A Comparative Analysis of Bayesian Inference for Logistic Regression

Mark O'Connor

The thesis is submitted to University College Dublin
in part fulfilment of the requirements for the degree of
BSc Statistics

School of Mathematics and Statistics
University College Dublin

Supervisor: Dr. Claire Gormley

May 18, 2025

Abstract

For binary classification and regression, the logistic model is typically used to capture the non-linear effects of the explanatory variables on the binary response variable. However, the posterior distribution of the logistic model is intractable due to the exponential form of the likelihood and lack of a conjugate prior. Hence, to approximate the posterior we are forced to turn to different Bayesian inferential approaches; this report explores Monte Carlo Markov Chain (MCMC), Laplace approximation and Variational approximation both theoretically and through illustrative datasets.

Acknowledgments

I would like to thank my supervisor Dr. Claire Gormley for her continuous help and support throughout this project. Claire consistently went above and beyond what was expected of a supervisor, helping me navigate challenges and understand complex concepts. I would also like to thank the module coordinator, Dr. Fengnan Gao, for providing the structure and support that enabled this project to take place.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Logistic regression is a fundamental statistical method used for modeling and classifying a binary response variable, where our only two outcomes are success and failure. Logistic regression follows a generalised linear model (GLM), where the random component has a binomial distribution and the link function is the logit function. This ensures that the predicted probabilities remain in the [0,1] range and also accounts for the non-linear relationship between the explanatory variables and the binary response variable, unlike linear regression.

Bayesian analysis is regularly adopted for logistic regression, to provide a full posterior distribution over the parameters, while also allowing for the quantification of uncertainty and the incorporation of prior information. However, unlike linear regression, the posterior distribution for logistic regression is analytically intractable due to the exponential form of the likelihood and the lack of a conjugate prior. Therefore, to approximate the posterior we are forced to turn to different Bayesian inferential approaches.

Multiple methods have been proposed to approximate the intractable Bayesian posterior. Early work includes the Laplace approximation, which provides a fast Gaussian approximation of the true posterior, introduced for the logistic model by Tierney and Kadane (1986). Monte Carlo Markov Chain was later introduced in this context to attempt to provide more accurate estimates, by generating samples directly from the intractable posterior, as discussed in Chib and Greenberg (1995). More recently, variational approximation has gained popularity in logistic regression thanks to both its speed and flexibility, with Jaakkola and Jordan (2000) laying the foundation for its application in this context.

These Bayesian inferential approaches are now widely recognised and incorporated in modern statistical and machine learning frameworks, as can be observed in Bishop (2006) and Agresti (2007). Our primary objective is to compare and contrast the different Bayesian approaches to logistic regression, discussing and implementing them to highlight their respective advantages and limitations.

1.2 The Logistic Regression Model

The logistic regression model utilses the logit function with a binomial random component to model and predict binary data. Let $\underline{x}_i = (x_{i1}, \ldots, x_{ip})$ denote the values of p explanatory variables for observation i for $i = 1, \ldots, N$. Then the logit function of our probability of success π for each observation can be

shown as:

$$\begin{aligned} \text{logit}\left[\pi(\underline{x}_i)\right] &= \log\left(\frac{\pi(\underline{x}_i)}{1-\pi(\underline{x}_i)}\right) \\ &= \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} \\ \Rightarrow \quad \pi(\underline{x}_i) &= \frac{\exp(\alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})}{1 + \exp(\alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})} \\ &= \frac{\exp\left(\alpha + \sum_{j=1}^{p} \beta_j x_{ij}\right)}{1 + \exp\left(\alpha + \sum_{j=1}^{p} \beta_j x_{ij}\right)} \end{aligned} \tag{1.2.1}$$

This logistic model is preferred to the regular linear regression model when dealing with a binary response variable to capture the non-linear effects of the explanatory variables on π. Hence, a fixed change on $\underline{x}_i$ may have a different effect on π depending on whether π is close to 0 or 1, or if it's value is middle of the range (Agresti 2007).

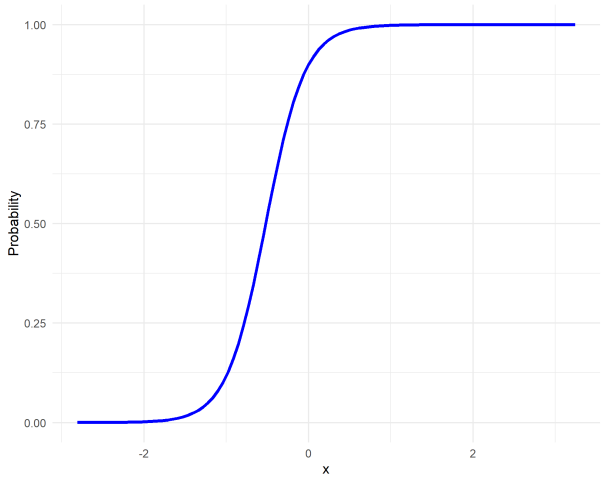

Figure 1.1: Univariate Logistic Function Plot

Figure 1.1 showcases the non-linear nature of the logistic model. Therefore, not only is there no closed-form solution for the Bayesian posterior, but there's also no closed-form solution for the maximum likelihood equation (MLE). Hence, we cannot solve it analytically so we must rely on numerical optimisation techniques for the logistic regression MLE, such as the Newton-Raphson algorithm.

Chapter 2

Methods and Theory

2.1 MLE - The Newton-Raphson Algorithm

The Netwton-Raphson Algorithm is an iterative optimisation tool, which can allow us to estimate the MLE of the parameters of the logistic model. It starts by choosing an initial guess at the solution. It then approximates the function to be optimised in a neighbourhood of the initial guess using a quadratic polynomial, and then maximises the polynomial to give its second guess. The algorithm continues in this way generating a sequence of guesses. The sequence converges to the true maximum if the function is well behaved and/or the initial guess is good.

For our logistic regression model with parameters $\underline{\boldsymbol{\beta}} = (\alpha, \beta_1, \ldots, \beta_p) = (\beta_0, \beta_1, \ldots, \beta_p)$ and log-likelihood function $\log \mathbb{L}(\underline{\boldsymbol{\beta}})$ which we wish to maximise, the Newton-Raphson algorithm can be written as:

$$\underline{\boldsymbol{\beta}}^{(t+1)} = \underline{\boldsymbol{\beta}}^{(t)} - \left[H^{(t)}\right]^{-1} \underline{u}^{(t)} \tag{2.1.1}$$

where:

- H is the Hessian matrix:
$$H = [h_{kp}] = \left[\frac{\partial^2 \log \mathbb{L}(\underline{\boldsymbol{\beta}})}{\partial \beta_k \partial \beta_l}\right]$$
for $k, l = 0, \ldots, p$.
- $\underline{u}$ is the gradient vector:
$$\underline{u} = \left(\frac{\partial \log \mathbb{L}(\underline{\boldsymbol{\beta}})}{\partial \beta_0}, \ldots, \frac{\partial \log \mathbb{L}(\underline{\boldsymbol{\beta}})}{\partial \beta_p}\right)$$
- $\underline{u}^{(t)}$ and $H^{(t)}$ are $\underline{u}$ and H evaluated at the guess for $\underline{\boldsymbol{\beta}}$ at iteration t of the algorithm.

We can let $\alpha = \boldsymbol{\beta}_0$ and $x_{i0} = 1$ such that $\exp(\alpha + \sum_{j=1}^{p} \beta_j x_{ij}) = \exp(\sum_{j=0}^{p} \beta_j x_{ij})$.
Hence, our logistic model from (1.1) becomes:

$$\pi(\underline{x}_i) = \frac{\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)}{1 + \exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)} \tag{2.1.2}$$

Denote the binary response data as $Y_1, \ldots, Y_N$, leading to the likelihood function of our logistic model:

$$\mathbb{L}(\underline{\boldsymbol{\beta}}) = \prod_{i=1}^{N} \pi(\underline{x}_i)^{y_i}[1-\pi(\underline{x}_i)]^{1-y_i}$$

Therefore, our log-likelihood function is:

$$\begin{aligned}
\log \mathbb{L}(\underline{\boldsymbol{\beta}}) &= \sum_{i=1}^{N} \left[y_i \log \pi(\underline{x}_i) + (1-y_i)\log(1-\pi(\underline{x}_i)) \right] \\
&= \sum_{i=1}^{N} \left[y_i \log \left(\frac{\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)}{1+\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)} \right) + (1-y_i)\log\left(\frac{1}{1+\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)} \right) \right] \\
&= \sum_{i=1}^{N} \left[y_i \sum_{j=0}^{p} \beta_j x_{ij} - \log\left(1+\exp\left(\sum_{j=0}^{p} \beta_j x_{ij} \right) \right) \right] \qquad (2.1.3)
\end{aligned}$$

By finding the first derivative of the log-likelihood with respect to parameters β_k for $k = 0, \ldots, p$ we can find the entries to the vector $\underline{u}$:

$$\begin{aligned}
\frac{\partial \log \mathbb{L}(\underline{\boldsymbol{\beta}})}{\partial \beta_k} &= \sum_{i=1}^{N} y_i x_{ik} - \sum_{i=1}^{N} \frac{x_{ik} \exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)}{1+\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)} \\
&= \sum_{i=1}^{N} y_i x_{ik} - \sum_{i=1}^{N} x_{ik} \hat{\pi}(\underline{x}_i) \\
&= \sum_{i=1}^{N} [y_i - \hat{\pi}(\underline{x}_i)] x_{ik} \qquad (2.1.4)
\end{aligned}$$

Now by finding the second derivative with respect to parameters β_l for $l = 0, \ldots, p$ we can find the entries to the Hessian Matrix H:

$$\begin{aligned}
\frac{\partial^2 \log \mathbb{L}(\underline{\boldsymbol{\beta}})}{\partial \beta_k \partial \beta_l} &= -\sum_{i=1}^{N} \left\{ \frac{[1+\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)] x_{ik} x_{il} \exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)}{[1+\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)]^2} \right\} \\
&\quad + \sum_{i=1}^{N} \left\{ \frac{x_{ik} \exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right) x_{il} \exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)}{[1+\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)]^2} \right\} \\
&= -\sum_{i=1}^{N} x_{ik} x_{il} \hat{\pi}(\underline{x}_i) + \sum_{i=1}^{N} x_{ik} x_{il} [\hat{\pi}(\underline{x}_i)]^2 \\
&= -\sum_{i=1}^{N} x_{ik} x_{il} \hat{\pi}(\underline{x}_i)[1-\hat{\pi}(\underline{x}_i)] \qquad (2.1.5)
\end{aligned}$$

The Newton-Raphson Algorithm can be a quick and effective method for finding the MLE estimates in the logistic model. However, the MLE only provides us with a singular point-estimate $\hat{\beta}_{\text{MLE}}$. This is why our preferred method is a Bayesian inferential approach, which provides us with an approximation of the full Bayesian posterior distribution.

2.2 The Bayesian Posterior

We wish to find the full Bayesian distribution for the logistic model, which allows us to quantify the uncertainty and incorporate prior information in our model, unlike the Newton-Raphson MLE estimates. However, the posterior distribution is intractable due to the exponential form of the likelihood and lack of a conjugate prior.

To approximate the posterior distribution, we first need to find the posterior formula (or equivalently the log-posterior in this case), by doing so we can also show that this result has no closed-form solution.

Using Bayes Theorem:

$$p(\underline{\boldsymbol{\beta}} \mid \mathbf{y}) \propto p(\mathbf{y} \mid \underline{\boldsymbol{\beta}})\, p(\underline{\boldsymbol{\beta}})$$
$$\log p(\underline{\boldsymbol{\beta}} \mid \mathbf{y}) \propto \log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}}) + \log p(\underline{\boldsymbol{\beta}})$$

The log-likelihood $\log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}})$ has already been derived in (2.1.3):

$$\log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}}) \propto \sum_{i=1}^{N} \left[y_i \sum_{j=0}^{p} \beta_j x_{ij} - \log \left(1 + \exp \left(\sum_{j=0}^{p} \beta_j x_{ij} \right) \right) \right]$$

We can assume an independent unspecified Normal prior $\beta_j \ \forall j = 1, \ldots, p$. Where:

$$\beta_j \sim \mathcal{N}(\mu_0, \sigma_0^2)$$

Hence, our prior distribution is:

$$p(\underline{\boldsymbol{\beta}}) = \prod_{j=0}^{p} \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(\beta_j - \mu_0)^2}{2\sigma_0^2} \right)$$

Taking the log-prior:

$$\begin{aligned} \log p(\underline{\boldsymbol{\beta}}) &= -\frac{1}{2} \sum_{j=0}^{p} \log\left(2\pi\sigma_0^2\right) - \sum_{j=0}^{p} \frac{(\beta_j - \mu_0)^2}{2\sigma_0^2} \\ &\propto -\sum_{j=0}^{p} \frac{(\beta_j - \mu_0)^2}{2\sigma_0^2} \end{aligned}$$

Therefore, the log-posterior for the logistic model is:

$$\begin{aligned} \log p(\underline{\boldsymbol{\beta}}|y) &\propto \log p(y|\underline{\boldsymbol{\beta}}) + \log p(\underline{\boldsymbol{\beta}}) \\ &\propto \sum_{i=1}^{N} \left[y_i \sum_{j=0}^{p} \beta_j x_{ij} - \log \left(1 + \exp \left(\sum_{j=0}^{p} \beta_j x_{ij} \right) \right) \right] - \sum_{j=0}^{p} \frac{(\beta_j - \mu_0)^2}{2\sigma_0^2} \end{aligned} \tag{2.2.1}$$

Therefore, the log-posterior has no closed-form solution and is intractable, so we must rely on different Bayesian inferential approaches to approximate it.

2.3 Monte Carlo Markov Chain (MCMC)

The first Bayesian approach we will look at is Monte Carlo Markov Chain (MCMC). MCMC generates a chain of dependent samples through probabilistic acceptance, which will theoretically converge to the true posterior over time.

To approximate our logistic regression posterior distribution we will use the Metropolis-Hastings algorithm within MCMC, firstly using parameter-wise updates as follows:

We firstly choose our initial starting values $\beta_j^{(0)}$ for $j = 1, \ldots, p$.

Then for iterations $t = 1, 2, 3, \ldots, T$ and each regression coefficient β_j $j = 1, \ldots, p$:

1. We sample a new proposal β_j^* from a proposal distribution centered at the previous sample $\beta_j^{(t-1)}$:

$$\beta_j^* \sim \mathcal{N}(\beta_j^{(t-1)}, \sigma_{jj}^2)$$

Here we will use a Normal proposal distribution where the proposal variance σ_{jj}^2 is the diagonal entries of our Newton-Raphson MLE covariance matrix, i.e. $\sigma_{jj}^2 = \text{diag}(\Sigma_{\text{MLE}})$. Where:

$$\Sigma_{\text{MLE}} = \left[-H(\underline{\boldsymbol{\beta}})\right]^{-1}$$

The proposal distribution at time t is denoted as $J_t(\beta_j^*|\beta_j^{(t-1)})$.

2. We compute the log acceptance rate R:

$$\begin{aligned} R &= \log\left(\frac{p(\beta_j^*|y)J_t(\beta_j^{(t-1)}|\beta_j^*)}{p(\beta_j^{(t-1)}|y)J_t(\beta_j^*|\beta_j^{(t-1)})}\right) \\ &= \log\left(\frac{p(\beta_j^*|y)}{p(\beta_j^{(t-1)}|y)}\right) \\ &= \log p(\beta_j^*|y) - \log p(\beta_j^{(t-1)}|y) \end{aligned}$$

Where $\log p(.|y)$ is our log-posterior of the logistic model derived in (2.2.1).

Note: We are using a symmetric Normal proposal distribution, so:

$$J_t(\beta_j^{(t-1)}|\beta_j^*) = J_t(\beta_j^*|\beta_j^{(t-1)})$$

3. Draw a random uniform number $u \sim U(0, 1)$. If $\log(u) < R$, we accept β_j^* and set:

$$\beta_j^{(t)} = \beta_j^*$$

Otherwise, we reject β_j^* and keep the previous sample:

$$\beta_j^{(t)} = \beta_j^{(t-1)}$$

We can then finally compute the posterior mean estimate for each β_j by averaging over the MCMC samples for each iteration t:

$$\hat{\beta}_j = \frac{1}{T}\sum_{t=1}^{T}\beta_j^{(t)}$$

We can also quantify the uncertainty of our mean estimates by finding the varaince of our MCMC samples:

$$\widehat{\text{Var}}(\beta_j) = \frac{1}{T-1}\sum_{t=1}^{T}\left(\beta_j^{(t)} - \hat{\beta}_j\right)^2$$

This process uses **parameter-wise updates**, as we are updating each parameter independently. However, we can also perform **block updates**, where at each iteration t we choose to update the whole block $\underline{\boldsymbol{\beta}}$. This process follows the same steps using $\underline{\boldsymbol{\beta}}$ where the proposal distribution is:

$$\underline{\boldsymbol{\beta}}^* \sim \mathcal{N}(\underline{\boldsymbol{\beta}}^{(t-1)}, \Sigma_{\text{MLE}})$$

These two processes are compared further in Chapter 3.

MCMC is a very effective Bayesian sampling approach which will theoretically always converge to the true posterior over time, assuming certain conditions are met (e.g. the proposal distribution is irreducible). However, it also assumes that the chain is ran over infinite iterations, which is obviously not possible in practice. Therefore, to get a accurate approximation we often have to run the chain over a large amount of iterations, which can be computationally expensive, especially when n and/ or p is large.

In the following sections we will examine alternative methods which allow us to approximate this true posterior without the high computational expense.

2.4 Laplace Approximation

The Laplace Approximation is a simple technique which allows us to find a Gaussian approximation to the true posterior of our logistic model. The advantage of using the Laplace Approximation over MCMC is that there is very little computational expense, as we are not sampling from the posterior but rather making a simplified approximation. However, as this is an approximation we do not get the exact full posterior, so we have to examine our given solution to find its accuracy.

The Laplace Approximation is obtained by first finding the mode of the log-posterior of our logistic model. We can again assume a Normal prior so can take the result derived previously in (2.2.1):

$$\log p(\underline{\boldsymbol{\beta}}|y) \propto \sum_{i=1}^{N}\left[y_i\sum_{j=0}^{p}\beta_j x_{ij} - \log\left(1+\exp\left(\sum_{j=0}^{p}\beta_j x_{ij}\right)\right)\right] - \sum_{j=0}^{p}\frac{(\beta_j-\mu_j)^2}{2\sigma_j^2}$$

To find the mode, we maximise the the log-posterior to give the MAP (maximum a posteriori) solution $\underline{\boldsymbol{\beta}}_{MAP}$ which defines the mean of the Gaussian approximation.

$$\underline{\boldsymbol{\beta}}_{MAP} = \arg\max_{\beta_k} \log p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})$$

Hence, to find the maximum we first differentiate the log-posterior with respect to β_k:

$$\begin{aligned}\frac{d \log p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})}{d\beta_k} &= \sum_{i=1}^{N} y_i x_{ik} - \sum_{i=1}^{N} \frac{\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)}{1 + \exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)} x_{ik} - \frac{(\beta_k - \mu_k)}{\sigma_k^2} \\ &= \sum_{i=1}^{N} [y_i - \hat{\pi}(\underline{x}_i) x_{ik}] - \frac{(\beta_k - \mu_k)}{\sigma_k^2} \end{aligned} \tag{2.4.1}$$

Then, setting to 0 and solving for β_k:

$$\begin{aligned}\sum_{i=1}^{N} [y_i - \hat{\pi}(\underline{x}_i)] x_{ik} - \frac{(\beta_k - \mu_k)}{\sigma_k^2} &= 0 \\ \beta_k - \mu_k &= \sigma_k^2 \sum_{i=1}^{N} [y_i - \hat{\pi}(\underline{x}_i)] x_{ik} \\ \beta_k &= \sigma_k^2 \sum_{i=1}^{N} [y_i - \hat{\pi}(\underline{x}_i)] x_{ik} + \mu_k \end{aligned}$$

Here $\hat{\pi}(\underline{x}_i)$ is the estimate of our logistic function from (2.1.2), which depends on the parameter β_k. Therefore, there is no closed-form solution for the MAP estimate.

Hence, to estimate $\underline{\boldsymbol{\beta}}_{MAP}$ we must solve the above system of nonlinear equations using an iterative method, such as Newton–Raphson.

From (2.1.1), we defined the Newton-Raphson algorithm as:

$$\underline{\boldsymbol{\beta}}^{(t+1)} = \underline{\boldsymbol{\beta}}^{(t)} - \left[H^{(t)}\right]^{-1} \underline{u}^{(t)}$$

However, we are now trying to maximize the log-posterior $\log p(\underline{\boldsymbol{\beta}}|y)$ rather than the likelihood function $L(\underline{\boldsymbol{\beta}})$.

Therefore, the gradient vector $\underline{u}$ now becomes:

$$\underline{u} = \left(\frac{\partial \log p(\underline{\boldsymbol{\beta}}|y)}{\partial \beta_0}, \ldots, \frac{\partial \log p(\underline{\boldsymbol{\beta}}|y)}{\partial \beta_p} \right)$$

for $k = 0, \ldots, p$.

Whose entries have already been derived in (2.4.1).

And the Hessian matrix H becomes:

$$H = [h_{kl}] = \left[\frac{\partial^2 \log p(\underline{\boldsymbol{\beta}} \mid y)}{\partial \beta_k \, \partial \beta_l} \right]$$

for $k, l = 0, \ldots, p$.

We can derive the entries for the Hessian matrix by finding the second derivative of (2.4.1) with respect to β_l:

$$\frac{\partial^2 \log p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})}{\partial \beta_k \, \partial \beta_\ell} = \begin{cases} -\sum_{i=1}^{N} x_{ik} x_{i\ell} \, \hat{\pi}(\underline{x}_i)[1 - \hat{\pi}(\underline{x}_i)], & \text{for } k \neq \ell \\ -\sum_{i=1}^{N} x_{ik}^2 \, \hat{\pi}(\underline{x}_i)[1 - \hat{\pi}(\underline{x}_i)] - \frac{1}{\sigma_k^2}, & \text{for } k = \ell \end{cases} \tag{2.4.2}$$

Once we have found $\underline{\boldsymbol{\beta}}_{MAP}$ using the Newton-Raphson algorithm, we make a second-order Taylor expansion to $\log p(\underline{\boldsymbol{\beta}}|y)$ around this point:

$$\begin{aligned} \log p(\underline{\boldsymbol{\beta}} \mid \mathbf{y}) &\approx \log p(\underline{\boldsymbol{\beta}}_{MAP} \mid \mathbf{y}) + (\underline{\boldsymbol{\beta}} - \underline{\boldsymbol{\beta}}_{MAP})^\top \underline{u} + \frac{1}{2}(\underline{\boldsymbol{\beta}} - \underline{\boldsymbol{\beta}}_{MAP})^\top H(\underline{\boldsymbol{\beta}})(\underline{\boldsymbol{\beta}} - \underline{\boldsymbol{\beta}}_{MAP}) \\ &\approx \log p(\underline{\boldsymbol{\beta}}_{MAP} \mid \mathbf{y}) + \frac{1}{2}(\underline{\boldsymbol{\beta}} - \underline{\boldsymbol{\beta}}_{MAP})^\top H(\underline{\boldsymbol{\beta}} - \underline{\boldsymbol{\beta}}_{MAP}) \end{aligned}$$

Note: The first-order term in the Taylor expansion goes to zero as we expand around the maximum, where the gradient vector $\underline{u}$ is zero.

By exponentiating, we get an approximation to the posterior distribution:

$$p(\underline{\boldsymbol{\beta}} \mid \mathbf{y}) \approx p(\underline{\boldsymbol{\beta}}_{MAP} \mid \mathbf{y}) \exp\left(-\frac{1}{2}(\underline{\boldsymbol{\beta}} - \underline{\boldsymbol{\beta}}_{MAP})^\top H(\underline{\boldsymbol{\beta}} - \underline{\boldsymbol{\beta}}_{MAP})\right)$$

where the covariance matrix Σ_L takes the form:

$$\Sigma_L = \left[-H(\underline{\boldsymbol{\beta}}_{MAP})\right]^{-1}$$

Therefore, the Gaussian approximation to the posterior distribution is:

$$p(\underline{\boldsymbol{\beta}} \mid \mathbf{y}) \approx \mathcal{N}(\underline{\boldsymbol{\beta}}_{MAP}, \Sigma_L)$$

Hence, under the Laplace approximation, the mean parameter estimates are given by $\underline{\boldsymbol{\beta}}_{MAP}$ and their uncertainty is approximated by the covariance matrix Σ_L, as also shown in Bishop (2006).

However, as Σ_L represents the covariance matrix of the approximated posterior distribution rather than the true posterior $p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})$, this underestimates the posterior uncertainty. This limitation reflects a key drawback of the Laplace approximation, its inability to accurately quantify uncertainty.

Another prominent drawback of the Laplace approximation is its local nature, as the approximation of a parameter is centred around one point $\boldsymbol{\beta}_{MAP}$ it provides us with a local approximation and ignores the globabl structure. This can lead to severely underestimating uncertainty away from the mode, especially when the true logistic regression posterior deviates substantially from a Gaussian shape around its maximum.

2.5 Variational Approximation

Our final Bayesian inferential approach is Variational Approximation, which works similarly to the Laplace framework for logistic regression but aims to improve accuracy due to its increased flexibility. This flexibility attempts to create a global posterior approximation, addressing a key limitation of the Laplace method.

Variational approximation works by approximating the intractable posterior distribution using a simpler distribution $q(\underline{\boldsymbol{\beta}})$, which minimises the Kullback-Leibler (KL) divergence between the variational approximation $q(\underline{\boldsymbol{\beta}})$ and the true posterior $p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})$:

$$q^*(\underline{\boldsymbol{\beta}}) = \arg\min_q \mathrm{KL}\left[q(\underline{\boldsymbol{\beta}}) \,\|\, p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})\right]$$

Equivalently, we can maximise the Evidence Lower Bound (ELBO) $\mathcal{L}(q)$, where:

$$\mathcal{L}(q) = \mathbb{E}_q\left[\log p(\mathbf{y}, \underline{\boldsymbol{\beta}})\right] - \mathbb{E}_q\left[\log q(\underline{\boldsymbol{\beta}})\right] \tag{2.5.1}$$

This can be shown by the definitions of ELBO and the KL-divergence:

$$\begin{aligned}
\mathrm{KL}\left(q(\underline{\boldsymbol{\beta}}) \,\|\, p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})\right) &= \mathbb{E}_q\left[\log q(\underline{\boldsymbol{\beta}})\right] - \mathbb{E}_q\left[\log p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})\right] \\
&= \mathbb{E}_q\left[\log q(\underline{\boldsymbol{\beta}})\right] - \mathbb{E}_q\left[\log p(\mathbf{y}, \underline{\boldsymbol{\beta}}) - \log p(\mathbf{y})\right] \\
&= \mathbb{E}_q\left[\log q(\underline{\boldsymbol{\beta}})\right] - \mathbb{E}_q\left[\log p(\mathbf{y}, \underline{\boldsymbol{\beta}})\right] + \mathbb{E}_q\left[\log p(\mathbf{y})\right] \\
&= -\mathcal{L}(q) + \log p(\mathbf{y}) \\
&\propto -\mathcal{L}(q)
\end{aligned}$$

$$\Rightarrow \quad q^*(\underline{\boldsymbol{\beta}}) = \arg\min_q \mathrm{KL}\left[q(\underline{\boldsymbol{\beta}}) \,\|\, p(\underline{\boldsymbol{\beta}} \mid \mathbf{y})\right] \; = \arg\max_q \mathcal{L}(q)$$

In order to calculate the ELBO, we require $\mathbb{E}_q\left[\log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}})\right]$, which has no closed-form solution due to its non-linear nature. So instead, we find a linear lower-bound of the log-likelihood.

To achieve this, we first consider the original logistic function from (2.1.2):

$$\pi(\underline{x_i}) = \frac{\exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)}{1 + \exp\left(\sum_{j=0}^{p} \beta_j x_{ij}\right)}$$

To place a lower bound on the function, we require it to be concave such that every tangent to the function sits entirely below the true curve. We can observe this isn't the case for our logistic function, however by taking the log we can convert it into a concave function:

$$\log \pi(\underline{x_i}) = \sum_{j=0}^{p} \beta_j x_{ij} - \log(1 + \exp(\sum_{j=0}^{p} \beta_j x_{ij}))$$

This can be shown to be concave by finding the second derivative, we can let $\eta_i = \sum_{j=0}^{p} \beta_j x_{ij}$:

$$\begin{aligned}
\log \pi(\eta) &= \eta - \log(1 + \exp(\eta)) \\
\frac{\partial \log \pi(\eta)}{\partial \eta} &= 1 - \frac{e^\eta}{1 + e^\eta} \\
\frac{\partial^2 \log \pi(\eta)}{\partial \eta^2} &= -\frac{e^\eta}{1 + e^\eta} \\
&= -\frac{(1 + e^\eta) \cdot e^\eta - e^\eta \cdot e^\eta}{(1 + e^\eta)^2} \\
&= -\frac{e^\eta}{(1 + e^\eta)^2} < 0 \quad \forall \eta \in \mathbb{R}
\end{aligned}$$

$$\Rightarrow \log \pi(\eta) \text{ is concave}$$

This gives us the concave function:

$$f(z) = \log \pi(z) = z - \log(1 + \exp(z))$$

A lower bound on $f(z)$ can be found using a modified version of the second-order Taylor expansion, where the second-order Taylor expansion at any point ξ is:

$$\begin{aligned}
T_\xi(z) &= f(\xi) + f'(\xi)(z - \xi) + f''(\xi)(z^2 - \xi^2) \\
&= \log \pi(\xi) + (1 - \pi(\xi))(z - \xi) - \tfrac{1}{2}\pi(\xi)(1 - \pi(\xi))(z - \xi)^2
\end{aligned}$$

The second-order Taylor expansion matches the concave function $f(z)$ locally, however away from the point ξ the dropped higher-order terms can make the quadratic rise above the true curve.

Therefore, we modify the Taylor expansion using Jaakola and Jordon (2000), which tightens the quadratic term such that it stays below the logistic function everywhere, i.e. it achieves a global lower bound.

Here we replace the quadratic term $f''(\xi) = -\pi(\xi)(1 - \pi(\xi))$ with $-2\lambda(\xi)$:

$$\text{where} \quad \lambda(\xi) = \frac{\pi(\xi) - \frac{1}{2}}{2\xi}$$

.

$$\text{such that} \quad -2\lambda(\xi) \leq f''(z) \quad \forall z$$

Our lower bound then becomes:

$$\begin{aligned}
\log \pi(z) &\geq \log \pi(\xi) + \frac{1}{2}(z - \xi) - \lambda(\xi)(z^2 - \xi^2) \\
\pi(z) &\geq \pi(\xi) \exp\left\{\frac{1}{2}(z - \xi) - \lambda(\xi)(z^2 - \xi^2)\right\} \qquad (2.5.2)
\end{aligned}$$

(2.5.2) shows us the lower bound for the logistic function $\pi(z)$, however we are trying to place a lower bound on our log-likelihood function:

$$
\begin{aligned}
p(y_i \mid \underline{\boldsymbol{\beta}}) &= \pi(\eta_i)^{y_i}[1-\pi(\eta_i)]^{1-y_i} \\
&= \left(\frac{e^{\eta_i}}{1+e^{\eta_i}}\right)^{y_i}\left(1-\frac{e^{\eta_i}}{1+e^{\eta_i}}\right)^{1-y_i} \\
&= e^{\eta_i y_i}\frac{e^{-\eta_i}}{1+e^{-\eta_i}} \\
&= e^{\eta_i y_i}\pi(-\eta_i) \\
&\geq e^{\eta_i y_i}\pi(\xi_i)\exp\left\{-\frac{1}{2}(\eta_i+\xi_i)-\lambda(\xi)(\eta_i^2-\xi_i^2)\right\} \\
&\geq \pi(\xi_i)\exp\left\{\eta_i y_i-\frac{1}{2}(\eta_i+\xi_i)-\lambda(\xi)(\eta_i^2-\xi_i^2)\right\}
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow \log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}}) &\geq \sum_{i=1}^{N}\left[\log\pi(\xi_i)+\eta_i y_i-\tfrac{1}{2}(\eta_i+\xi_i)-\lambda(\xi_i)(\eta_i^2-\xi_i^2)\right] \\
&\propto \sum_{i=1}^{N}\left[\eta_i(y_i-\tfrac{1}{2})-\lambda(\xi_i)\eta_i^2\right]
\end{aligned} \tag{2.5.3}
$$

Therefore, (2.5.3) gives us the lower bound for the log-likelihood. This also allows us to calculate a lower bound for the log-posterior, where the prior is $\underline{\boldsymbol{\beta}} \sim \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$:

$$
\begin{aligned}
\log p(\underline{\boldsymbol{\beta}} \mid \mathbf{y}) &\propto \log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}}) + \log p(\underline{\boldsymbol{\beta}}) \\
&\geq \sum_{i=1}^{N}\left[\eta_i\left(y_i-\tfrac{1}{2}\right)-\lambda(\xi_i)\eta_i^2\right]-\frac{1}{2}(\underline{\boldsymbol{\beta}}-\boldsymbol{\mu}_0)^T\Sigma_0^{-1}(\underline{\boldsymbol{\beta}}-\boldsymbol{\mu}_0)
\end{aligned}
$$

Since this is a quadratic function of $\underline{\boldsymbol{\beta}}$, the optimal variational distribution $q(\underline{\boldsymbol{\beta}})$ must be a Gaussian of the form:

$$
q(\underline{\boldsymbol{\beta}}) \approx \mathcal{N}(\mathbf{m}_N, S_N)
$$

Therefore, to find parameters m_N and S_N, we maximise the ELBO knowing that our variational approximation distribution $q(\underline{\boldsymbol{\beta}})$ follows a normal distribution. From (2.5.1), the EBLO is:

$$
\begin{aligned}
\mathcal{L}(q) &= \mathbb{E}_q\left[\log p(\mathbf{y}, \underline{\boldsymbol{\beta}})\right] - \mathbb{E}_q\left[\log q(\underline{\boldsymbol{\beta}})\right] \\
&= \mathbb{E}_q\left[\log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}})\right] - \mathrm{KL}\left(q(\underline{\boldsymbol{\beta}}) \,\|\, p(\underline{\boldsymbol{\beta}})\right)
\end{aligned}
$$

We replace the log-likelihood $\log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}})$ with our derived lower-bound from (6.3):

$$
\log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}}) \geq \sum_{i=1}^{N}\left[\log\pi(\xi_i)+(y_i-\tfrac{1}{2})\eta_i-\lambda(\xi_i)(\eta_i^2-\xi_i^2)\right]
$$

$$\begin{aligned}
\mathbb{E}_q\left[\log p(\mathbf{y} \mid \underline{\boldsymbol{\beta}})\right] &\geq \sum_{i=1}^{N}\left[\log \pi(\xi_i) + (y_i - \tfrac{1}{2})\mathbb{E}_q[\eta_i] - \lambda(\xi_i)\left(\mathbb{E}_q[\eta_i^2] - \xi_i^2\right)\right] \\
&\geq \sum_{i=1}^{N}\Big[\log \pi(\xi_i) + (y_i - \tfrac{1}{2})\sum_{j=0}^{p} x_{ij}m_j - \lambda(\xi_i)\big(\sum_{j=0}^{p}\sum_{k=0}^{p} x_{ij}x_{ik}S_{jk} + \big(\sum_{j=0}^{p} x_{ij}m_j\big)^2 - \xi_i^2\big)\Big] \\
&\geq \sum_{i=1}^{N}\Big[\log \pi(\xi_i) \;+\; \left(y_i - \tfrac{1}{2}\right) x_i^\top \mathbf{m} \;-\; \lambda(\xi_i)\left(x_i^\top S x_i + (x_i^\top \mathbf{m})^2 - \xi_i^2\right)\Big]
\end{aligned}$$

And hence the KL divergence between the normal approximation distribution and our normal prior is:

$$\begin{aligned}
\mathrm{KL}\left(q(\underline{\boldsymbol{\beta}}) \,\|\, p(\underline{\boldsymbol{\beta}})\right) &= \mathbb{E}_q\left[\log q(\underline{\boldsymbol{\beta}})\right] - \mathbb{E}_q\left[\log p(\underline{\boldsymbol{\beta}})\right] \\
&= \mathbb{E}_q\left[-\tfrac{p}{2}\log(2\pi) - \tfrac{1}{2}\log|S| - \tfrac{1}{2}(\underline{\boldsymbol{\beta}} - \mathbf{m})^\top S^{-1}(\underline{\boldsymbol{\beta}} - \mathbf{m})\right] \\
&\quad - \mathbb{E}_q\left[-\tfrac{p}{2}\log(2\pi) - \tfrac{1}{2}\log|\Sigma_0| - \tfrac{1}{2}(\underline{\boldsymbol{\beta}} - \boldsymbol{\mu}_0)^\top \Sigma_0^{-1}(\underline{\boldsymbol{\beta}} - \boldsymbol{\mu}_0)\right] \\
&= -\tfrac{1}{2}\log|S\Sigma_0^{-1}| - \tfrac{1}{2}\mathbb{E}_q\left[(\underline{\boldsymbol{\beta}} - \mathbf{m})^\top \Sigma^{-1}(\underline{\boldsymbol{\beta}} - \mathbf{m})\right] \\
&\quad + \tfrac{1}{2}\mathbb{E}_q\left[(\underline{\boldsymbol{\beta}} - \boldsymbol{\mu}_0)^\top \Sigma_0^{-1}(\underline{\boldsymbol{\beta}} - \boldsymbol{\mu}_0)\right] \\
&= -\tfrac{1}{2}\log|S\Sigma_0^{-1}| - \tfrac{1}{2}p + \tfrac{1}{2}\left(\mathrm{tr}(\Sigma_0^{-1}S) + (\mathbf{m} - \boldsymbol{\mu}_0)^\top \Sigma_0^{-1}(\mathbf{m} - \boldsymbol{\mu}_0)\right) \\
&= \tfrac{1}{2}\left(\mathrm{tr}(\Sigma_0^{-1}S) + (\mathbf{m} - \boldsymbol{\mu}_0)^\top \Sigma_0^{-1}(\mathbf{m} - \boldsymbol{\mu}_0) - \log|S\Sigma_0^{-1}| - p\right)
\end{aligned}$$

$$\begin{aligned}
\therefore \mathcal{L}(q) = \sum_{i=1}^{N}\Big[&\log \pi(\xi_i) \;+\; \left(y_i - \tfrac{1}{2}\right) x_i^\top \mathbf{m} \;-\; \lambda(\xi_i)\left(x_i^\top S x_i + (x_i^\top \mathbf{m})^2 - \xi_i^2\right)\Big] \\
&-\tfrac{1}{2}\left(\mathrm{tr}(\Sigma_0^{-1}S) + (\mathbf{m} - \boldsymbol{\mu}_0)^\top \Sigma_0^{-1}(\mathbf{m} - \boldsymbol{\mu}_0) - \log|S\Sigma_0^{-1}| - p\right)
\end{aligned}$$

Therefore, we maximise $\mathcal{L}(q)$ with respect to $\mathbf{m}$, S and ξ, which as observed in Bishop (2006) gives us:

$$\mathbf{m}_N = S_N\left(\Sigma_0^{-1}\mu_0 + \sum_{i=1}^{N}(y_i - \tfrac{1}{2})\mathbf{x}_i\right)$$

$$S_N^{-1} = \Sigma_0^{-1} + 2\sum_{i=1}^{N}\lambda(\xi_n)\mathbf{x}_i\mathbf{x}_i^T$$

$$\xi_n^2 = x_i^\top\left(S_N + \mathbf{m}_N\mathbf{m}_N^\top\right)x_i$$

We can finally perform coordinate-descent until convergence to find our approximate distribution parameters $\mathbf{m}_N$ and S_N, where ξ_n is used to tighten the bound.

Where our Variational approximation of the true posterior is given by:

$$q(\underline{\boldsymbol{\beta}}) \approx \mathcal{N}(\mathbf{m}_N, S_N)$$

As with the Laplace framework, we have again obtained a Gaussian approximation to the posterior distribution. However, the additional flexibility provided by the variational parameters ξ_n leads to

improved accuracy in the approximation (Jaakkola & Jordan 2000).

Again, similarly to Laplace, this is an approximation so we do not get the exact full posterior, this also means that we are unable to accurately quantify the uncertainty of the model. There is however, very little computational expense compared to exact methods such as MCMC.

Chapter 3

Results

3.1 Data & Results Overview

To examine the performance of the stated methods, we can apply them to illustrative datasets in R, here we use two datasets: a simulation study and the inbuilt R Pima Indians Diabetes dataset.

In the simulation study, we generated 1,000 observations with three explanatory variables, forming a 1000×3 matrix of normally distributed values, the true parameter vector was specified as $\underline{\boldsymbol{\beta}} = (\beta_0 = 2, \beta_1 = 4, \beta_2 = 0, \beta_3 = -3)$. We then assigned a binary outcome y to each observation by computing the logistic probability using the simulated data and the known parameter values, and sampling from a binomial distribution. The simulation study allows to accurately assess each methods accuracy by comparing the beta estimates to their known values.

The Pima Indians Diabetes dataset contains 8 health-related predictor variables such as age, BMI, glucose levels etc. for women of Pima Indian descent, along with a binary response indicating whether diabetes was diagnosed. We use this real-world dataset along with the simulation study to evaluate the methods performance on practical, noisy data where the true parameters are unknown.

For the purpose of this results section, we will mainly refer to the simulation study as we can make a direct comparison between our estimates and the true beta values. To measure the performance of each method within the simulation study, we will look at the following 3 performance metrics, along with plots and other diagnostics.

1. Mean Squared Error (MSE):
   The mean squared error calculates how close our beta parameter estimates are compared to the true values, where:

$$\text{MSE} = \sum_{j=0}^{p} \left(\beta_j - \hat{\beta}_j\right)^2$$

2. Run Time (s):
   The run time simply evaluates the computational expense, by calculating how long it takes to perform each method, given in seconds.

3. Misclassification Rate:
   To calculate the misclassification rate, i.e. the proportion of incorrect predictions, we first split our data into training and test data sets (80/ 20 split). We use the training data to predict the beta

values across T iterations/ samples and then predict the binary outcome of each observation in the out-of-sample test data using our posterior predictive probabilities.

Where our posterior predictive probabilities for each test observation are:

$$\hat{p}_i^{(t)} = \frac{\exp\left(\sum_{j=0}^{p} \hat{\beta}_j^{(t)} x_{ij}\right)}{1 + \exp\left(\sum_{j=0}^{p} \hat{\beta}_j^{(t)} x_{ij}\right)}, \quad \text{for } t = 1, \ldots, T \text{ samples}$$

We then average over all our samples giving us the posterior mean predictive probability for each test observation:

$$\bar{p}_i = \frac{1}{T} \sum_{t=1}^{T} \hat{p}_i^{(t)}$$

By averaging over the posterior, we ensure that predictions reflect the uncertainty in the model's parameter estimates.

We then draw binary predictions from the posterior predictive to predict the outcome of each test observation.

$$\hat{y}_i \sim \text{Bernoulli}(\bar{p}_i)$$

And hence, our misclassification rate is calculated as:

$$\text{Misclassification Rate} = \frac{1}{T} \sum_{i=1}^{T} \mathbb{I}(\hat{y}_i \neq y_i)$$

3.2 Results: The Newton Raphson Algorithm

Firstly, we employed the Newton-Raphson algorithm to find the MLE of our beta parameters. Using initial values of $\underline{\beta}^{(0)} = \underline{0}$, we get parameter MLEs of $\underline{\beta}_{\text{MLE}} = (2.26,\ 4.50,\ -0.30,\ -2.32)$. The convergence of these values alongside the true values can be observed in Figure 3.2:

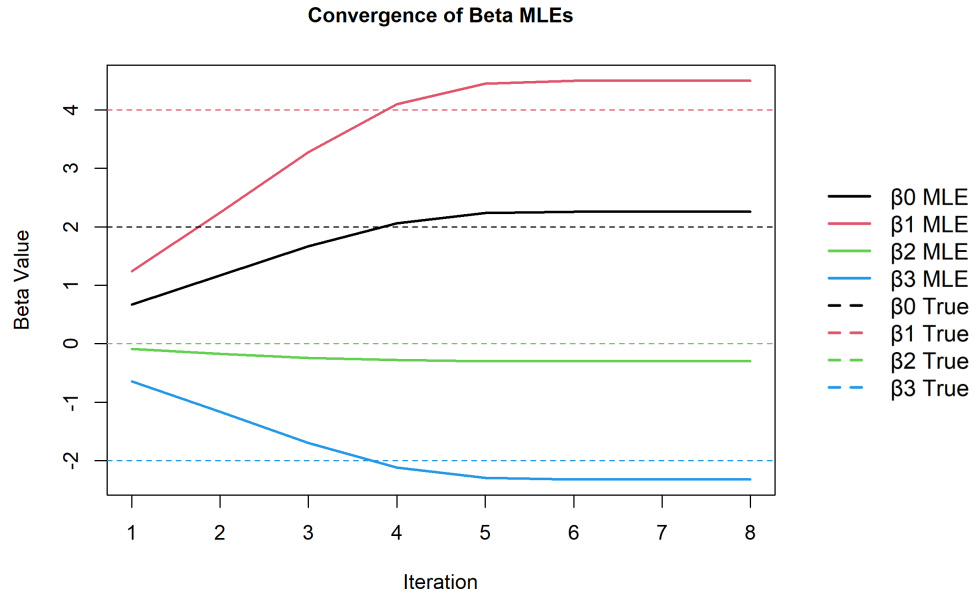

Figure 3.1: Convergence of Beta MLEs in the Simulation Study

Figure 3.2 shows us the convergence of the MLEs from the Newton-Raphson algorithm with the dotted lines representing the true beta values. We can observe a fast convergence, with the initial values converging to the MLE within 8 iterations for all 4 parameters, this took approximately 0.122 seconds to run, showing the algorithms low computational expense.

The MLEs from the Newton-Raphson algorithm gave us a MSE of 0.505 and misclassification rate of 0.155.

3.3 Results: MCMC Parameter-Wise

For all the following Bayesian methods, we assume an independent unspecified Normal $N(0, 100)$ prior for all p parameters. So our log-posterior becomes:

$$\log p(\boldsymbol{\beta} \mid \mathbf{y}) \propto \sum_{i=1}^{N} y_i \left[\sum_{j=0}^{p} \beta_j x_{ij} - \sum_{i=1}^{N} \log \left(1 + \exp \left(\sum_{j=0}^{p} \beta_j x_{ij} \right) \right) \right] - \sum_{j=0}^{p} \frac{\beta_j^2}{200} \tag{3.3.1}$$

The first Bayesian inferential approach we employed was MCMC, we performed MCMC using both block and parameter-wise updates. We firstly performed parameter-wise, i.e. updating one parameter at a time conditioned on the current value of all other parameters.

We ran the MCMC parameter-wise sampler for 10,000 iterations using initial values of $\underline{\beta}^{(0)} = \underline{0}$. We also applied a burn-in of 1,000 iterations, to discard early samples before the chain reaches its target distribution, and thinning of 10, to ensure more independent samples.

The MCMC parameter-wise sampler returns beta parameter estimates of $\underline{\hat{\beta}} = (2.29, 4.56, -0.31, -2.36)$, which gives us a MSE of 0.625 and misclassification rate of 0.140. The marginal posterior density plots of the 4 beta parameters with their true value can be observed in Figure 3.2:

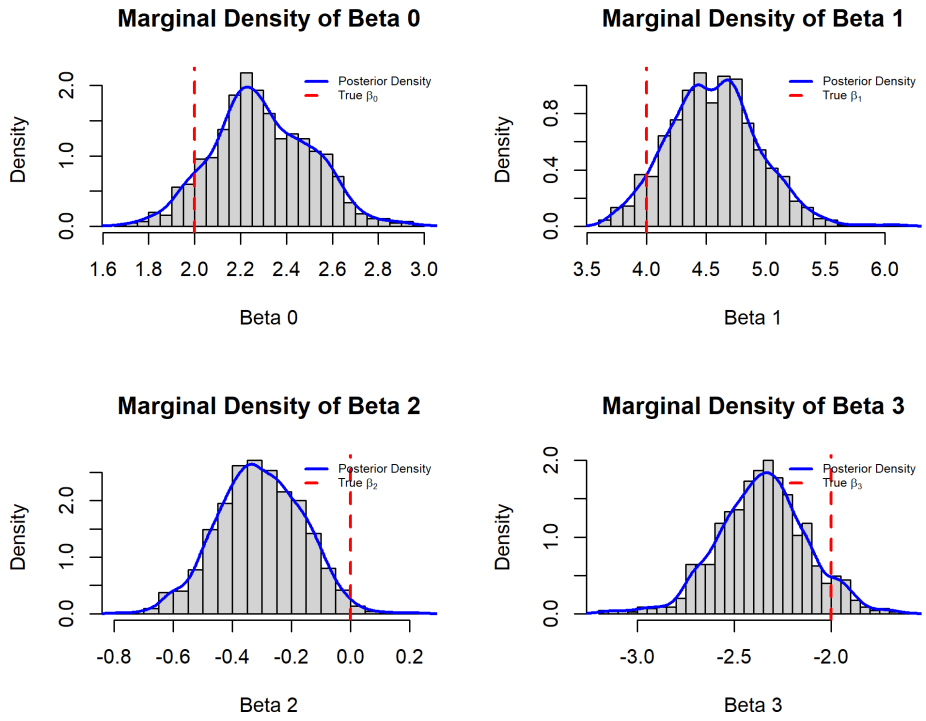

Figure 3.2: MCMC Parameter-Wise Density Plots in the Simulation Study

The run time of the algorithm was 22.358 seconds which is extremely high and highlights a key weakness of the MCMC parameter-wise sampler, as it has to run through each parameter for each iteration one-by-one it becomes very computationally expensive. It also had a very high average acceptance rate of 60.9%, suggesting poor mixing and causing high autocorrelation between the samples.

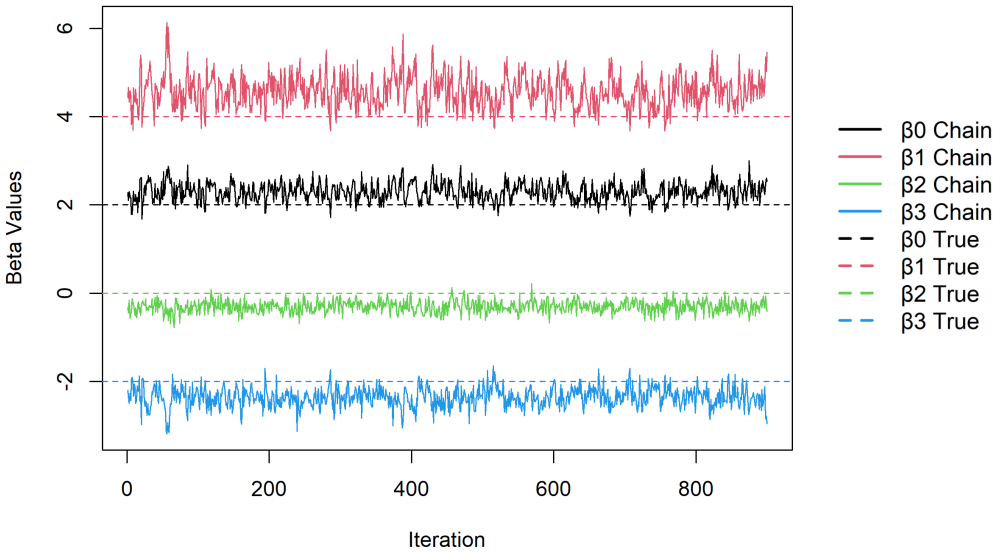

Figure 3.3: MCMC Parameter-Wise Trace Plot in the Simulation Study

Figure 3.3 shows the trace plots for the MCMC parameter-wise in the simulation study, with the true beta values also plotted. We can observe poor convergence and inconsistent mixing across the parameters, suggesting that the posterior estimates may be noisy or biased. We can examine this further by looking at some diagnostic plots.

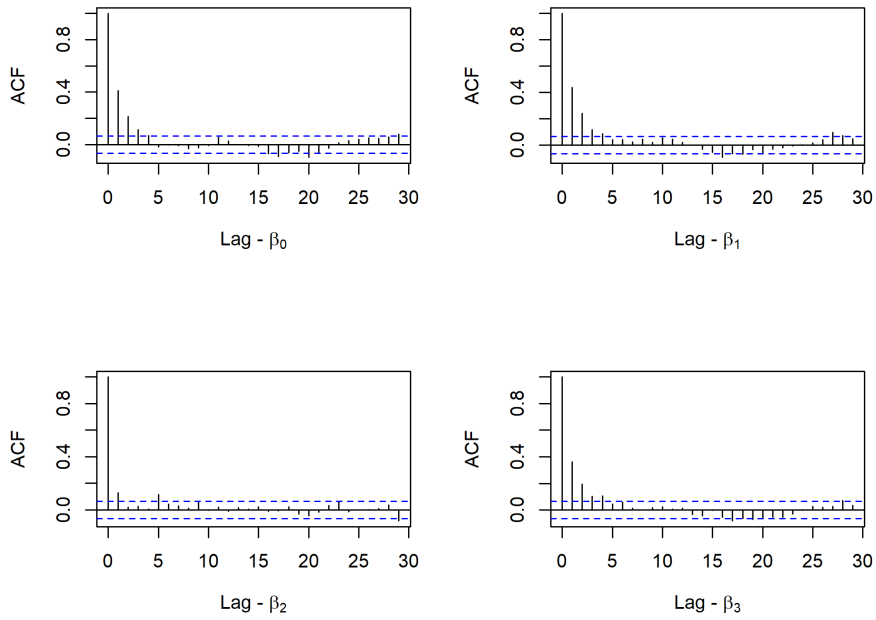

Figure 3.4: MCMC Parameter-Wise ACF Plots in the Simulation Study

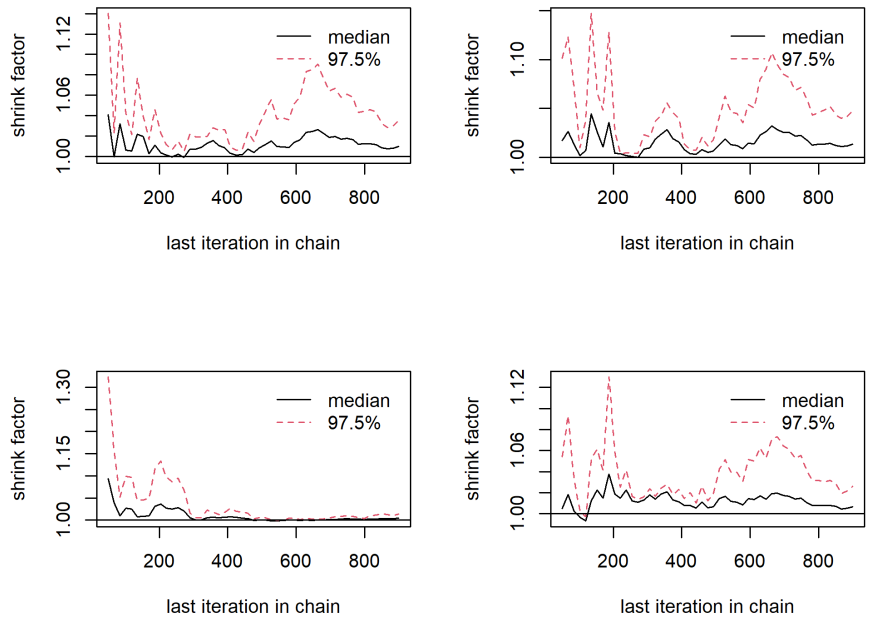

Figure 3.5: MCMC Parameter-Wise Gelman-Ruben Convergence Plots in the Simulation Study

Figures 3.4 and 3.5 show the autocorrelation and convergence, respectively, of the 4 beta parameters in the simulation study. We can observe high autocorrelation for β_0 and β_1 in Figure 3.4 and incomplete convergence for β_0, β_1 and β_3 in Figure 3.5 with elevated $\hat{R}$ values not converging to 1. This suggests overall poor performance for the MCMC parameter-wise sampler.

These problems are even more prominent in the Diabetes dataset, where there exists multicollinearity in the dataset. This can be observed in Figure 3.6.

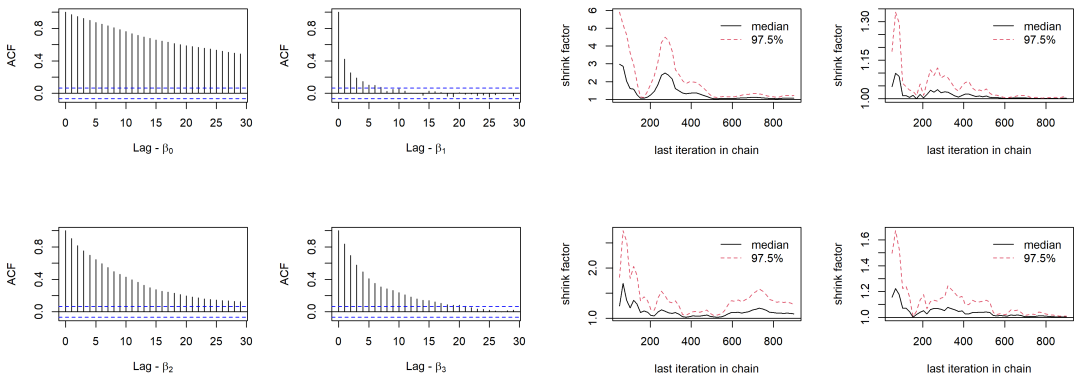

Figure 3.6: MCMC Parameter-Wise ACF and Gelman-Ruben Plots in the Diabetes Dataset, for the first 4 Beta Parameters

3.4 Results: MCMC Block

Next we performed MCMC with block updates, this method works similarly to the parameter-wise updates except we now update the whole 'block' of parameters simultaneously in each update. The aim of doing it this way is to improve on the high autocorrelation and computational costs of the MCMC parameter-wise sampler.

As was the case with the parameter-wise sampler, we ran the block sampler for 10,000 iterations using initial values of $\underline{\beta}^{(0)} = \underline{0}$, with a burn-in of 1,000 and thinning of 10.

The MCMC Block sampler returns beta parameter estimates of $\hat{\underline{\beta}} = (2.29, 4.55, -0.31, -2.35)$, which are nearly identical to that of the parameter-wise sampler. This gives us a MSE of 0.606 and misclassification rate of 0.145. The marginal posterior density plots of the 4 beta parameters with their true value can be observed in Figure 3.7:

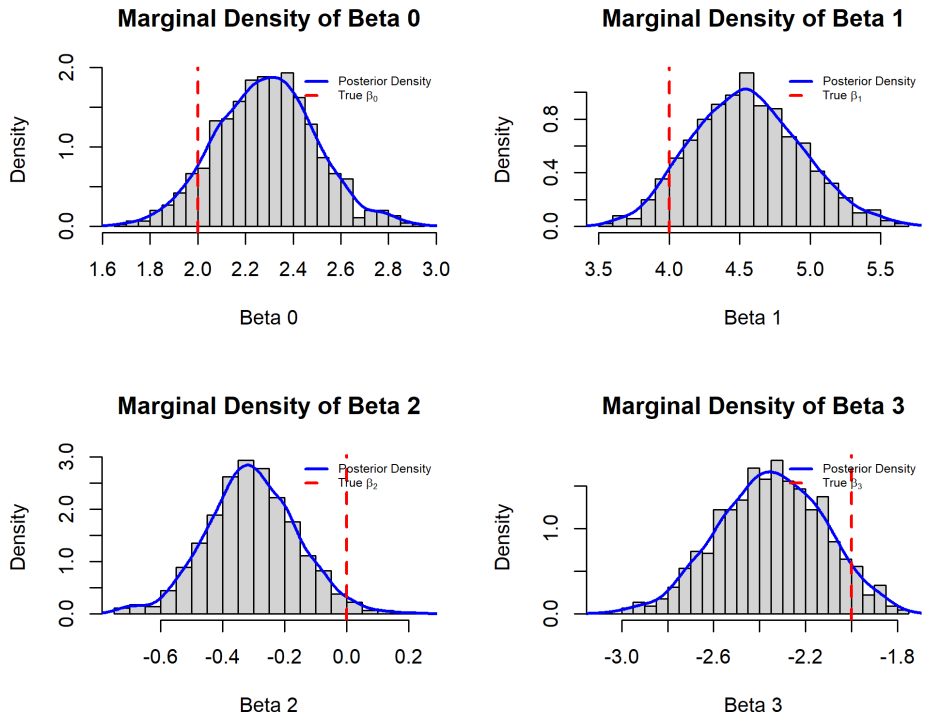

Figure 3.7: MCMC Block Density Plots in the Simulation Study

We can observe that although the MCMC parameter-wise and block methods produce very similar parameter point-estimates, the block method produces a much better overall posterior distribution as seen in Figure 3.7, with a narrower and more symmetric posterior density for all parameters suggesting less uncertainty and better sampling efficiency, compared to Figure 3.2.

The run time for the block sampler was 6.480 seconds, again highlighting a major drawback of MCMC, the high computational expense. Although this is much improved on the parameter-wise sampler, as it doesn't have to individually go through each parameter for each iteration. An acceptance rate of 37.24% of was also observed, an improvement on the high 60.9% acceptance from the parameter-wise sampler, displaying efficient convergence.

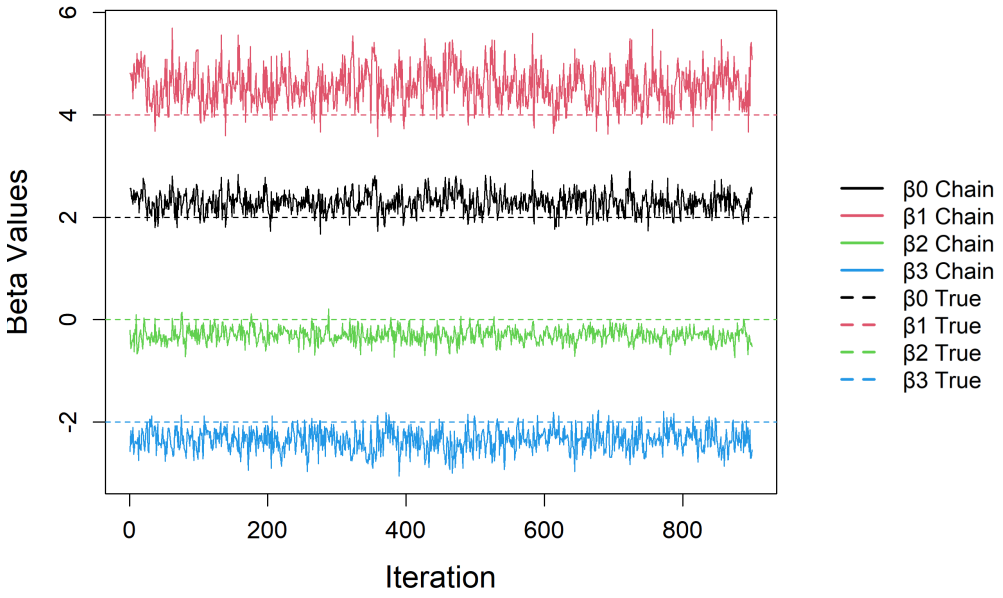

Figure 3.8: MCMC Parameter-Wise Trace Plot in the Simulation Study

Figure 3.8 shows the trace plots for the MCMC block in the simulation study, with the true beta values also plotted. Again, improvements can be observed from the parameter-wise trace plots with better looking convergence and consistent mixing. This can be examined more through the diagnostic plots in Figures 3.9 and 3.10.

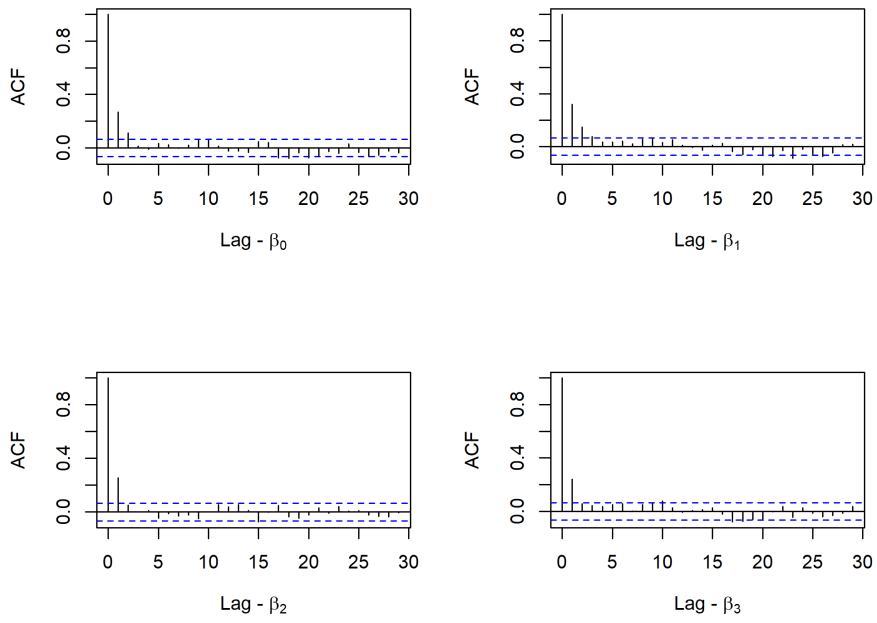

Figure 3.9: MCMC Block ACF Plots in the Simulation Study

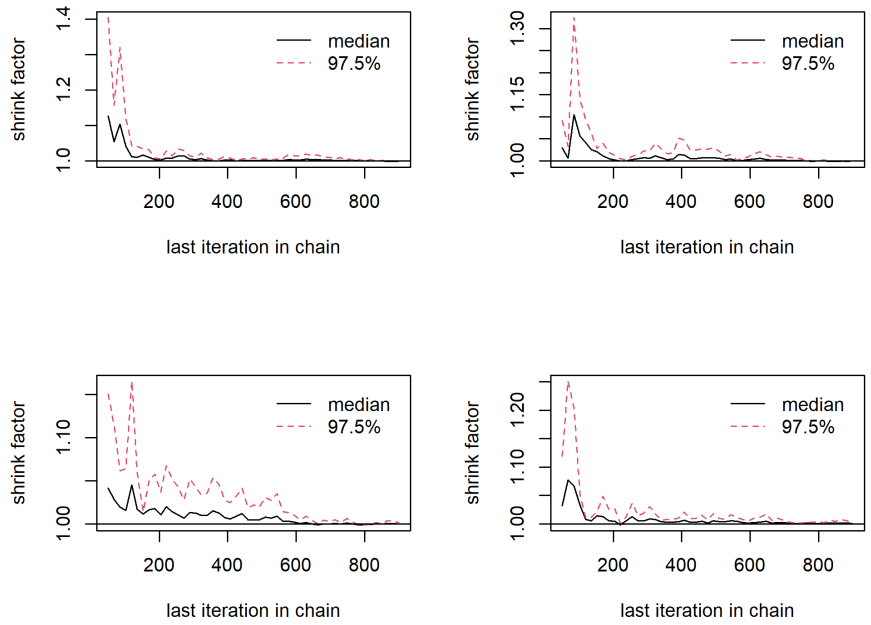

Figure 3.10: MCMC Block Gelman-Ruben Convergence Plots in the Simulation Study

In Figures 3.9 and 3.10, we can observe the ACF drop to near zero after 1-2 lags indicating low autocorrelation, and the $\hat{R}$ converging to 1 indicating good convergence, for all parameters.

These improved diagnostics using the block sampler are also observed in the Diabetes dataset, where there is a high degree of multicollinearity. As can be observed in Figure 3.11 compared to figure 3.6.

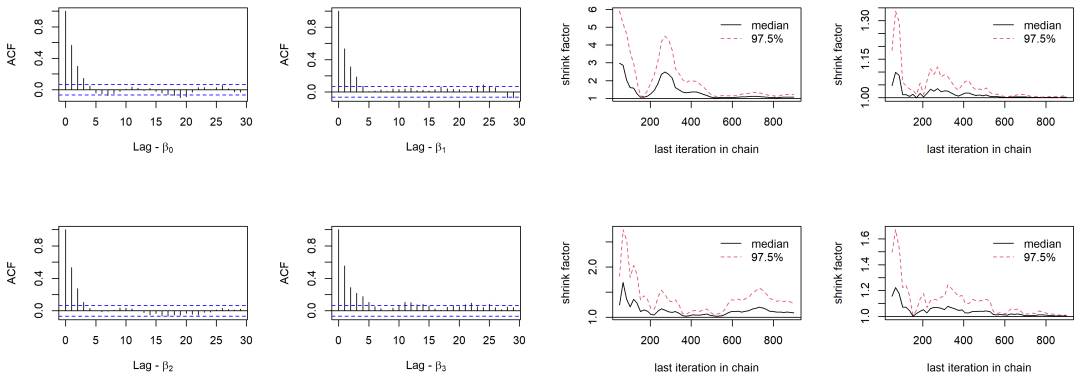

Figure 3.11: MCMC Block ACF and Gelman-Ruben Plots in the Diabetes Dataset, for the first 4 Beta Parameters

3.5 Results: Laplace Approximation

We then performed Laplace Approximation on the simulation study dataset using initial values of $\underline{\beta}^{(0)} = \underline{0}$, returning beta estimates of $\hat{\underline{\beta}} = (2.25, 4.49, -0.30, -2.32)$, which gives us a MSE of 0.490 and misclassification rate of 0.150. We can observe its Gaussian marginal posterior densities in Figure 3.12.

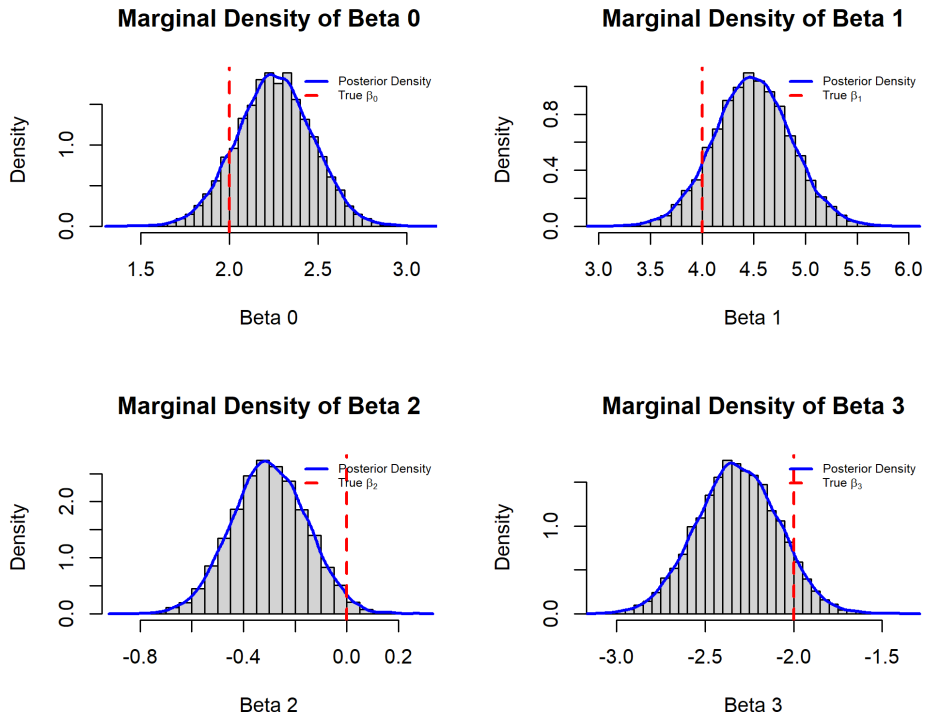

Figure 3.12: Laplace Density Plots for the Simulation Study

Figure 3.12 shows us that the Laplace approximation provides us with good point estimates for the true beta values, however as observed in the tails, it slightly underestimates the uncertainty due to its Gaussian assumption around the MAP.

The run time for the Laplace approximation was 0.138 seconds and took only 8 iterations, showcasing one of its main advantages, its ability to produce an approximation very quickly with little computational expense.

3.6 Results: Variational Approximation

Finally, we performed Variational approximation, which returned beta estimates of $\underline{\hat{\beta}} = (2.27, 4.52, -0.30, -2.33)$, which gives us a MSE of 0.549 and misclassification rate of 0.155. Similarly to Laplace, we get a Gaussian posterior density for our parameters as observed in Figure 3.13.

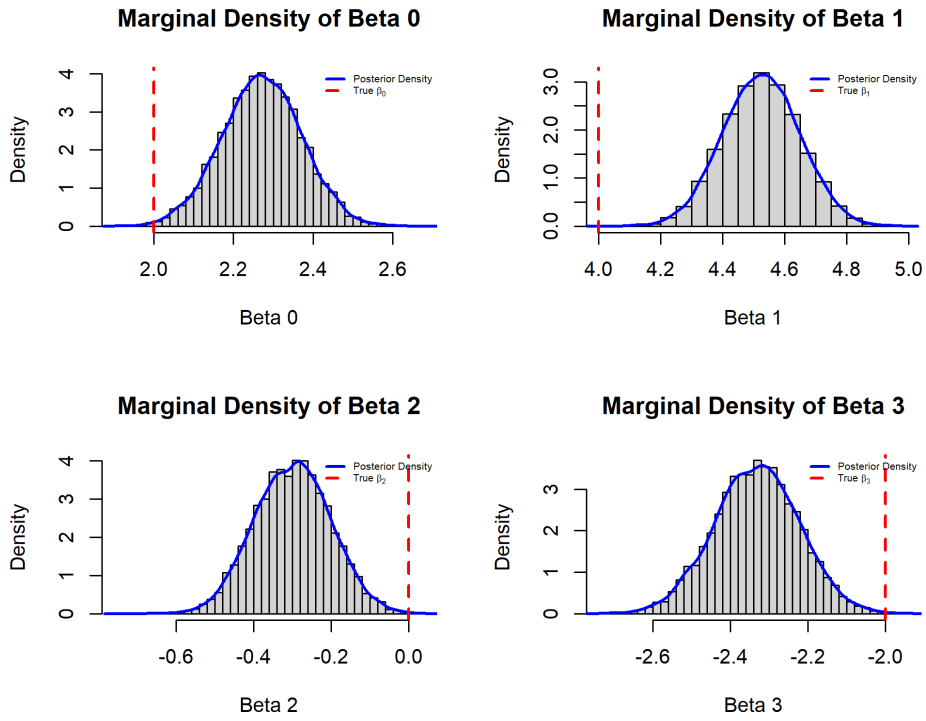

Figure 3.13: Variational Density Plots for the Simulation Study

Similarly to Laplace, we can observe in Figure 3.13 that the Variational approximation fails to accurately capture uncertainty, and is biased towards the mode. Despite its global estimation framework, it actually performs worse than Laplace in this case, producing very concentrated distributions with the true parameter values falling at the very end of the tails.

However, it took only 0.088 seconds to run with 130 iterations to maximise its parameters, showing variational approximation's minimal computational expense similarly to the Laplace approximation.

3.7 Results Summary

We can summarise the results for both the Simulation Study and Diabetes dataset in tables 3.1 and 3.2:

Table 3.1: Simulation Study Results Table

Method	MSE	Run Time (s)	Misclassification Rate
Newton-Raphson MLE	0.505	0.122	0.155
MCMC - Block	0.602	6.480	0.145
MCMC - Parameter-Wise	0.625	22.358	0.140
Laplace Approximation	0.490	0.138	0.150
Variational Approximation	0.549	0.088	0.155

Table 3.2: Pima Indians Diabetes Results Table

Method	**Run Time (s)**	**Misclassification Rate**
Newton-Raphson MLE	0.085	0.325
MCMC - Block	5.408	0.338
MCMC - Parameter-Wise	58.484	0.299
Laplace Approximation	0.079	0.286
Variational Approximation	0.039	0.299

Therefore, from Tables 3.1 and 3.2, and Sections 3.2 - 3.6 we can summarise the results from the Bayesian methods:

- The Laplace approximation had the highest point accuracy, shown by the lowest MSE.
- The MCMC parameter-wise sampler and Laplace approximation showed the best classification performance with the lowest misclassification rates.
- The Laplace and Variational approximation had the least computational expense, with the lowest run times.
- The MCMC block sampler showed a much better performance in terms of convergence, autocorrelation and computational costs than parameter-wise sampler when we observed their respective diagnostics, especially in the Diabetes dataset where there existed correlation between the parameters.
- Despite the point estimates being more accurate for the Laplace and Variational approximation in the simulation study, the overall posterior density distribution in the MCMC block sampler showed the best performance with proper uncertainty quantification and realistic posterior tails.

Chapter 4

Conclusion

All the stated Bayesian methods can be effectively implemented to approximate the posterior for logistic regression, as seen in the simulation study and Pima Indians Diabetes datasets. We also observed that the Newton-Raphson algorithm was a very effective method for calculating the MLE with good accuracy and little computational expense. However, it only gives us a point-estimate and fails to account for uncertainty in the estimates or include prior information, unlike our Bayesian approaches.

MCMC was implemented using both parameter-wise and block updates. The block sampler significantly outperformed the parameter-wise approach in terms of computational efficiency and mixing. The parameter-wise sampler suffered from high autocorrelation and required substantially more runtime. Surprisingly, across both the simulation and the diabetes dataset, MCMC showed slightly lower point-estimate accuracy than expected, although it provided a more realistic posterior distribution with better uncertainty quantification. We would expect these estimates to improve as we run the chain for longer, especially for the block sampler as it showed good diagnostic convergence.

The Laplace and Variational approximations demonstrated the best performance in terms of accuracy and computational speed. However, they both underestimated uncertainty due to their Gaussian assumptions and deterministic nature.

Therefore, in conclusion, each of the Monte Carlo Markov Chain block sampler, Laplace approximation and Variational approximation methods are effective Bayesian inferential approaches in approximating the posterior of the logistic regression model, each with their own respective strengths and limitations as discussed. The so-called 'best' method ultimately depends on the data and the statistician's priorities; whether the focus is on accuracy, computational efficiency, or the quality of uncertainty quantification.

Further Work

While this project provides a comprehensive report on different Bayesian approaches for logistic regression, there are still several ways in which I would love to extend it if I were to pursue further research on this topic:

1. Sample Size: Examine how different sample sizes n affect accuracy and computational costs.
2. Class Imbalance: Investigate how the ratio of success to failures in the response variable ($y = 1/y = 0$) affects a models stability.
3. Variable Selection: Perform variable selection within in methods and explore its impact.

4. Parameters and Distributions: Explore how different priors, hyperparameters and MCMC proposal distributions influence convergence and posterior accuracy.

Chapter 5

Appendix

This appendix includes the key blocks of code used to apply these methods in R and provides with the results shown. The complete code files are included in the accompanying files submitted with the report.

5.1 Newton-Raphson Function Code

```r
n <- nrow(X)
p <- ncol(X)
beta <- rep(0, p)
beta_history <- matrix(NA, nrow = 10000, ncol = p)

logistic <-
  function(z) {
  exp(z) / (1 + exp(z))
}

start_time <- Sys.time()
for (iter in 1:10000) {
  pi_hat <- logistic(X %*% beta)

  u <- t(X) %*% (y - pi_hat)

  W <- diag(as.vector(pi_hat * (1 - pi_hat)), n, n)
  H <- -t(X) %*% W %*% X

  beta_new <- beta - solve(H) %*% u
  beta <- beta_new

  beta_history[iter, ] <- as.vector(beta_new)
  if (max(abs(u)) < 1e-6) {
  beta_history <- beta_history[1:iter, , drop = FALSE]
  break}
}
end_time <- Sys.time()

NR_time <- end_time - start_time
beta_NR <- beta
var_beta <- solve(-H)
```

Listing 5.1: Newton-Raphson Function Code

5.2 Bayesian Posterior Code

```r
log_likelihood <- function(beta, y, X) {
  XB <- X %*% beta
  log_p <- sum(y * XB - log(1 + exp(XB)))
  return(log_p)
}

log_prior <- function(beta, mu = 0, sigma = 10) { # N(0,100)
  return(-sum((beta - mu)^2 / (2 * sigma^2)))
}

log_posterior <- function(beta, y, X) {
  return(log_likelihood(beta, y, X) + log_prior(beta))
}
```

Listing 5.2: Bayesian Posterior Code

5.3 MCMC Parameter-Wise Code

```r
MHGibbs_logistic <- function(X, y, init, iter, burnin, thin, Sigma_prop) {

  p <- ncol(X)
  beta <- matrix(NA, nrow = iter, ncol = p)
  beta[1, ] <- init

  accept_counts <- rep(0, p)

  for (i in 2:iter) {
    beta_current <- beta[i - 1, ]

    for (j in 1:p) {
      beta_star <- beta_current
      beta_star[j] <- rnorm(1, mean = beta_current[j], sd = sqrt(Sigma_prop[j, j]))

      log_R <- log_posterior(beta_star, y, X) - log_posterior(beta_current, y, X)

      if (log(runif(1)) < log_R) {
        beta_current[j] <- beta_star[j]
        accept_counts[j] <- accept_counts[j] + 1
      }
    }

    beta[i, ] <- beta_current
  }

  overall_acceptance_rate <- 100 * sum(accept_counts) / (iter * p)
  cat("Overall acceptance rate was ", overall_acceptance_rate, "%\n", sep = "")

  per_param_acceptance <- 100 * accept_counts / iter
  cat("Acceptance rate per parameter (in %):\n")
  print(per_param_acceptance)

  beta_thinned <- beta[(burnin + 1):iter,]
  beta_thinned <- beta_thinned[seq(1, nrow(beta_thinned), by = thin),]

  return(beta_thinned)
```

```
}

beta_init <- c(0,0,0,0)
cov_mle <- var_beta

start_time <- Sys.time()
mcmc_gibbs <- MHGibbs_logistic(X, y, beta_init, iter = 10000, burnin = 1000, thin = 10,
    Sigma_prop = cov_mle)
end_time <- Sys.time()
```

Listing 5.3: MCMC Parameter-Wise Code

5.4 MCMC Block Code

```
MH_logistic <- function(X, y, init, iter, burnin, thin, Sigma_prop) {

  p <- ncol(X)
  beta <- matrix(NA, nrow = iter, ncol = p)
  beta[1, ] <- init #initial values
  accept_sum <- 0 #no. of accepted proposals

  for (i in 2:iter) {
    beta_star <- mvrnorm(1, mu = beta[i - 1, ], Sigma = Sigma_prop)


    log_R <- log_posterior(beta_star, y, X) - log_posterior(beta[i - 1, ], y, X)

    if (log(runif(1)) < log_R) {
       beta[i, ] <- beta_star
      accept_sum <- accept_sum + 1
    } else {
      beta[i, ] <- beta[i - 1, ]
    }
  }

  cat("Acceptance rate was ", 100 * accept_sum / iter, "%\n", sep = "")

  beta_thinned <- beta[(burnin + 1):iter, ]
  beta_thinned <- beta_thinned[seq(1, nrow(beta_thinned), by = thin), ]

  return(beta_thinned)
}
beta_init <- c(0,0,0,0)
cov_mle <- var_beta

start_time <- Sys.time()
mcmc <- MH_logistic(X, y, beta_init, iter = 10000, burnin = 1000, thin = 10, Sigma_prop =
     cov_mle)
end_time <- Sys.time()
```

Listing 5.4: MCMC Block Code

5.5 Laplace Approximation Code

```r
mu<- rep(0, p)
sigma2 <- rep(100, p)
sigma2_inv <- diag(1 / sigma2, nrow = p, ncol = p)
beta <- rep(0, p)

start_time <- Sys.time()
for (iter in 1:10000) {
  pi_hat <- logistic(X %*% beta)

  u <- (t(X) %*% (y - pi_hat)) - (beta - mu)/sigma2

  W <- diag(as.vector(pi_hat * (1 - pi_hat)), n, n)
  H_like <- - t(X) %*% W %*% X

  H <- H_like - sigma2_inv

  beta_new <- beta - solve(H) %*% u
  beta <- beta_new

  if (max(abs(u)) < 1e-6) break
}
end_time <- Sys.time()

beta_MAP <- beta
logpost_betaMAP <- log_posterior(beta_MAP, y, X)

log_laplace <- function(beta) {
  logpost_betaMAP + 0.5 * t(beta - beta_map) %*% H %*% (beta - beta_map)
}

laplace <- function(beta) {
  exp(log_laplace(beta))
}
Sigma_laplace <- solve(-H)
```

Listing 5.5: Laplace Approximation Code

5.6 Variational Approximation Code

```r
N <- nrow(X)
p <- ncol(X)
mu0 <- rep(0, p)
Sigma0 <- diag(100, p) #N(0,100)
Sigma0_inv <- solve(Sigma0)

variational_approximation <- function(X, y, mu0, Sigma0, maxiter, tol) {
  mu  <- mu0
  Sig <- Sigma0
  eta  <- rep(1, N)

  for (iter in seq_len(maxiter)) {

    lambda <- (logistic(eta) - 0.5) / (2 * eta)

    Sig_inv <- Sigma0_inv + 2 * t(X) %*% (lambda * X)

```

```
    Sig_new <- solve(Sig_inv)
    mu_new  <- Sig_new %*% (Sigma0_inv %*% mu0 + t(X) %*% (y - 0.5))

    eta_square    <- rowSums((X %*% (Sig_new + tcrossprod(mu_new))) * X)
    eta_new  <- sqrt(eta_square)

    if (max(abs(mu_new - mu)) < tol) {
      mu  <- mu_new;
      Sig <- Sig_new;
      eta <- eta_new
      break
    }
    mu  <- mu_new
    Sig <- Sig_new
    eta  <- eta_new
  }

  list(mu = drop(mu), Sigma = Sig, eta = eta, iter = iter)
}

start_time <- Sys.time()
var_approx <- variational_approximation(X, y, mu0 = mu0, Sigma0 = Sigma0, maxiter = 500,
    tol = 1e-6)
end_time <- Sys.time()
```

Listing 5.6: Variational Approximation Code

References

1. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
2. Agresti, A. (2007). An Introduction to Categorical Data Analysis (2nd ed.). Wiley.
3. Hoff, P. D. (2009). A First Course in Bayesian Statistical Methods. Springer.
4. Robert, C. P., Casella, G. (2010). Introducing Monte Carlo Methods with R. Springer.
5. Jaakkola, T. S., Jordan, M. I. (2000). Bayesian parameter estimation via variational methods. Statistics and Computing, 10(1), 25–37.
6. Tierney, L., Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. Journal of the American Statistical Association, 81(393), 82–86.
7. Chib, S., Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. The American Statistician, 49(4), 327–335.