binom{m_i}{y_i}\right]\exp\left\{\beta^T\mathbf{s}\right\}\exp\left\{\gamma^T\mathbf{Z}^T\mathbf{y}\right\}}{\sum_{\mathbf{y}^*\in C(s)}\left[\prod_{i=1}^{n}\binom{m_i}{y_i^*}\right]\exp\{\beta^T\mathbf{s}\}\exp\{\gamma^T\mathbf{Z}^T\mathbf{y}^*\}} \\
&= \frac{\left[\prod_{i=1}^{n}\binom{m_i}{y_i}\right]\exp\left\{\gamma^T\mathbf{Z}^T\mathbf{y}\right\}}{\sum_{\mathbf{y}^*\in C(s)}\left[\prod_{i=1}^{n}\binom{m_i}{y_i^*}\right]\exp\{\gamma^T\mathbf{Z}^T\mathbf{y}^*\}} \propto \left[\prod_{i=1}^{n}\binom{m_i}{y_i}\right]\exp\left\{\gamma^T\mathbf{Z}^T\mathbf{y}\right\} \quad (1.1)
\end{aligned}
$$

In order to make exact (i.e. small-sample) inference for γ based on $f(\mathbf{y}|\mathbf{S}=\mathbf{s})$, we need to be able to evaluate this distribution. Approximate exact inference for γ is based on an estimate of $f(\mathbf{y}|\mathbf{S}=\mathbf{s})$ obtained by sampling from the distribution. However the computation of the proportionality constant is infeasible for most applications, because it requires enumeration of the support $C(\mathbf{s})$ of the conditional distribution $f(\mathbf{y}|\mathbf{S}=\mathbf{s})$, which is not always attainable. Fortunately, Markov Chain Monte Carlo (MCMC) approaches only require knowledge of $f(\mathbf{y}|\mathbf{S}=\mathbf{s})$ up to a proportionality constant, which is exactly what we have.

1.4 Rationale for Undertaking this project

Exact inference for logistic regression is based on generating variates from the conditional distribution of the sufficient statistics for the regression parameters of interest given the sufficient statistics for the remaining nuisance parameters. A recursive algorithm for generating the required conditional distribution is implemented in the software LogXact [10].

However, the algorithm can only handle problems with modest samples sizes and numbers of covariates [3]. To increase the size of problem that can be analyzed, Mehta et al. (2000) developed a Monte Carlo method for (approximate) exact inference and implemented it in LogXact. Their method represents the support of the conditional distribution of the binomial response given the sufficient statistics for the nuisance parameter by a network of arcs and nodes. The network consists of n layers (or levels) of nodes, where n represents the length of the response vector. The nodes at the i^{th} level of the network correspond to the possible values of the sufficient statistics for the parameters of interest based on the first i observations in the response vector. The arcs connecting nodes in level $i-1$ to nodes in level i of the network represent possible values of the i^{th} component of the response vector. Arcs are placed between nodes at each level so that a path through the network corresponds to a unique response vector consistent with the observed values of the sufficient statistics for the nuisance parameters. To avoid traversing all possible paths through the network, they describe how to randomly sample paths with the appropriate probabilities. The limiting factor for their Monte Carlo approach is the size of the network, which must be stored in memory. Forster, McDonald, and Smith (1996) attempted to circumvent this difficulty by developing a Gibbs sampler to generate the Monte Carlo samples. A Gibbs sampler would sample from the conditional distribution of a particular sufficient statistic given the observed values of the sufficient statistics for the nuisance parameters *and* the current values of the sufficient statistics for the remaining parameters of interest. In exact conditional inference for logistic regression, conditioning on too many sufficient statistics can greatly restrict the set of support points for the conditional distribution, making the distribution highly discrete or even degenerate. This "overconditioning" problem is particularly acute when conditioning on sufficient statistics associated with continuous covariates in the logistic regression model. In the context of Gibb's sampling, such degeneracy due to overconditioning could lead to poor mixing or even a degenerate conditional distribution for the complete vector of sufficient statistics of interest. This view is supported by the general observation that the Gibb's sampling approach of Forster, McDonald, and Smith (1996) is only effective for logistic regression problems with equally spaced covariate values and moderately large covariate groups [12]. Our experience with the Gibb's sampling method,

as implemented in LogXact, also supports this view. As described in Chapter 3, we obtained a degenerate conditional distribution for the complete vector of sufficient statistics when we tried to apply the approach to our data. For "large" problems, in which storage of the network is not possible and the Gibbs sampler proves unreliable, Forster et al. (2003) propose an alternative method that makes use of the Metropolis-Hastings algorithm. The focus of this project is to both improve and implement their alternative approach for large logistic regression problems.

Chapter 2

Methods

2.1 Review of Markov Chains

Before introducing the Metropolis-Hastings algorithm a brief review of Markov chains, adapted from [15], is necessary. The sequence $\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_n$ is a Markov chain if the transition probability between any two different values in the state space depends only on the current value of the Markov chain. Thus, in a Markov chain the only information required to generate the value of the next state is the value of the current state; knowledge of the values of earlier states do not change the transition probabilities. Throughout, we consider Markov chains with a finite discrete state space $\mathbf{\Psi} = \{\psi_1, \psi_2, ..., \psi_k\}$. In a Markov chain, the probability of moving from state ψ_i at time t to state ψ_j at time $t+1$ is

$$P(\psi_i \rightarrow \psi_j) = \Pr(\mathbf{Y}_{t+1} = \psi_j \mid \mathbf{Y}_t = \psi_i).$$

This conditional probability is called the *transition kernel*. The properties of a Markov chain depend heavily on the chosen transition kernel.

Let $\pi_t(\psi)$ denote the probability that the chain is in state ψ at time t, then $\pi_{t+1}(\psi)$ is obtained by summing over the probability of making a transition from any state ψ_i (at

time t) to state ψ (at time $t+1$):

$$\begin{aligned}\pi_{t+1}(\psi) &= \Pr(\mathbf{Y}_{t+1} = \psi) \\ &= \sum_{j=1}^{k} \Pr(\mathbf{Y}_{t+1} = \psi \mid \mathbf{Y}_t = \psi_j) \Pr(\mathbf{Y}_t = \psi_j) \\ &= \sum_{j=1}^{k} P(\psi_j \to \psi)\, \pi_t(\psi_j)\end{aligned}$$

A chain is **irreducible** if every state is accessible from every other state. A chain is said to be **aperiodic** when the number of transitions needed to move between any two states is not required to be a multiple of some integer greater than one. Aperiodicity prevents the chain from having a cycle of fixed length between any two states. A chain with a finite state-space is **ergodic** if it is both irreducible and aperiodic.

A Markov chain may reach a stationary distribution π, where the probability of being in any particular state is independent of the initial condition. A Markov chain is said to have reached a **stationary distribution** (or steady state) by time t if $\pi_{t+1}(\psi) = \pi(\psi)$ for all $\psi \in \mathbf{\Psi}$. Once a stationary distribution is reached, all subsequent realizations of the chain are from the distribution. It can be shown that if a Markov chain is ergodic, then it is guaranteed to reach a unique stationary distribution π.

The **Ergodic Theorem** states that if $\mathbf{Y}_0, \mathbf{Y}_1, ...$ are realizations from an ergodic Markov chain, the mean value of any function, h, under the stationary distribution of the Markov chain can be obtained by taking the average of $h(\mathbf{Y_i})$ over an arbitrarily large number, n, of realizations $\mathbf{Y}_0, \mathbf{Y}_1, ..., \mathbf{Y}_n$ of that Markov chain. The Ergodic Theorem can be viewed as a Markov chain law of large numbers.

2.2 The Metropolis-Hastings Algorithm

Suppose that we are interested in calculating

$$\theta = E[h(\mathbf{Y})] = \sum h(\mathbf{y})\, \pi(\mathbf{y})$$

for a random vector $\mathbf{Y}$ with density $\pi(\mathbf{y})$ and a specified function $h(\mathbf{y})$. If we have a sequence $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n$ of iid realizations from the density $\pi(\mathbf{y})$, then

$$\hat{\theta} = \frac{1}{n} \sum h(\mathbf{y}_i) \tag{2.1}$$

is a consistent estimate of θ.

There are occasions when it is not possible to generate independent random vectors with density π. This is particularly the case when π is only known up to a multiplicative constant (i.e., we can write $\pi(\mathbf{y}) = C \cdot g(\mathbf{y})$ where $g(\mathbf{y})$ is ascertainable but C is not). An alternative procedure is to generate a Markov chain sequence $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n$ with stationary density $\pi(\mathbf{y})$. If the Markov chain is ergodic, then, by the Ergodic Theorem, (2.1) is still a consistent estimate of θ [15].

The Metropolis-Hastings algorithm constructs a Markov chain with a stationary distribution equal to π. To ensure that π is a stationary distribution, it is sufficient for π and the transition kernel to satisfy the detailed-balance equation

$$\pi(\mathbf{y}) P\left(\mathbf{y} \rightarrow \mathbf{y}'\right) = \pi\left(\mathbf{y}'\right) P\left(\mathbf{y}' \rightarrow \mathbf{y}\right). \tag{2.2}$$

However, the chain must be ergodic for π to be a unique stationary distribution and the Ergodic Theorem to apply. The algorithm requires, as input, a mechanism that allows one to move from one state to another. This mechanism is called the *proposal distribution*. The proposal distribution, which we will denote as $q(\mathbf{y}^*|\mathbf{y})$, should be easy to generate from. In the algorithm, a transition from one state to another is done as follows. Suppose that the current state of the Markov chain is $\mathbf{y}$. Then the next state of the Markov chain, $\mathbf{y}'$, is obtained by generating a "proposal" random vector $\mathbf{y}^* \sim q$ and an independent uniform random variable u on the interval $(0, 1)$ and setting

$$\mathbf{y}' = \begin{cases} \mathbf{y}^* & u \leq min\left\{\frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})}, 1\right\} \\ \\ \mathbf{y} & u > min\left\{\frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})}, 1\right\} \end{cases}. \tag{2.3}$$

We refer to $min\left\{\frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})}, 1\right\}$ as the acceptance probability and denote it by $\alpha(y, y^*)$. In order to implement the Metropolis-Hastings algorithm we must compute

$$\frac{\pi(\mathbf{y}^*)}{\pi(\mathbf{y})}.$$

In the case where

$$\pi(\mathbf{y}) = C \cdot g(\mathbf{y})$$

with $g(\mathbf{y})$ known but C unknown,

$$\frac{\pi(\mathbf{y}^*)}{\pi(\mathbf{y})} = \frac{C \cdot g(\mathbf{y}^*)}{C \cdot g(\mathbf{y})} = \frac{g(\mathbf{y}^*)}{g(\mathbf{y})}$$

can be calculated using the known values of $g(\mathbf{y}^*)$ and $g(\mathbf{y})$ alone. Hence, the algorithm only requires knowledge of the target distribution π up to a constant of proportionality.

In order for the Metropolis-Hastings algorithm to produce a chain with π as a stationary distribution, the detailed-balance condition (2.2) should hold. To verify this, it suffices to restrict attention to the case $\mathbf{y} \neq \mathbf{y}'$ because condition (2.2) trivially holds when $\mathbf{y} = \mathbf{y}'$. For $\mathbf{y} \neq \mathbf{y}'$, the probability of moving from state $\mathbf{y}$ to state $\mathbf{y}' = \mathbf{y}^*$ is

$$P(\mathbf{y} \to \mathbf{y}^*) = q(\mathbf{y}^*|\mathbf{y})\,\alpha(\mathbf{y}, \mathbf{y}^*) = q(\mathbf{y}^*|\mathbf{y}) \min\left\{\frac{\pi(\mathbf{y}^*)\,q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})\,q(y^*|\mathbf{y})}, 1\right\}$$

$$= \begin{cases} q(\mathbf{y}|\mathbf{y}^*)\,\frac{\pi(\mathbf{y}^*)}{\pi(\mathbf{y})} & \text{if } \frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(y^*|\mathbf{y})} \leq 1 \\ q(\mathbf{y}^*|\mathbf{y}) & \text{if } \frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})} \geq 1 \end{cases}$$

Therefore,

$$\pi(\mathbf{y})\,P(\mathbf{y} \to \mathbf{y}^*) = \begin{cases} \pi(\mathbf{y}^*)\,q(\mathbf{y}|\mathbf{y}^*) & \frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})} \leq 1 \\ \pi(\mathbf{y})\,q(\mathbf{y}^*|\mathbf{y}) & \frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})} \geq 1 \end{cases} \tag{2.4}$$

Notice that when $\alpha(\mathbf{y}, \mathbf{y}^*) = \min\left\{\frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(y^*|\mathbf{y})}, 1\right\} = 1$, so that $\pi(\mathbf{y}^*)\,q(\mathbf{y}|\mathbf{y}^*) = \pi(\mathbf{y})\,q(\mathbf{y}^*|\mathbf{y})$, either one of these terms in equation (2.4) can be used interchangeably and so there is no ambiguity. Interchanging $\mathbf{y}$ and $\mathbf{y}^*$ (2.4) we have

$$\pi(\mathbf{y}^*)\,P(\mathbf{y}^* \to \mathbf{y}) = \begin{cases} \pi(\mathbf{y})\,q(\mathbf{y}^*|\mathbf{y}) & \frac{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})}{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)} \leq 1 \\ \pi(\mathbf{y}^*)\,q(\mathbf{y}|\mathbf{y}^*) & \frac{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})}{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)} \geq 1 \end{cases}$$

$$= \begin{cases} \pi(\mathbf{y})\,q(\mathbf{y}^*|\mathbf{y}) & \frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})} \geq 1 \\ \pi(\mathbf{y}^*)\,q(\mathbf{y}|\mathbf{y}^*) & \frac{\pi(\mathbf{y}^*)q(\mathbf{y}|\mathbf{y}^*)}{\pi(\mathbf{y})q(\mathbf{y}^*|\mathbf{y})} \leq 1 \end{cases}$$

$$= \pi(\mathbf{y})\,P(\mathbf{y} \to \mathbf{y}^*).$$

Hence, the detailed-balance condition holds and so the target distribution $\pi(\mathbf{y})$ is a stationary distribution of the Markov chain.

2.3 Algorithm Proposed by Forster et al. (2003)

Let

$$\pi(\mathbf{y}) = P(\mathbf{Y} = \mathbf{y})$$

denote the stationary probability of state $\mathbf{y}$ in a Markov chain. In our case, the stationary probabilities $\pi(\mathbf{y})$ correspond to (1.1); i.e.,

$$\pi(\mathbf{y}) \propto \left[\prod_{i=1}^{n} \binom{m_i}{y_i}\right] \exp\left\{\gamma^T \mathbf{Z}^T \mathbf{y}\right\}, \text{ for } \mathbf{y} \in \mathbf{C}(\mathbf{s}).$$

Therefore, if we use a Metropolis-Hastings algorithm to construct a Markov chain with stationary distribution (1.1) and this chain has the additional property of being ergodic, we may extract dependent samples whose distributions converge to the required conditional distribution.

The Metropolis-Hastings algorithm proposed by Forster et al. (2003) involves generating proposals y^* of the form $\mathbf{y}^* = \mathbf{y} + d \cdot \mathbf{v}$, where $\mathbf{v}$ is a vector of coprime integers such that

$$\mathbf{W}^T \mathbf{v} = 0, \tag{2.5}$$

$\sum_{i=1}^{n} |v_i| \leq r$ for a given even integer r and d is an integer such that $0 \leq y_i + dv_i \leq m_i$ for $i = 1, ..., n$. The proposal distribution $q(\mathbf{y}^*|\mathbf{y})$ is chosen so that the acceptance probabilities are identically 1, as shown below. Hence, from now on, we do not distinguish between the proposal $\mathbf{y}^*$ and the next state $\mathbf{y}'$ of the Markov chain. Since $\mathbf{y}^* = \mathbf{y} + d\mathbf{v}$, where $\mathbf{W}^T\mathbf{v} = \mathbf{0}$, the sufficient statistics for the nuisance parameters are maintained. To see why, suppose the Markov chain is currently on its j^{th} iteration and denote the corresponding value of the response vector by $\mathbf{y}^{(j)}$, then

$$\mathbf{W}^T\mathbf{y}^{(j+1)} = W^T\left(\mathbf{y}^{(j)} + d \cdot \mathbf{v}\right) = \mathbf{W}^T\mathbf{y}^{(j)} + d \cdot \mathbf{W}^T\mathbf{v} = \mathbf{W}^T\mathbf{y}^{(j)},$$

where $\mathbf{y}^{(j+1)}$ denotes the value of the response vector for the $(j+1)\,st$ iteration of the Markov chain.

So far, we have given a *recipe* to move from one $\mathbf{y}$ to another $\mathbf{y}^*$ where $\mathbf{y}^*$ is guaranteed to lie under the support of the required distribution. All we need now is to figure out a way to make the stationary probabilities conform to (1.1). The proposal distribution involves sampling of $\mathbf{v}$ and d given $\mathbf{y}$. Sampling proceeds in two steps. In the first step, $\mathbf{v}$ is sampled unconditionally, without regard to $\mathbf{y}$. Then, in the second step, d is sampled conditionally on the realized value of $\mathbf{v}$ and $\mathbf{y}$. In the first step, $\mathbf{v}$ is uniformly selected from among the vectors with coprime elements (i.e., greatest common divisor of any pair of elements is 1) that satisfy (2.5) and the additional constraint that

$$\sum_{i=1}^{n} |v_i| \leq r \tag{2.6}$$

for a given r, chosen so that the enumeration of such vectors is feasible. Usually the vector of ones is in the column space of $\mathbf{W}$, as a constant (intercept) term is included in the linear model, and so

$$\sum_{i=1}^{n} v_i = 0; \tag{2.7}$$

therefore, $\sum_{i=1}^{n} |v_i|$ in (2.6) must be even. Large values of r would be desirable as they allow for larger transitions in the Markov chain and a potentially quicker exploration of the conditional distribution of $\mathbf{Y}$ given $\mathbf{S} = \mathbf{W}^T\mathbf{Y}$. On the other hand, since the size of the set

$$\mathbf{V} = \left\{ \mathbf{v} : \sum_{i=1}^{n} |v_i| \leq r \text{ and } v_i \text{ coprime for } i = 1, ..., n,\ \mathbf{W}^T\mathbf{v} = \mathbf{0} \right\}$$

increases with r, smaller values of r will help to keep the size within feasible limits. Additionally, the chosen value of r affects the second-stage sampling of d conditional on the realized value of $\mathbf{v}$ and $\mathbf{y}$. Large values of r will increase the chance that $d = 0$ is the only integer satisfying the constraints (2.9). If $d = 0$ with high probability the Markov chain will mix poorly as this represents a "transition" to the current state. Forster et al. (2003) suggest choosing r to be $4, 6$, or 8. Small values of r correspond to more transitions to new states, but the Markov chain may remain trapped in a local neighborhood.

To illustrate enumeration of $\mathbf{V}$, consider $r = 2$ and

$$\mathbf{W}^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

There are (3 choose 2) = 6 elements of $\mathbf{V}$: $v1 = (0, 1, -1, 0, 0)$, $v2 = (0, -1, 1, 0, 0)$, $v3 = (0, 1, 0, -1, 0)$, $v4 = (0, -1, 0, 1, 0)$, $v5 = (0, 0, 1, -1, 0)$, and $v6 = (0, 0, -1, 1, 0)$. In general, one can never find a $\mathbf{W}$ such that the only vector $\mathbf{v}$ satisfying $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ is the vector of zeroes. To see this, note that $\mathbf{W}$ is an $n \times q$ matrix specifying the columns of the design matrix for the nuisance parameters, where $q < n$. The set of all $\mathbf{v}$ such that $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ is the null space of $\mathbf{W}$ and contains the set $\mathbf{V}$ as well as the vector of zeroes. Suppose the null space of $\mathbf{W}$ is composed of the zero vector only (so that $\mathbf{V}$ must be an empty set). Then the only vector in $\mathcal{R}^n$ that is orthogonal to the column space of $\mathbf{W}$ is the zero vector; i.e. $\mathbf{W}$ is of rank n, which contradicts $q < n$. That said, it is still possible for the set $\mathbf{V}$ to be empty for certain values of r. For example, take

$$\mathbf{W}^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 8 & 15 & 0 \end{pmatrix}$$

and $r = 2$. The possible vectors in the set $\mathbf{V}$ are all permutations of (-1 1 0 0). Hence, in order that $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ for some $\mathbf{v} \in \mathbf{V}$, each row of $\mathbf{W}^T$ must have at least two entries of equal value. As this does not hold for this $\mathbf{W}^T$, $\mathbf{V}$ is empty. It can also be shown that $\mathbf{V}$ is empty for $r = 4$ and 6. However, if we take $r = 8$, the vector $\mathbf{v} = (-4, 1, 0, 3)$ satisfies $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ and the other restrictions for $\mathbf{V}$, and so $\mathbf{V}$ is non-empty in this case. In fact, Forster et al. (2003) state that for large enough r their Markov chain is irreducible, which implies there are enough directions (i.e. vectors) in $\mathbf{V}$ to move the Markov chain from

any one point in the state space to any other. Assuming that the conditional distribution of interest is non-degenerate, so that the state space of their Markov chain contains more than one point, this irreducibility implies that $\mathbf{V}$ must be non-empty. When $\mathbf{V}$ is found to be empty in their algorithm, it would make sense to notify the user that there is either a degenerate distribution or that a bigger value of r is required.

Assuming $\mathbf{V}$ is non-empty and its enumeration is feasible, the algorithm selects one of the possible $\mathbf{v}$ from $\mathbf{V}$ with equal probability and then generates a d using

$$q\left(d|\mathbf{v},\mathbf{y}\right) \propto \exp\left\{\gamma^T \mathbf{Z}^T \left(\mathbf{y}+d\mathbf{v}\right)\right\} \prod_{i=1}^{n} \begin{pmatrix} m_i \\ y_i + dv_i \end{pmatrix} = \eta\left(d|\mathbf{v},\mathbf{y}\right), \tag{2.8}$$

where the support of $q\left(d|\mathbf{v},\mathbf{y}\right)$ is given by

$$0 \leq y_i + dv_i \leq m_i \tag{2.9}$$

for all i. We introduce the additional notation $\eta\left(d|\mathbf{v},\mathbf{y}\right)$ to remind the reader that we are working with the density $q\left(d|\mathbf{v},\mathbf{y}\right)$ up to a constant of proportionality. According to equation (2.3), the Markov chain will have the target distribution as a stationary distribution if we include an acceptance probability $\alpha\left(\mathbf{y},\mathbf{y}^*\right)$ for moving from $\mathbf{y}$ to $\mathbf{y}^*$ such that

$$\alpha\left(\mathbf{y},\mathbf{y}^*\right) = \min\left\{\frac{\pi\left(\mathbf{y}^*\right) q\left(\mathbf{y}|\mathbf{y}^*\right)}{\pi\left(\mathbf{y}\right) q\left(\mathbf{y}^*|\mathbf{y}\right)}, 1\right\}. \tag{2.10}$$

We now show that the acceptance probability in (2.10) is one. The proposal distribution is

$$q(\mathbf{y}^*|\mathbf{y}) = \sum_{\{(\delta,\nu):\delta\nu=\mathbf{y}^*-\mathbf{y}\}} q(\delta|\mathbf{y}) f(\nu),$$

where δ is an integer that satisfies equation (2.9), $\nu \in \mathbf{V}$ and $f(\nu) = 1/|\mathbf{V}|$ is the uniform distribution over the set $\mathbf{V}$, described earlier. The sum simplifies to

$$q(\mathbf{y}^*|\mathbf{y}) = q(d|\mathbf{v},\mathbf{y}) f(\mathbf{v}) + q(-d|-\mathbf{v},\mathbf{y}) f(-\mathbf{v}),$$

for some d satisfying (2.9) and some $\mathbf{v} \in \mathbf{V}$ because of the restriction that all the vectors in the set $\mathbf{V}$ have coprime elements. Consequently, the acceptance probability of (2.10) is

$$\alpha\left(\mathbf{y},\mathbf{y}^*\right) = \min\left\{\frac{\pi\left(\mathbf{y}^*\right)\left[q\left(-d|\mathbf{v},\mathbf{y}^*\right) f\left(\mathbf{v}\right) + q\left(d|-\mathbf{v},\mathbf{y}^*\right) f\left(-\mathbf{v}\right)\right]}{\pi\left(\mathbf{y}\right)\left[q\left(d|\mathbf{v},\mathbf{y}\right) f\left(\mathbf{v}\right) + q\left(-d|-\mathbf{v},\mathbf{y}\right) f\left(-\mathbf{v}\right)\right]}, 1\right\}.$$

For a fixed vector $\mathbf{v}$, let $N_{\mathbf{v}}(\mathbf{y}) = \{\mathbf{u} : 0 \leq u_i \leq m_i \text{ and } \mathbf{u} = \mathbf{y} + \delta\mathbf{v} \text{ for some integer } \delta\}$. Then, from equations (2.8) and (2.9), the normalizing constant for $q(d|\mathbf{v}, \mathbf{y})$ is

$$\sum_{\{\delta\,:\,0 \leq y_i + \delta v_i \leq m_i\}} \eta(\delta|\mathbf{v}, \mathbf{y}) = \sum_{\mathbf{u} \in N_{\mathbf{V}}(\mathbf{y})} \exp\left\{\gamma^T \mathbf{Z}^T \mathbf{u}\right\} \prod_{i=1}^{n} \binom{m_i}{u_i}.$$

By definition, $N_{\mathbf{v}}(\mathbf{y}) = N_{-\mathbf{v}}(\mathbf{y})$ and, for $\mathbf{y}^* = \mathbf{y} + d\mathbf{v}$, $N_{\mathbf{v}}(\mathbf{y}) = N_{\mathbf{v}}(\mathbf{y}^*)$. Hence, for $\mathbf{y}^* = \mathbf{y} + d\mathbf{v}$, the distributions $q(\cdot|\mathbf{v}, \mathbf{y})$, $q(\cdot| - \mathbf{v}, \mathbf{y})$, $q(\cdot|\mathbf{v}, \mathbf{y}^*)$ and $q(\cdot| - \mathbf{v}, \mathbf{y}^*)$ all have the same normalizing constant. Since $f(\mathbf{v}) = 1/|\mathbf{V}|$, we therefore obtain

$$\alpha(\mathbf{y}, \mathbf{y}^*) = \frac{\pi(\mathbf{y}^*)\left[q(-d|\mathbf{v}, \mathbf{y}^*) f(\mathbf{v}) + q(d| - \mathbf{v}, \mathbf{y}^*) f(-\mathbf{v})\right]}{\pi(\mathbf{y})\left[q(d|\mathbf{v}, \mathbf{y}) f(\mathbf{v}) + q(-d| - \mathbf{v}, \mathbf{y}) f(-\mathbf{v})\right]} = \frac{\pi(\mathbf{y}^*)\left[q(-d|\mathbf{v}, \mathbf{y}^*) + q(d| - \mathbf{v}, \mathbf{y}^*)\right]}{\pi(\mathbf{y})\left[q(d|\mathbf{v}, \mathbf{y}) + q(-d| - \mathbf{v}, \mathbf{y})\right]}$$

$$= \frac{\exp\left\{\gamma^T \mathbf{Z}^T (\mathbf{y}+d\mathbf{v})\right\} \prod_{i=1}^{n} \binom{m_i}{y_i + dv_i} \left[\exp\left\{\gamma^T \mathbf{Z}^T \mathbf{y}\right\} \prod_{i=1}^{n} \binom{m_i}{y_i} + \exp\left\{\gamma^T \mathbf{Z}^T \mathbf{y}\right\} \prod_{i=1}^{n} \binom{m_i}{y_i}\right]}{\exp\left\{\gamma^T \mathbf{Z}^T \mathbf{y}\right\} \prod_{i=1}^{n} \binom{m_i}{y_i} \left[\exp\left\{\gamma^T \mathbf{Z}^T (\mathbf{y}+d\mathbf{v})\right\} \prod_{i=1}^{n} \binom{m_i}{y_i + dv_i} + \exp\left\{\gamma^T \mathbf{Z}^T (\mathbf{y}+d\mathbf{v})\right\} \prod_{i=1}^{n} \binom{m_i}{y_i + dv_i}\right]}$$

$$= 1.$$

A summary of the resulting Metropolis-Hastings algorithm is:

1. Enumerate the set $V = \left\{\mathbf{v} : \sum_{i=1}^{n} |v_i| \leq r \text{ and } v_i \text{ coprime for } i = 1, ..., n,\ \mathbf{W}^T\mathbf{v} = \mathbf{0}\right\}$.
2. Select a $\mathbf{v} \in \mathbf{V}$ with uniform probability.
3. Find all integers d such that $0 \leq y_i + dv_i \leq m_i$. Let k denote the number of values of d that were found.
4. Calculate $\sum_{i=1}^{k} \eta(d_i|\mathbf{v}, \mathbf{y})$ and assign $P(d_i) = \frac{\eta(d_i|\mathbf{v},\mathbf{y})}{\sum_{j=1}^{k} \eta(d_j|\mathbf{v},\mathbf{y})} \quad i = 1, 2, \ldots, k$
5. Choose a d according to the probabilities assigned to each d_i in Step 4.
6. Set $\mathbf{y}^* = \mathbf{y} + d\mathbf{v}$
7. Let $\mathbf{y} = \mathbf{y}^*$, go to Step 2

As was mentioned earlier, a limitation of this algorithm is that complete enumeration of the set $\mathbf{V}$ is required. In the next section we describe a modification to the algorithm that circumvents the need to enumerate $\mathbf{V}$.

2.4 Modified Algorithm

The Forster et al. (2003) algorithm proposes uniform sampling from the set

$$\mathbf{V} = \left\{ \mathbf{v} : \sum_{i=1}^{n} |v_i| \leq r \text{ and } v_i \text{ coprime for } i = 1, ..., n, \ \mathbf{W}^T \mathbf{v} = \mathbf{0} \right\},$$

by enumerating and storing it in memory. However, these authors provide no guidance for how to enumerate and store $\mathbf{V}$ in practice. One important difference between our method and theirs is that we sample uniformly from a subset $\mathbf{V_A}$ of $\mathbf{V}$ whose vectors satisfy the additional constraint that $|v_i| \leq m_i$ for all $1 \leq i \leq n$. We use $\mathbf{V}_A$ to improve mixing, as vectors for which some $|v_i| > m_i$ will only satisfy constraint (2.9) if $d = 0$, so that $\mathbf{y}^* = \mathbf{y}$ with probability one. A second way in which our method differs is that we sample vectors uniformly *without enumeration* from the larger set

$$\mathbf{V}' = \left\{ \mathbf{v} : \sum_{i=1}^{n} |v_i| \leq r, \quad \text{and} \quad v_i \text{ coprime for } i = 1, \ldots, n, \ \sum_{i=1}^{n} v_i = 0 \right\}$$

and then reject all vectors that are not in $\mathbf{V_A}$. This amounts to rejecting all $\mathbf{v}$ for which either $\mathbf{W^T v} \neq \mathbf{0}$ or $|v_i| > m_i$ for some $1 \leq i \leq n$. Our algorithm rejects on average, a proportion of $1 - |\mathbf{V_A}|/|\mathbf{V}'|$ realized $\mathbf{v} \in \mathbf{V}'$. In section 2.7, we discuss how this additional rejection-sampling step of the modified algorithm affects its speed.

To uniformly sample from $\mathbf{V}'$, consider r-dimensional vectors in the set

$$\mathcal{R} = \left\{ \mathbf{r} : \sum_{i=1}^{r} |r_i| \leq r \text{ and } r_i \text{ coprime for } i = 1, \ldots r, \ \sum_{i=1}^{r} r_i = 0 \right\},$$

where $r \leq n$. Classify as equivalent two vectors in $\mathcal{R}$ if one can be obtained as a permutation of the other. We start by enumerating a seed set $\mathbf{R}$ of representative vectors from the resulting equivalence classes on $\mathcal{R}$. To obtain the seed set $\mathbf{R}$ we do the following:

1. List all positive integers $\leq r/2$ and mutually coprime with one another. For example, if $r = 6$, we obtain $1, 2, 3$.

2. For $1 \leq j \leq r/2$, list all possible combinations of size j taken from the list in step 1. For example, if r=6, we obtain

   for j=1: $\{1\}, \{2\}, \{3\}$

for $j = 2 : \{1,2\},\{1,3\},\{2,3\}$

for $j = 3 : \{1,2,3\}$.

3. For each combination of size j above, prefix $0 \leq k \leq r - j$ ones to form vectors of size $\leq r$. For example, if j=2 above, the combination $\{2,3\}$ would lead to vectors: $(2,3),(1,2,3),(1,1,2,3),(1,1,1,2,3)$, and $(1,1,1,1,2,3)$.

4. Remove all vectors for which $\sum_i r_i > r$ or for which $\sum_i r_i$ is odd. For instance, of the five vectors listed in the example of step 3, all but $(1,2,3)$ would be removed. Also, remove any duplicate vectors.

5. Enumerate all the possible ways that negative signs can be assigned to the elements of the remaining vectors so that they sum to zero. The seed set $\mathbf{R}$ is the list of vectors that results after appending the appropriate number of zeros so that there are r elements.

Conceptually, $\mathbf{V}'$ can be obtained by uniformly assigning the non-zero components of each vector in the seed set $\mathbf{R}$ to an empty n–dimensional vector, and then filling in the remaining empty components with zeros. Each seed vector $\mathbf{r}$ in $\mathbf{R}$ thus represents

$$m(\mathbf{r}) = \frac{n!}{(n - \sum \lambda_j)!(\prod \lambda_j!)} \tag{2.11}$$

vectors in $\mathbf{V}'$, where the λ_j's denote the number of occurrences of each distinct non-zero value of the components of the seed vector. [For example, if $\mathbf{r} = (1,1,1,0,0,-1,-1,-1)$ and $n = 12$, then $\lambda_1 = \lambda_2 = 3$ and $m(\mathbf{r}) = \frac{12!}{(12-3-3)!3!3!} = 18480$.] Hence,

$$|\mathbf{V}'| = \sum_{\mathbf{r}\in\mathbf{R}} m(\mathbf{r}). \tag{2.12}$$

To avoid enumerating all the vectors in $\mathbf{V}'$, we then select seed vectors $\mathbf{r} \in \mathbf{R}$ with probability

$$P(\mathbf{r}) = \frac{m(\mathbf{r})}{\sum_{\mathbf{r}\in\mathbf{R}} m(\mathbf{r})} = \frac{m(\mathbf{r})}{|\mathbf{V}'|}.$$

Each vector in $\mathbf{V}'$ can only be obtained from one seed vector in $\mathbf{R}$. For a given $\mathbf{v} \in \mathbf{V}'$, denote the corresponding seed vector by $\mathbf{r}^*$. Then

$$P(\mathbf{v}) = \sum_{\mathbf{r}\in\mathbf{R}} P(\mathbf{v} \mid \mathbf{r}) \times P(\mathbf{r}) = P(\mathbf{v} \mid \mathbf{r}^*) \times P(\mathbf{r}^*) = \frac{1}{m(\mathbf{r}^*)} \times \frac{m(\mathbf{r}^*)}{|\mathbf{V}'|} = \frac{1}{|\mathbf{V}'|}$$

and so sampling from the set $\mathbf{V}'$ is uniform, as claimed.

Section 2.3 discusses why the Metropolis-Hastings algorithm of Forster et al. (2003) has the desired stationary distribution. Their algorithm samples perturbations $\mathbf{v}$ of the response from the set $\mathbf{V}$, whereas ours samples $\mathbf{v}$ from $\mathbf{V_A} \subset \mathbf{V}$. Both algorithms have an acceptance probability of 1. We argue as follows that our modified algorithm also has the desired stationary distribution. Given the way $q(d|\mathbf{v}, \mathbf{y})$ is defined in (2.8), only one condition on $\mathbf{V}$ and one condition on sampling from $\mathbf{V}$ are needed to show that the algorithm of Forster et al. (2003) has the desired stationary distribution and that the required acceptance probability is 1:

1. The vectors in $\mathbf{V}$ are unique in the sense that no vector can be obtained as a scalar multiple of another (due to the restriction to coprime elements).

2. There is uniform sampling from $\mathbf{V}$.

Thus, to show that our algorithm with acceptance probability 1, has the desired stationary distribution, it suffices to show the same conditions for $\mathbf{V}_A$. By definition, the vectors in $\mathbf{V}_A$ have coprime elements and so satisfy the first condition. Our algorithm is constructed to sample uniformly from $\mathbf{V}_A$ and so satisfies the second condition. Hence our stationary distribution is the same as that of Forster et al. (2003).

2.5 Exact Conditional Inference

Let $S_1, ..., S_p$ denote the sufficient statistics for $\beta_1, ..., \beta_p$, the parameters not of interest (nuisance parameters) in the logistic regression model. Likewise let $T_1, ..., T_q$ denote the sufficient statistics for $\gamma_1, ..., \gamma_q$, the parameters of interest. In this section we describe the methods used by our program to conduct hypothesis tests and produce point estimates for the parameters of interest.

2.5.1 Joint Conditional Inference

To conduct joint inference on $\gamma_1, ..., \gamma_q$ our program first produces a sample of dependent observations from a distribution with density

$$f_{T_1,...,T_q}(t_1, ..., t_q \mid S_1 = s_1, ..., S_p = s_p,\ \gamma_1 = \cdots = \gamma_q = 0). \tag{2.13}$$

In words, (2.13) is the joint conditional density for the sufficient statistics of interest given the observed values, $s_1, ..., s_p$, under the hypothesis H_0 that $\gamma_1 = \cdots = \gamma_q = 0$. In order to test

$$H_0 : \gamma_1 = \cdots = \gamma_q = 0$$

against the two-sided alternative

$$H_1 : \exists\ \gamma_i \neq 0, \quad i = 1, ..., q$$

we compute an approximate two-sided p-value for the *conditional probabilities test*. The conditional probabilities two-sided p-value is obtained by summing estimates of (2.13) over the critical region

$$E_{cp} = \left\{ \mathbf{v} : \hat{f}_{\mathbf{T}}(\mathbf{v} \mid \mathbf{S} = \mathbf{s},\ \gamma = \mathbf{0}) \leq \hat{f}_{\mathbf{T}}(\mathbf{t} \mid \mathbf{S} = \mathbf{s},\ \gamma = \mathbf{0}) \right\},$$

where $\mathbf{t}$ is the observed value of the sufficient statistics for $\gamma_1, ..., \gamma_q$ and $\hat{f}_{\mathbf{T}}$ is an estimate of (2.13). The Monte Carlo standard error of the resulting p-value is computed by the batch-means method [7], as described in Appendix A.

In the future, once we are able to devise an unbiased estimate of $\mathbf{\Sigma}$, the variance-covariance matrix of $(\mathbf{T_1}, ..., \mathbf{T_q})$ from (2.13), we plan to implement the *conditional scores test*. The conditional scores two-sided p-value is obtained by summing estimates of (2.13) over the critical region

$$E_{cs} = \left\{ \mathbf{v} : (\mathbf{v} - \hat{\mu})^T \hat{\mathbf{\Sigma}}^{-1} (\mathbf{v} - \hat{\mu}) \geq (\mathbf{t} - \hat{\mu})^T \hat{\mathbf{\Sigma}}^{-1} (\mathbf{t} - \hat{\mu}) \right\},$$

where μ and $\mathbf{\Sigma}$ are, respectively, the mean and variance-covariance matrix of $(T_1, ..., T_q)$ from (2.13). We cannot use the sample variance-covariance matrix as an estimate of $\mathbf{\Sigma}$, because the generated sample is composed of dependent observations. In this case, the sample mean is an unbiased estimator of μ; however, the sample variance-covariance matrix

is not an unbiased estimator of $\mathbf{\Sigma}$. Section 4.2 provides a brief discussion of a proposal to obtain a less biased estimate of $\mathbf{\Sigma}$.

Point estimates for each parameter of interest are obtained by jointly maximizing the conditional likelihood function of $f_{T_1,...,T_q}(t_1,...,t_q \mid S_1 = s_1,...,S_p = s_p,\ \gamma_1,\cdots,\gamma_q)$. If, however, the observed value t_i of the sufficient statistic for γ_i obtains the maximum or minimum value of the generated sample, then $\hat{\gamma}_i$ is $\pm\infty$ and so does not attain its maximum within the boundary of the parameter space.

2.5.2 Inference on a Single Parameter

Suppose we are interested in conducting inference on γ_i alone. In this case, $\gamma_1,...,\gamma_{i-1},\gamma_{i+1},...,\gamma_q$ together with $\beta_1,...,\beta_p$ would be considered nuisance parameters and according to (2.13), inference would be based on generating a sample of dependent observation from

$$f_{T_i}(t_i \mid T_1 = t_1,...,T_{i-1} = t_{i-1},T_{i+1} = t_{i+1},...,T_q = t_q,\ S_1 = s_1,...,S_p = s_p,\ \gamma_i = 0). \tag{2.14}$$

If we already have a sample of dependent observations from (2.13) and we would like to conduct inference on γ_i alone, it would be computationally expensive to generate another sample of dependent observations from (2.14) from scratch. On one hand, a sample from (2.14) can be easily extracted from an already existing sample generated under (2.13). On the other hand, the resulting sample can be too small to be practical, as it is difficult to adequately estimate the full conditional distribution of a single sufficient statistic from the joint conditional distribution of several. The procedure is best explained through a simple example, where we are able to explicitly enumerate the exact conditional joint distribution.

Suppose we have the data shown in Table 2.1 and we would like to find the exact distribution of the sufficient statistic, T_1, for γ_1 conditional on those for γ_0, and γ_2. Suppose also that the joint conditional distribution, shown in Table 2.2, of (T_1,T_2) given the observed value of T_0 is already available. The observed sufficient statistic for γ_1 is $t_1 = \begin{pmatrix} 1 & 2 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = 3$. Likewise, the observed sufficient statistics for γ_0, and

γ_2 are $t_0 = 2$ and $t_2 = 1$, respectively. The exact conditional distribution of T_1 given the observed values of T_0, and T_2, shown in Table 2.3, is simply obtained from Table 2.2 by extracting the information for every vector where $t_0 = 2$ and $t_2 = 1$.

Table 2.1: Example Data

Observation	y	x_0	x_1	x_2
1	0	1	1	0
2	1	1	2	1
3	0	1	1	1
4	1	1	1	0

Table 2.2: Tabulation of Possible Response Vectors

	y_1	y_2	y_3	y_4	t_0	t_1	t_2	$P(Y = y \mid T_0 = 2)$
1	1	1	0	0	2	3	1	p_1
2	0	1	1	0	2	3	2	p_2
3	0	0	1	1	2	2	1	p_3
4	1	0	0	1	2	2	0	p_4
5	0	1	0	1	2	3	1	p_5
6	1	0	1	0	2	2	1	p_6

Table 2.3: Extracted Conditional Distribution

t_1	Probability
2	$(p_3 + p_6) \,/\, (p_1 + p_3 + p_5 + p_6)$
3	$(p_1 + p_5) \,/\, (p_1 + p_3 + p_5 + p_6)$

In order to test

$$H_0 : \gamma_i = 0$$

against the two-sided alternative

$$H_1 : \gamma_i \neq 0$$

we compute an approximate two-sided p-value for the *conditional probabilities test*. The conditional probabilities two-sided p-value for a single parameter, γ_i, of interest is obtained

by summing (2.14) over the critical region

$$E_{cp} = \left\{ v : \hat{f}_{T_i}\left(v \mid \mathbf{S} = \mathbf{s},\ \mathbf{T}_{-i} = \mathbf{t}_{-i},\ \gamma_i = 0\right) \leq \hat{f}_{T_i}\left(t_i \mid \mathbf{S} = \mathbf{s},\ \mathbf{T}_{-i} = \mathbf{t}_{-i},\ \gamma_i = 0\right) \right\},$$

where $\mathbf{t}_{-i}$ is the observed value of the sufficient statistics for the remaining $q-1$ parameters of interest. The critical region, E_{cp}, contains all values of the estimated conditional probability (ie. the test statistic) that are no larger than the estimated conditional probability at the observed value of t_i. The Monte Carlo standard error of each resulting p-value is computed by the method of batched means.

In the future we plan to implement the *conditional scores test* for inference on a single parameter. The conditional scores two-sided p-value for a single parameter, γ_i, of interest is obtained by summing (2.14) over the critical region

$$E_{cs} = \left\{ v : (v - \hat{\mu}_i)^2 \frac{1}{\hat{\sigma}_i{}^2} \geq (t_i - \hat{\mu}_i)^2 \frac{1}{\hat{\sigma}_i{}^2} \right\},$$

where μ_i and σ_i^2 are, respectively, the mean and variance of T_i based on (2.14) and $\hat{\mu}_i$ and $\hat{\sigma}_i^2$ are their estimates.

2.6 Design and Implementation of Modified Algorithm

The R function that we provide is called *elrm* which stands for Exact inference for a Logistic Regression Model. The help file for the elrm function is included in Appendix B. The elrm function currently returns an object of class elrm. An elrm object contains the sampled values of the sufficient statistics, the joint conditional maximum likelihood estimates of the parameters of interest, the p-value for jointly testing that the parameters of interest are simultaneously equal to zero, the full conditional p-values from separately testing each parameter equal to zero, and the Monte Carlo standard error of each p-value estimate.

The elrm class has a summary method and a plot method. The summary method prints a table summarizing the results. The plot method produces both a trace and a histogram plot of the sampled values of the sufficient statistics for the parameters of interest.

We designed our program with two basic goals in mind: minimize computing time and optimize scalability. To address our first goal, we implemented our Markov chain algorithm entirely in C++, because C++ handles loops much quicker than R and the construction of

the Markov chain unfortunately requires the use of nested loops. Our C++ code is called within the *elrm*function in R by using the conventional ".C" method. A listing of the C++ functions used by *elrm* is included in Appendix D. The Markov chain produced by our C++ program is written to a file, which is then read by R. All inferences are conducted in R in order to take advantage of the fast built in methods for matrices supported by the R language. This also facilitates future scalability of our program as the generation of the Markov chain and inferences are done separately and not all at once on the fly. In the future we may wish to add more capabilities and inferential methods to our program and this may easily be done by simply appending the required code.

2.7 Analysis of Modified Algorithm

The *speed* of our algorithm is primarily determined by the size of $\mathbf{V}'$ relative to $\mathbf{V}_A$, which in turn depends on the number of nuisance parameters included in the model and the chosen value of r. Here, we define *speed* as the time it takes our algorithm to complete a fixed number of iterations. Recall that the only difference between the two sets $\mathbf{V}_A$ and $\mathbf{V}'$ is that the vectors in the larger set $\mathbf{V}'$ do not necessarily satisfy the constraints that $|v_i| \leq m_i$ for $i = 1, ..., n$ and that $\mathbf{W}^T\mathbf{v} = \mathbf{0}$, where $\mathbf{W}$ represents the design matrix for terms in the regression model that are not of interest. Within each iteration our algorithm repeatedly samples vectors from $\mathbf{V}'$ until the constraints $|v_i| \leq m_i$ for $i = 1, ..., n$ and $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ are satisfied. On average $(1 - |\mathbf{V}_A|/|\mathbf{V}'|) \times 100\%$ of sampled vectors are rejected within each iteration and the closer the two sets are to one another, the faster our algorithm will run as fewer rejections are required per iteration.

An increase in the number of nuisance parameters included in the model increases the size of $\mathbf{V}'$ relative to $\mathbf{V}_A$ resulting in more rejections per iteration. This is attributed to the fact that the size of $\mathbf{V}_A$ decreases for each additional nuisance parameter included in the model, as extra constraints need to be satisfied in $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ whereas the size of $\mathbf{V}'$ remains unaffected.

Both the speed and mixing rate of our algorithm are greatly determined by the chosen value of r. An increase in the value of r increases the number of non-zero indices in a sampled

vector $\mathbf{v} \in \mathbf{V}'$ which results in more rejections per iteration as constraint $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ and $|v_i| \leq m_i$ for $i = 1, ..., n$ become more difficult to satisfy. An increase in the value of r results in slower mixing because the constraint $0 \leq y_i + dv_i \leq m_i$ for all i becomes more difficult to satisfy for values of d different from zero. However, the Markov chain may remain trapped in a local neighborhood if r is chosen too small.

2.8 Implementation Impediments

2.8.1 Computing Time

When testing early versions of our program we found that in some cases it took a very long time to generate the Markov chain. The slow down was attributed to the number of variables that were being conditioned on. Let n be the number of observations and p be the number of nuisance parameters. The design matrix, $\mathbf{W}$, for the nuisance parameters was rather large in these cases. Recall that to sample from the set $\mathbf{V}_A$, our algorithm samples vectors $\mathbf{v} \in \mathbf{V}'$ until the constraints $|v_i| \leq m_i$ for $i = 1, ..., n$ and $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ are satisfied. The matrix multiplication for testing the condition $\mathbf{W}^T\mathbf{v} = \mathbf{0}$ for a sampled $\mathbf{v}$ from the set $\mathbf{V}'$ requires $2pn$ number of flops, where n is usually much larger than p. When the set $\mathbf{V}'$ is relatively large compared to $\mathbf{V}_A$, our algorithm spends most of its time checking the constraint $\mathbf{W}^T\mathbf{v} = \mathbf{0}$. For example, the size of the set $\mathbf{V}'$ for the diabetes data model that we wish to investigate was estimated (by counting the average number of rejections per iteration) to be approximately $412,779$ times larger than that of the set $\mathbf{V}_A$ for $r = 6$. To speed up the process of checking the constraint $\mathbf{W}^T\mathbf{v} = 0$, we can take advantage of the fact, that except for at most $r << n$ entries, the vector $\mathbf{v}$ is filled with zeros. By keeping track of which entries of a sampled $\mathbf{v}$ are nonzero, we are able to check the constraint $\mathbf{W}^T\mathbf{v} = 0$ with at most $2pr$ flops and check the constraint $|v_i| \leq m_i$ for $i = 1, ..., n$ with at most r flops, greatly speeding up the program.

2.8.2 Memory Constraints in R

We were not able to store large Markov chains in virtual memory, because the data structures used to store these chains require a lot of memory and RAM in a computer is limited.

To get around this problem, we generate the Markov chain in a sequence of batches. Each batch is then saved to a text file and stored in the hard drive. After a batch is written to the text file, the virtual memory allocated for that batch is released by our program. In this way, the size of the Markov chain is not restricted by the available amount of virtual memory and, given enough time, chains of essentially any size may be generated. Similarly, all our inferences are also accomplished by processing the Markov chain in batches to avoid having to store the entire Markov chain in memory.

Chapter 3

Analysis of Diabetes Data

3.1 Background Information and Data Summary

Insulin dependent (type 1) diabetes is typically characterized by the presence of antibodies to a variety of proteins. Having type 1 diabetes increases the risk for many serious complications such as heart disease, blindness, nerve damage, and kidney damage. Antibodies are frequently present years before the onset of symptoms and are therefore useful to predict future diabetes. Antibodies are not present in type 2 diabetes, and therefore antibody tests are useful to distinguish between these forms of diabetes. The data at hand were gathered to study the relationship between concentration levels (low and high) of the *islet antigen 2 antibody* (IA2A) and several covariates of potential interest in type 1 diabetes patients. The covariates of interest that were included in this study are *age*, *gender*, and the number of copies (0,1 or 2) of the *HLA-DQ2*, *HLA-DQ8* and *HLA-DQ6.2 haplotypes* (nDQ2, nDQ8 and nDQ6.2 respectively). Information on the variables of interest and the IA2A level of 669 type 1 diabetes patients was collected.

Table 3.1: Two-Way Contingency Tables Involving IA2A

		IA2A: low	IA2A: high
nDQ2	0	171 (44)	169 (60)
	1	189 (49)	108 (39)
	2	29 (7)	3 (**1**)
		389(100)	280(100)

		IA2A: low	IA2A: high
nDQ8	0	145 (37)	56 (20)
	1	222 (57)	192 (69)
	2	22 (6)	32 (11)
		389(100)	280(100)

		IA2A: low	IA2A: high
nDQ6.2	0	378 (97)	280 (100)
	1	11 (3)	0 (**0**)
	2	0 (**0**)	0 (**0**)
		389(100)	280(100)

		IA2A: low	IA2A: high
Gender	female	162 (42)	114 (41)
	male	227 (58)	166 (59)
		389(100)	280(100)

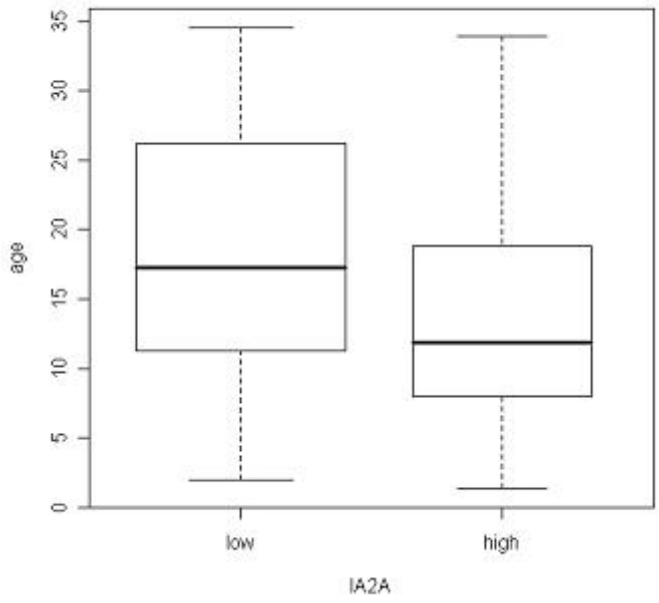
Two-way contingency tables for IA2A versus nDQ8, nDQ2, nDQ6.2 and gender are shown in Table 3.1, with percentages in each IA2A group given in brackets. A boxplot of age by IA2A is shown in Figure 3.1. Those cells which have bold numbers are problematic as they have expected counts smaller than 5, under the null hypothesis of no association. Notice that the data are sparse with a few cells having zero counts. The sparseness arises because very few patients have a copy of the "protective" HLA-DQ6.2 haplotype. Consequently, for the diabetes data at hand, any inferences made using methods that rely on large sample theory will produce unreliable results. Hence, this is a situation where exact inference is needed.

3.2 Results Obtained from R and LogXact

3.2.1 Logistic Regression in R

The investigators were primarily interested in age-specific genetic risks after adjusting for the effect of gender (Å Lernmark, personal communication). Therefore, the logistic regression

Figure 3.1: Boxplot of age by IA2A



model we chose to work with includes all main effects and two way interactions involving genetic effects and age. In R model formula notation, our model was:

IA2A $\sim$ age + gender + nDQ2 + nDQ8 + nDQ6.2 + age:nDQ2 + age:nDQ8 + age:nDQ6.2

The age of each subject was rounded to the nearest year. The results obtained by the glm function in R are shown in Appendix C. The estimates returned for terms involving nDQ6.2 are rather meaningless as they are accompanied by very large variances. The magnitude of the effect of nDQ6.2 gives us an indication as to what has gone wrong with the fitting. The parameter estimate for the effect of nDQ6.2 appears to be going off to negative infinity (and the corresponding fitted probability to zero). This indicates that the true value of the parameter could lie on or near the boundary of the parameter space which undermines the asymptotic theory for inference.

3.2.2 Exact Logistic Regression in LogXact

When applied to the diabetes data, the exact inference method provided by LogXact quickly runs out of computer memory during the network construction phase. As a consequence, no results were produced. The alternative Gibbs sampling approach provided by Logxact [5] produced a degenerate Markov chain with a single value.

3.3 Results Obtained by ELRM

3.3.1 Calling ELRM on the Diabetes Data

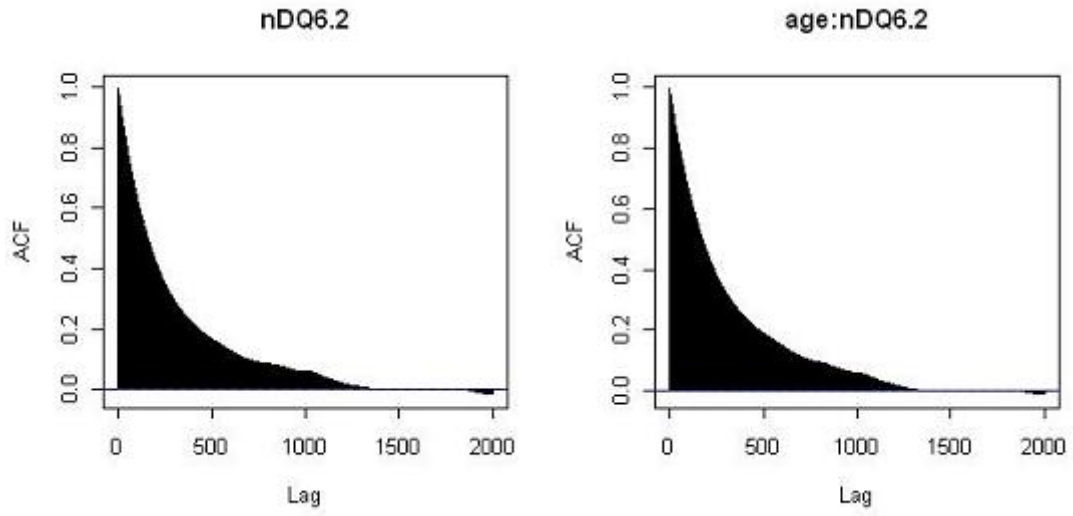
Here we assume the same model as before and include all main effects and two way interactions involving genetic effects and age (rounded to the nearest year). We are interested in testing the hypothesis that the regression coefficients for all model terms involving nDQ6.2 are zero. To investigate the effect of varying r on the properties of the sampler we ran the *elrm* method for several values of r $(2, 4, 6,$ and $8)$. The program was run on a pentium D, 3.2 GHz PC. The time it took to complete a sample of 1000 is shown in Table 3.2 along with the estimated size of $\mathbf{V}'$ relative to $\mathbf{V}_A$, the number of unique response vectors sampled and the corresponding number of unique sufficient statistic vectors sampled. There was a considerable overlap in the sampled vectors obtained using the value of $r = 4$ with those obtained using the values $r = 2, 6$ and 8. Irrespective of the sample size, the value of $r = 2$ produced Markov chains with the exact same set of response vectors suggesting that a reducible Markov chain is produced for this value of r. In the case of $r = 8$, very poor mixing was obtained. Apart from the noticeable jump in completion time between $r = 4$ and $r = 6$ there is a significant drop in the number of unique vectors sampled by the program. For these reasons, we chose to use the value of 4 for r. Results are based on a sample of 1000,000 (after a burn in of 2,000 iterations). Autocorrelation plots of the sampled sufficient statistics are shown in Figure 3.2. The burn-in period required may be calculated by looking at the time it takes for the autocorrelation to decay to a negligible level [7] such as ≤ 5 percent. In the multivariate scenario, the burn-in period required may be calculated as the maximum time needed for the autocorrelation of each covariate to decay to a negligible level. From the plots in Figure 3.2, a burn-in of 2000 iterations

appears to be reasonable for the autocorrelation to decay below 0.05.

Table 3.2: Time Required to Generate a Sample of 1000

r	**Time (minutes)**	$\|\mathbf{V}'\| / \|\mathbf{V}\|$	**Unique # of Visited States**	**Unique # of Sufficient Statistic Vectors**
2	0.32	4,730	8	8
4	0.43	17,713	198	14
6	5.37	412,779	84	6
8	52.03	3,934,130	6	2

Figure 3.2: Autocorrelation Plots of the Sampled Sufficient Statistics



The function call to *elrm* is shown below and required approximately 6 hours to complete.

```
diabetes.elrm = elrm(formula=IA2A/n~age+gender+nDQ2+nDQ8+nDQ6.2+age:nDQ2+age:nDQ8+age:nDQ6.2,
                     zCols=c(nDQ6.2,age:nDQ6.2),
                     r=4,
                     iter=1002000,
                     dataset=diabetes
                     burnIn = 2000)
```

Trace plots and histograms of the sampled sufficient statistics (T_1, T_2) for the regression parameters of interest, $\gamma = (\gamma_1, \gamma_2)$, corresponding to nDQ6.2 and age:nDQ6.2, respectively, are shown in Figures 3.3 and 3.4. The plots were obtained using the plot.elrm method with sampling fraction $p = .20$. The plots pertain to the conditional distributions of T_1 and of T_2 given the vector $\mathbf{S}$ of sufficient statistics observed for the remaining nuisance parameters, under the null hypothesis that $\gamma = (0, 0)$. Therefore, the histograms approximate the distributions of $(T_1|\mathbf{S}, \gamma = (0, 0))$ and $(T_2|\mathbf{S}, \gamma = (0, 0))$. The plots show that the two distributions take on more than a single value and so are not degenerate.

In Figure 3.3 a multimodal distribution is to be expected because the sufficient statistic is of the form $\sum_{i=1}^{n} y_i \times age_i \times DQ6.2_i$. To see this, consider the simplest case of a binary response vector $\mathbf{y}$. Index the 11 patients who carry DQ6.2 by 1 to 11. Suppose their ages range from 10-33 years. In the sum, the random response vectors $\mathbf{y}$ generated by the Markov chain may be viewed as selecting a random subset of ages of random size from these 11 subjects. Suppose that our 11 patients who carry DQ6.2 have $y_1 = \cdots = y_{11} = 0$ so that the size of the random subset of ages from these patients is zero. Then the distribution of the sufficient statistic must be a point mass at 0. Now suppose that exactly one of y_1 through y_{11} is 1 and that all the rest are zero, so that the size of the random subset of ages is 1. Then the distribution of the sufficient statistic has support from 10-33 years. Continuing, suppose that exactly two of y_1 through y_{11} are 1 and that all the rest are zero, so that the size of the random subset of ages is 2. Then the distribution of the sufficient statistic has support from 20-66 years. Continuing the argument, we see that the distribution of the sufficient statistic shown in Figure 3.3 is a mixture of component distributions corresponding to age subset sizes as low as zero and as high as 11. The apparent multimodality of the distribution shown in Figure 3.3 arises from the mixture property of this distribution.

The p-value estimate obtained for the joint effect of the parameters of interest is 0.914. The Monte Carlo standard error of the p-value estimate (using batch-means) is 0.00444. Looking at the the histogram of the joint distribution $(T_1, T_2|\mathbf{S}, \gamma_1 = \gamma_2 = 0)$ shown in Figure 3.5, one can see that the probability of observing $(T_1, T_2) = (0, 0)$ is relatively large compared to the other possible values, producing a large p-value for the joint test that the regression parameters corresponding to nDQ6.2 and age:nDQ6.2 are both zero.

When testing each parameter separately, we must also condition on the sufficient statistics for the remaining parameters of interest. Thus, the approximate two-sided p-values, for separately testing the effect of each parameter, are obtained using a subset of the sample produced to test their joint effect as described in Chapter 2, Section 2.5. In our case, the subsets were quite small (35,630) resulting in a significant loss of accuracy. Taking a closer look, we also found that the extracted samples were comprised of only the observed value (i.e., were degenerate). In fact, we expect the distributions of $(T_1|T_2, \mathbf{S}, \gamma_1 = 0)$ and $(T_2|T_1, \mathbf{S}, \gamma_2 = 0)$ to be degenerate, because the test statistics for T_1 and T_2 are

$$\begin{aligned} T_1 &= \sum_i Y_i \times nDQ6.2_i \\ T_2 &= \sum_i Y_i \times nDQ6.2_i \times age_i \end{aligned}$$

and the summands in these expressions are all greater than or equal to zero. Hence $T_1 = 0$ if and only if all summands are zero and similarly for T_2. It follows that $T_1 = 0$ implies $T_2 = 0$. Furthermore, since age is greater than zero for all patients, $\mathrm{T}_2 = 0$ implies $T_1 = 0$. Thus $T_1 = 0$ if and only if $T_2 = 0$. In other cases, where the conditional distributions of each sufficient statistic are not degenerate and more accuracy is required, one can simply run elrm over again and generate chains of the desired length for each conditional distribution of interest.

Maximum likelihood estimates for the parameters of interest were negative infinity and so did not fall within the boundary of the parameter space. The observed value of the sufficient statistic for each parameter of interest was the minimum possible value of zero.

Figure 3.3: Plots for nDQ6.2

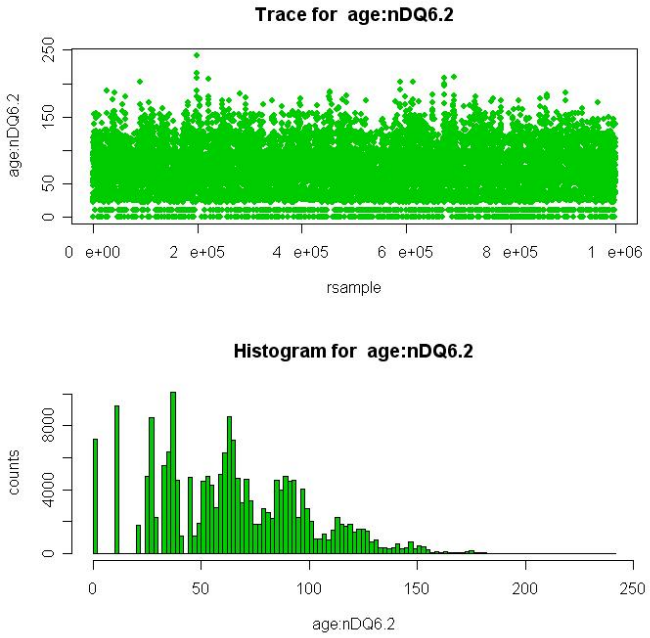


Figure 3.4: Plots for age:nDQ6.2

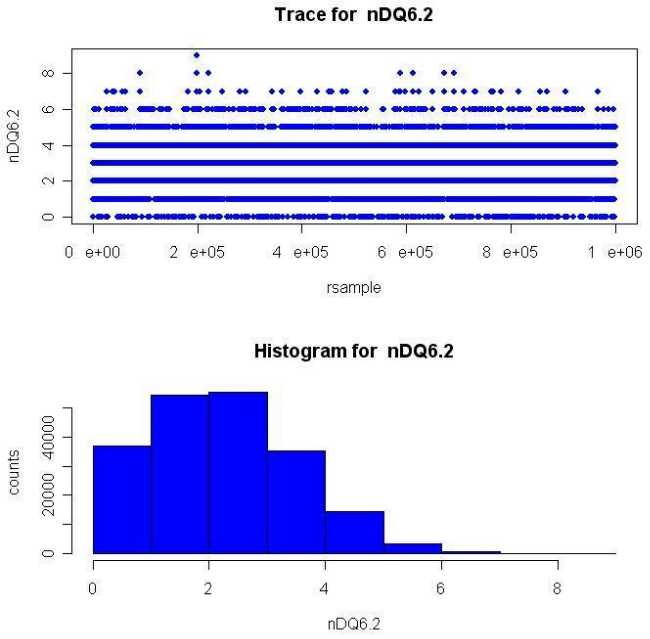
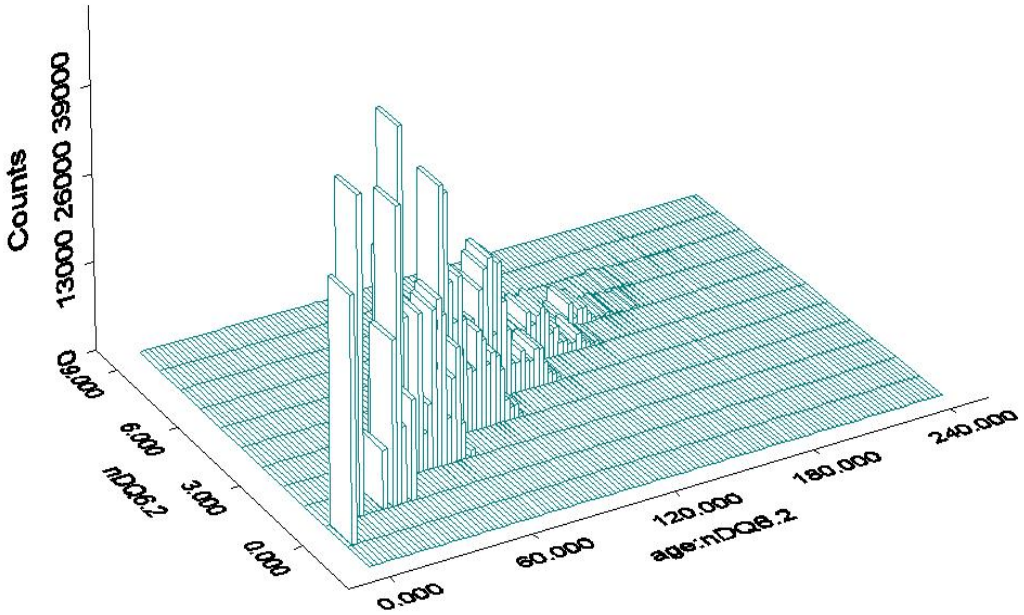


Figure 3.5: Histogram for (nDQ6.2, age:nDQ6.2)



Chapter 4

Conclusions and Future Work

4.1 Conclusions

The modified version of the Forster et al. (2003) algorithm developed in this work was found effective for performing exact inference for binomial regression models for large data sets. Instead of sampling from the set $\mathbf{V}$ used in the Forster et al. (2003) algorithm we sample from the set $\mathbf{V_A}$ to promote better mixing. The main advantage our algorithm has over the one described by Forster et al. (2003) is that we uniformly sample from the set $\mathbf{V}_A$ without having to enumerate it. This allows us to provide (approximate) exact inference for logistic regression models with response vectors of large length and further grouping of the predictor variable data are not needed.

In addition to running our program on the diabetes data set, which could not directly be analyzed by Logxact, we tested our program on several other data sets where reliable estimates may be obtained using Logxact. In all cases elrm provided estimates which were in close agreement with those produced by Logxact's exact method. In one particular case, elrm was able to produce accurate results where the Gibbs sampler method implemented in Logxact failed. We also compared results obtained by elrm on the diabetes data, but using smaller models that Logxact was able to handle. In most scenarios, age needed to be discretized into three or four categories so that Logxact did not run out of memory. In each case, the results obtained by elrm were in close agreement with those obtained using Logxact's exact method.

In the near future, to facilitate the use of this inferential approach, we plan to submit our method, called 'elrm', as an R package to the Comprehensive R Archive Network. In the meantime our program is available for both Windows and Unix users at *www.sfu.ca/dzamar*.

4.2 Future Improvements

4.2.1 Inverting the Test to Produce Confidence Intervals

In the future we plan to include confidence intervals for our parameter estimates. To obtain a confidence interval for γ_p we may invert the test from section 2.5.2. To obtain a level-α confidence interval, (γ_-, γ_+) for γ_i, we invert the conditional probabilities test for γ_i. Let $t_{\min}$ and $t_{\max}$ be the smallest and largest possible values of t_i in distribution (2.14). The lower confidence bound, γ_-, is such that

$$\begin{aligned} \sum_{v \geq t_i} f_{T_i}\left(v \mid \mathbf{S}=\mathbf{s},\, \mathbf{T}_{-i}=\mathbf{t}_{-i},\ \gamma_i=\gamma_-\right) &= \alpha/2 \qquad \text{if } \ t_{\min} < t_i \leq t_{\max}, \\ \gamma_- &= -\infty \qquad \text{if } \ t_i = t_{\min}, \end{aligned}$$

where $\mathbf{t}_{-i}$ are the observed values of the sufficient statistics for the remaining parameters of interest. Similarly the upper confidence bound, γ_+, is such that

$$\begin{aligned} \sum_{v \leq t_i} f_{T_i}\left(v \mid \mathbf{S}=\mathbf{s},\, \mathbf{T}_{-i}=\mathbf{t}_{-i},\ \gamma_i=\gamma_+\right) &= \alpha/2 \qquad \text{if } \ t_{\min} \leq t_i < t_{\max}, \\ \gamma_+ &= \infty \qquad \text{if } \ t_i = t_{\max}. \end{aligned}$$

4.2.2 Testing Other Hypotheses

Our program is currently only able to test the hypothesis that all the parameters of interest are equal to zero versus the alternative that at least one of the parameters is different from zero. We plan to extend elrm to allow for the testing of any hypothesis of the form:

$$\begin{aligned} H_0 &: \ \gamma_1 = \acute{\gamma}_1, \cdots, \gamma_q = \acute{\gamma}_q \\ &\textit{vs.} \\ H_1 &: \ \exists\ \gamma_i \neq \acute{\gamma}_i, \quad i = 1, ..., q \end{aligned}$$

where $\acute{\gamma}_1, \cdots, \acute{\gamma}_q$ are possibly nonzero values, by weighting the generated sampled frequencies of (2.13). This allows one to adjust the sampled frequencies under $H_0 : \gamma = \mathbf{0}$ to obtain sample frequencies under $H_0 : \gamma = \acute{\gamma}$. Suppose we wish to test $H_0 : \gamma = \acute{\gamma}$, then the corresponding sample frequencies, $f_{\acute{\gamma}}$, needed are obtained from the sampled frequencies, f_0, under $H_0 : \gamma = \mathbf{0}$ as follows:

$$f_{\mathbf{T}}\left(\mathbf{T} = \mathbf{t} \mid \mathbf{S} = \mathbf{s}, \gamma = \acute{\gamma}\right) = \frac{f_{\mathbf{T}}\left(\mathbf{t} \mid \mathbf{S} = \mathbf{s}, \gamma = \mathbf{0}\right) \exp\left\{\acute{\gamma}^T \mathbf{t}\right\}}{\sum\limits_{\tilde{\mathbf{t}}} f_{\mathbf{T}}\left(\mathbf{T} = \tilde{\mathbf{t}} \mid \mathbf{S} = \mathbf{s}, \gamma = \mathbf{0}\right) \exp\left\{\acute{\gamma}^T \tilde{\mathbf{t}}\right\}} \ .$$

4.2.3 Interlaced Estimator of the Variance

The univariate score test requires one to estimate both the mean and variance of the test statistic under the null hypothesis. In our case, the distribution of the test statistic is non-trivial and we are forced to approximate it using a sample of correlated observations generated via Markov chain Monte Carlo. The sample mean of dependent observations remains an unbiased estimate of $E[X_i] = \mu$:

$$E\left[\hat{\mu}\right] = E\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n}\sum_{i=1}^{n} E\left[X_i\right] = \frac{1}{n}\sum_{i=1}^{n} \mu = \mu \ .$$

However, the sample variance is not an unbiased estimate of $V(X_i) = \sigma^2$. The expected value of the sample variance, $\hat{\sigma}^2$ is

$$\begin{aligned}
E\left[\hat{\sigma}^{\mathbf{2}}\right] &= E\left[\frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2\right] \\
&= \frac{1}{n-1}E\left[\sum_{i=1}^{n}X_i^2 - n\bar{X}^2\right] \\
&= \frac{1}{n-1}\left[\sum_{i=1}^{n}E\left(X_i^2\right) - nE\left(\bar{X}^2\right)\right] \\
&= \frac{1}{n-1}\left\{\sum_{i=1}^{n}\left(\sigma^2+\mu^2\right) - n\left[V\left(\bar{X}\right) + E\left(\bar{X}^2\right)\right]\right\} \\
&= \frac{1}{n-1}\left\{n\left(\sigma^2+\mu^2\right) - n\left[V\left(\bar{X}\right) + \mu^2\right]\right\} \\
&= \frac{n}{n-1}\left[\sigma^2 - V\left(\bar{X}\right)\right] \\
&= \frac{n}{n-1}\left[\sigma^2 - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}cov\left(X_i, X_j\right)\right] \\
&= \frac{n}{n-1}\left[\sigma^2 - \frac{\sigma^2}{n} - \frac{2}{n^2}\sum_{i>j}cov\left(X_i, X_j\right)\right] \\
&= \sigma^2 - \frac{2}{n\left(n-1\right)}\sum_{i>j}cov\left(X_i, X_j\right) \\
&= \sigma^2 - \frac{2\sigma^2}{n\left(n-1\right)}\sum_{i>j}\rho_{ij} \\
&= \sigma^2\left[1 - \frac{2}{n\left(n-1\right)}\sum_{i>j}\rho_{ij}\right],
\end{aligned}$$

which is smaller than σ^2 if the covariance between X_i, and X_j is greater than zero for all $i \neq j$. In the future, we would like use an *interlaced variance estimator* [1] which is

less biased than the sample variance for time-dependent correlated data. The interlaced variance estimator is calculated as follows:

1. Find the smallest integer k such that the lag k autocorrelation of the sample is smaller than a chosen constant ε, where $-1 \leq \varepsilon \leq 1$. Usually ε is chosen close to zero. The lag k autocorrelation of a sample, denoted as r_k, is computed as follows:

$$r_k = \frac{\sum_{i=1}^{N-k} (x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_{i=1}^{N} (x_i - \bar{x})^2} .$$

2. Extract the following (less correlated) sub-samples:

$$\mathbf{U}_i = \left\{ x_i, x_{i+k}, x_{i+2k}, ..., x_{i+\lfloor(\frac{N-i}{k})\rfloor \cdot k} \right\}, \qquad i = 1, ..., k .$$

3. Calculate the sample variance, S_i^2, for each sub-sample:

$$S_i^2 = Var(\mathbf{U}_i), \qquad i = 1, ..., k .$$

4. Finally, the interlaced estimate of σ^2 is computed by summing up the values of S_i^2 and dividing by k:

$$\tilde{\sigma}^2 = \frac{1}{k} \sum_{i=1}^{k} S_i^2 .$$

The interlaced variance estimator can also be used in the multivariate scenario to obtain an unbiased estimate of the variance-covariance matrix by defining the lag k autocorrelation function as the maximum lag k autocorrelation among all pairs of components in the multivariate vector.

Appendix A

Monte Carlo Standard Error via Batch Means

The following description is adapted from the online-notes of Charlie Geyer (http://www.stat.umn.edu/geyer/mcmc/talk/mcmc.pdf) and Murali Haran (http://www.stat.psu.edu/~mharan/MCMCtut/batchmeans.pdf). Suppose that our Markov chain consists of n observations $X_1, X_2, ..., X_n$ and we would like to obtain an approximately unbiased estimate of the variance of $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i$, where $Y_i = g(X_i)$ for some function g. In Section 2.5 we used the batch means estimator to compute the variance of a p-value estimate obtained via Markov chain Monte Carlo. In this case, $g(x) = I[f_k(x) \leq f(x_{obs})]$, where $f_k(x)$ is the frequency of x in the k^{th} batch and $f(x_{obs})$ is the frequency of the observed value of the sufficient statistic in the Markov chain. By the Markov chain central limit theorem (CLT)

$$\lim_{n\to\infty} \sqrt{n}(\hat{\mu}_n - \mu) \quad \xrightarrow{d} \quad N(0, \sigma^2),$$

where

$$\sigma^2 = var\,(Y_i) + 2\sum_{i=1}^{\infty} cov\,(Y_i, Y_{i+k}) \tag{A.1}$$

(see Chan and Geyer, 1994, for details). Therefore,

$$var\,(\hat{\mu}_n) \approx \frac{\sigma^2}{n}.$$

Unfortunately, σ^2 as given in (A.1) is difficult to estimate directly. To remedy this problem, the batch means approach splits the Markov chain into k batches of size b and forms the batch averages:

$$\bar{Y}_i\,(b) = \frac{1}{b}\sum_{j=1}^{b} Y_{(i-1)b+j}, \quad i = 1, ..., k.$$

By the Markov chain CLT, we have

$$\sqrt{b}(\bar{Y}_i\,(b) - \mu) \xrightarrow{d} N(0, \sigma^2), \quad i = 1, ..., k$$

Hence, for large b,

$$b(\bar{Y}_i\,(b) - \mu)^2 \;\dot{\sim}\; \sigma^2\,\chi^2(1), \quad i = 1, ..., k.$$

Since the mean of a $\chi^2(1)$ random variable is 1, we can approximate σ^2 by averaging the left-hand side over several batches. Thus,

$$\frac{1}{k}\sum_{i=1}^{k} b\,(\bar{Y}_i\,(b) - \mu)^2 \approx \sigma^2.$$

But since we don't know μ, we replace it by $\hat{\mu}_n$ and divide by $k-1$ rather than k. Hence σ^2 may be approximated by

$$\begin{aligned}\hat{\sigma}^2 &= \frac{1}{k-1}\sum_{i=1}^{k} b\,(\bar{Y}_i\,(b) - \hat{\mu}_n)^2 \\ &= \frac{b}{k-1}\sum_{i=1}^{k}(\bar{Y}_i\,(b) - \hat{\mu}_n)^2\end{aligned}$$

and the Monte Carlo standard error of $\hat{\mu}_n$ is therefore

$$\frac{\hat{\sigma}}{\sqrt{n}} = \frac{\hat{\sigma}}{\sqrt{kb}} = \sqrt{\frac{1}{k(k-1)}\sum_{i=1}^{k}(\bar{Y}_i\,(b) - \hat{\mu}_n)^2}.$$

An important application aspect is the choice of batch size, b. The idea is to take b large enough so that the first-order autocorrelation of batch means is negligible. A popular approach is to calculate the autocorrelation between batch means for increasing batch sizes and stop when the autocorrelation falls below a given threshold (e.g., 0.05). Since this rule is implemented with empirical autocorrelations that are subject to sampling variability, it sometimes results in standard errors that are too small as the autocorrelation may rise again for larger batch sizes. Moreover, for Markov chains of finite size, the correlation between the batch means will always be positive resulting in residual dependence among the batch means.

Appendix B

ELRM Help File

Exact Logistic Regression Model

Description:

'elrm' implements a Markov Chain Monte Carlo algorithm to approximate exact conditional inference for logistic regression models. Exact conditional inference is based on the distribution of the sufficient statistics for the parameters of interest given the sufficient statistics for the remaining nuisance parameters. Users specify a logistic model and a list of model terms of interest for exact inference.

Usage:

```
elrm(formula, zCols, r=4, iter=1000, dataset, burnIn=iter/10)
```

Arguments:

formula: a formula object that contains a symbolic description of the logistic regression model of interest in the usual R formula format. One exception is that the binomial response should be specified as 'success'/'trials', where 'success' gives the number of successes and 'trials' gives the number of binomial trials for each row of 'dataset'. See the Examples section for an example elrm formula.

zCols: a vector of model terms for which exact conditional inference is of interest.

r: a parameter of the MCMC algorithm that influences how the Markov chain moves around the state space. Small values of r cause the chain to take small, relatively frequent steps through the state space; larger values cause larger, less

```
         frequent steps. The value of r must be an even integer less
         than or equal to the length of the response vector. Typical
         values are 4, 6 or 8. See the References for further details.

   iter: an integer representing the number of Markov chain
         iterations to make (must be larger than or equal to 1000).

dataset: a data.frame object where the data are stored.

 burnIn: the burn-in period to use when conducting inference. Values
         of the Markov chain in the burn-in period are discarded.
```

Details:

```
Exact inference for logistic regression is based on generating the
conditional distribution of the sufficient statistics for the
regression parameters of interest given the sufficient statistics
for the remaining nuisance parameters. A recursive algorithm for
generating the required conditional distribution is implemented in
statistical software packages such as LogXact and SAS. However,
their algorithms can only handle problems with modest samples sizes
and numbers of covariates. ELRM is an improved version of the
algorithm developed by Forster et. al (2003) which is capable of
conducting (approximate) exact inference for large sparse data sets.

A typical predictor has the form "response/n~terms" where 'response'
is the name of the (integer) response vector, 'n' is the column name
containing the binomial sizes and 'terms' is a series of terms which
specifies a linear predictor for the log odds ratio.

A specification of the form 'first:second' indicates the
interactions of the term 'first' with the term 'second'.
```

Value:

```
'elrm' returns an object of class '"elrm"'.

An object of class '"elrm"' is a list containing the following
components:

  estimates: a vector containing the parameter estimates.

   p.values: a vector containing the estimated p-value for each
             term of interest.

p.values.se: a vector containing the standard errors of the
```

```
             estimated p-values of each term of interest.

         mc: a data.frame containing the Markov chain produced.

sample.size: a vector containing the number of states from the
             Markov chain that were used to get each conditional
             distribution for testing each individual parameter.

       call: the matched call.
```

References:

Forster J, McDonald J, Smith P. Markov Chain Monte Carlo Exact Inference for Binomial and Multinomial Logistic Regression Models. Statistics and Computing, 13:169 177, 11 2003.

Mehta CR, Patel NR. Logxact for Windows. Cytel Software Corporation, Cambridge, USA, 1999.

Zamar David. Monte Carlo Markov Chain Exact Inference for Binomial Regression Models. Master's thesis, Statistics and Actuarial Sciences, Simon Fraser University, 2006.

See Also:

'summary.elrm' 'plot.elrm'.

Examples:

```
# Example using the urinary tract infection data set
uti.dat=read.table("uti.txt",header=T);

# call the elrm function on the urinary tract infection data with
#'dia' as the single parameter of interest
uti.elrm=elrm(formula=uti/n~age+oc+dia+vic+vicl+vis, zCols=c(dia),
              r=2, iter=50000, dataset=uti.dat, burnIn=100);
```

Summarizing Exact Logistic Regression Model Tests

Description:

These functions are all ‘methods’ for class ‘elrm’ objects.

Usage:

```
summary(object)
plot(x, p=0.2, breaks=‘Sturges’)
```

Arguments:

object: ‘elrm’ object, typically result of ‘elrm’.

x: ‘elrm’ object, typically result of ‘elrm’.

p: the sampling fraction of points to be plotted.

breaks: one of:

* a vector giving the number of cells to use for the histogram of each sufficient statistic of interest.
* a single number giving the number of cells for each histogram.
* the character string naming an algorithm to compute the number of cells (see Details).

Details:

‘summary.elrm’ formats and prints out the results of an elrm object. These include coefficient estimates for models terms of interest, their corresponding p-value and MCMC standard error of the p-value as well as the sample size each inference is based on.

‘plot.elrm’ produces both a trace and histogram of the sampled values of the sufficient statistics for each parameter of interest.

The default for ‘breaks’ is ‘"Sturges"’: see ‘nclass.Sturges’. Other names for which algorithms are supplied are ‘"Scott"’ and ‘"FD"’.

References:

Forster J, McDonald J, Smith P. Markov Chain Monte Carlo Exact Inference for Binomial and Multinomial Logistic Regression Models. Statistics and Computing, 13:169 177, 11 2003.

Mehta CR, Patel NR. Logxact for Windows. Cytel Software Corporation, Cambridge, USA, 1999.

Zamar David. Monte Carlo Markov Chain Exact Inference for Binomial Regression Models. Master's thesis, Statistics and Actuarial Sciences, Simon Fraser University, 2006.

Examples:

```
# Example using the urinary tract infection data set
uti.dat=read.table("uti.txt",header=T);

# call the elrm function on the urinary tract infection data with
#'dia' as the single parameter of interest
uti.elrm=elrm(formula=uti/n~age+oc+dia+vic+vicl+vis, zCols=c(dia),
              r=2, iter=50000, dataset=uti.dat, burnIn=100);

# call the summary method
summary(uti.elrm);

# plot a random sample of size 10000 from the Markov chain generated.
# the observations chosen remain in the order in which they were generated.
plot(uti.elrm,0.20);
```

Appendix C

Diabetes Data GLM Output

```
Call:
glm(formula = IA2A/n ~ age + gender + nDQ2 + nDQ8 + nDQ6.2 +
    age:nDQ2 + age:nDQ8 + age:nDQ6.2, family = "binomial", weights = n)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3268  -0.8601  -0.2893   0.8323   2.4584

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  8.551e-01  4.284e-01   1.996 0.045938 *
age         -8.162e-02  2.373e-02  -3.440 0.000582 ***
gender       1.901e-01  1.732e-01   1.098 0.272340
nDQ2        -9.233e-01  3.340e-01  -2.765 0.005696 **
nDQ8         3.489e-01  3.524e-01   0.990 0.322216
nDQ6.2      -1.758e+01  3.021e+03  -0.006 0.995357
age:nDQ2     1.614e-02  1.947e-02   0.829 0.407270
age:nDQ8     6.904e-03  1.981e-02   0.349 0.727464
age:nDQ6.2   7.779e-02  1.102e+02   0.001 0.999437
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 407.21  on 284  degrees of freedom
Residual deviance: 307.99  on 276  degrees of freedom
AIC: 507.15

Number of Fisher Scoring iterations: 15
```

Appendix D

C++ Code Listing

Files	Functions	Description
binomMCMC.cpp	Matrix nextV(Matrix, Matrix, Array2D<int>, int)	Choose a vector from the set $\mathbf{V}'$ with uniform probability.
	int getd(Matrix, Matrix, Matrix)	Choose an integer d according to the probabilities assigned to each d_i.
	void writeToFile(ofstream& , Matrix*, int)	Write the buffered Markov chain chain output to a file and clear the buffer.
	extern "C" void MCMC(int*, int*, int*, int*, int*, int*, int*, char**, int*)	Function called by elrm to generate the Markov chain.
functions.cpp	void swap(int*, int, int)	Swap two entries of an integer array.
	double factorial(double)	Calculate the factorial of an integer number stored as a double.
	double nCk(double, double)	Combinatorial calculator.
	void randPerm(int*, int)	Randomly permute an array of integers.
	void rsample(int*, int*, int, int)	Takes a random sample of a specified size from the elements of an integer array without replacement.

Files	Functions	Description
	int permute(int*, int)	Obtain the next (nonrandom) sequential permutation of an array of integers [9].
	void quicksort(int*, int)	Sort an array of integers from smallest to largest [14].
	List<int*> unique(List<int*>, int)	Find all unique integers in a list of integers.
	Array2D<int> findR(int, int)	Find all vectors in the seed set **R**.
	List<List<double>*> readDataFile(char*)	Read a data file.
	double* getWeightings(Array2D<int>, int)	Obtain weightings for vectors in the seed set **R**.
	rand_int(int)	Obtain a random integer between 0 (inclusive) and N (exclusive).
functions.h		Header file of functions.cpp.
List.h		List template class.
matrix.h		Matrix template class [16].

Bibliography

[1] Ceylan D. and Shmeiser B.W. Interlaced Variance Estimators. *Proceedings of the Winter Simulation Conference*, pages 1382–1383, 1993.

[2] Chan, K.S. and Geyer C.J. Discussion of the paper by Tierney. *Annals of Statistics*, 22:1747–1758, 1994.

[3] Corcoran C., Metha C., Patel N.R. and Senchaudhuri P. Computational Tools for Exact Conditional Logistic Regression. *Stat Med*, 20:2723–2739, 2001.

[4] Cox D. *Analysis of Binary Data*. Chapman and Hall, New York, USA, 1970.

[5] Forster J.J., McDonald J.W. and Smith P.W.F. Monte Carlo Exact Conditional Tests for Log-Linear and Logistic Models. *Journal of the Royal Statistical Society*, 58(2):445–453, 1996.

[6] Forster J.J., McDonald J.W. and Smith P.W.F. Markov Chain Monte Carlo Exact Inference for Binomial and Multinomial Logistic Regression Models. *Statistics and Computing*, 13:169–177, 2003.

[7] Geyer C.J. Practical Markov Chain Monte Carlo. *Statistical Science*, 7:473–511, 1992.

[8] Hirji K., Mehta C.R. and Patel N.R. Computing Distributions for Exact Logistic Regression. *Journal of The American Statistical Association*, 82(400):1110–1117, 1987.

[9] Jeffrey A. Johnson. SEPA: A Simple, Efficient Permutation Algorithm, 2001. www.freewebz.com/permute/soda_submit.html.

[10] Mehta C.R. and Patel N.R. *Logxact for Windows*. Cytel Software Corporation, Cambridge, USA, 1999.

[11] Mehta C.R., Cyrus R. and Patel N.R. Exact Logistic Regression: Theory and Examples. *Statistics in Medicine*, 14:2143–2160, 1995.

[12] Mehta C.R., Patel N.R. and Senchaudhuri P. Efficient Monte Carlo Methods for Conditional Logistic Regression. *Journal of The American Statistical Association*, 95(449):99–108, 2000.

[13] Mehta C.R., Patel N.R. and Wei L.J. Constructing Exact Significance Tests With Restricted Randomization Rules. *Biometrika*, 75(2):295–302, 1988.

[14] Michael Main. *Data Structures and Other Objects Using C++*. Addison-Wesley, 2005.

[15] Ross S. *Introduction to Probability Models (2nd Edition)*. Harcourt Academic Press, San Diego, USA, 2000.

[16] Techsoft Pvt. Ltd. Matrix TCL Lite v1.13 Copyright (c) 1997-2002.

[17] Walsh B. Lecture Notes for EEB 581, 2004.