Bayesian modelling of latent Gaussian models featuring variable selection

Keith Newman

Thesis submitted for the degree of
Doctor of Philosophy

School of Mathematics & Statistics
Newcastle University
Newcastle upon Tyne
United Kingdom

January 2017

"Hofstadter's Law: It always takes longer than you expect, even when you take into account Hofstadter's Law"

— Douglas R. Hofstadter.

Acknowledgements

Many thanks to my supervisor, Professor Darren Wilkinson, whose unfaltering patience and support leaves me astounded—a perfect example of PhD supervision. I must also give thanks to the Engineering and Physical Sciences Research Council for the generous funding that made all this possible. Thanks to David Lydall, Conor Lawless, and the rest of the *Lydall Lab*, for allowing me to be involved with their QFA-related experiments and access data which underpins Part II of this thesis. And thanks to Andrew Watson, a great friend for many years who gave me a crash-course on the MRX complex mechanisms mentioned in Chapter 6.

Thanks to all the School of Maths & Stats staff for their assistance, to Dr. Lee Fawcett for providing enjoyable teaching and programming opportunities, and Lizzie Vinnicombe, an unsung hero to a few students at Newcastle University.

My thanks to those from MathSoc across the years for helping me maintain useful links with the undergraduate community. Additional thanks to Emma Bradley and Amy Green, who gave much needed encouragement and company in the final months.

Thanks to all on Herschel level 4, both past and present. In particular: Holly Ainsworth and Nina Wilkinson, who guided me through the first few tough months; Jamie Owen, an incredibly loyal friend who always offers help; George Stagg, for indispensable computer knowledge; Sarah Jowett, David Robertson and Joe Matthews, for encouraging me to be more social; Clarissa Barratt for entertaining me with the *best conversations ever* while I performed thesis corrections.

I wish to thank my friends—Rhydian, Miriam, Simon, Seb, Sean and Jonathan—for making life a pleasure when work was a pain. Thanks to my family for their patience while I worked on this instead of spending quality time at home.

It could surprise and puzzle many people when I reveal that I'm saving my final and greatest thanks for a fellow PhD student who I've rarely ever seen in-person. But this special credit goes to my best friend and most dependable ally, who, perhaps unknowingly, guided me out of my toughest moments over the last four years; it's very possible I would not have created this thesis without the encouragement, understanding and well-timed interventions from the wisest person I know. Thank you, Colleen Nooney.

Thank you, all.

Abstract

Latent Gaussian models are popular and versatile models for performing Bayesian inference. In many cases, these models will be analytically intractable creating a need for alternative inference methods. Integrated nested Laplace approximations (INLA) provides fast, deterministic inference of approximate posterior densities by exploiting sparsity in the latent structure of the model. Markov chain Monte Carlo (MCMC) is often used for Bayesian inference by sampling from a target posterior distribution. This suffers poor mixing when many variables are correlated, but careful reparameterisation or use of blocking methods can mitigate these issues. Blocking comes with additional computational overheads due to the matrix algebra involved; these costs can be limited by harnessing the same latent Markov structures and sparse precision matrix properties utilised by INLA, with particular attention paid to efficient matrix operations.

We discuss how linear and latent Gaussian models can be constructed by combining methods for linear Gaussian models with Gaussian approximations. We then apply these ideas to a case study in detecting genetic epistasis between telomere defects and deletion of non-essential genes in *Saccharomyces cerevisiae*, for an experiment known as Quantitative Fitness Analysis (QFA). Bayesian variable selection is included to identify which gene deletions cause a genetic interaction. Previous Bayesian models have proven successful in detecting interactions but time-consuming due to the complexity of the model and poor mixing. Linear and latent Gaussian models are created to pursue more efficient inference over standard Gibbs samplers, but we find inference methods for latent Gaussian models can struggle with increasing dimension. We also investigate how the introduction of variable selection provides opportunities to reduce the dimension of the latent model structure for potentially faster inference.

Finally, we discuss progress on a new follow-on experiment, Mini QFA, which attempts to find epistasis between telomere defects and a pair of gene deletions.

Contents

List of Figures

List of Tables

Part I

Preliminaries and methodologies

Chapter 1

Introduction

The desire to perform Bayesian inference on increasingly complicated models has required investigations into methods that can perform inference on high-dimension models. Markov chain Monte Carlo methods became prevalent as a means to perform Bayesian inference, but as model dimension increases, especially in the presence of highly correlated variables, so does the time needed to perform a good analysis. Correlated variables being inferred by these stochastic means can often have slow convergence to the target posterior distribution.

This thesis explores methods to make inference on latent Gaussian models more *efficient*, which we will define as the ability to produce a suitably uncorrelated MCMC chain in minimal CPU user time. Our main avenues of exploration for achieving this will be the use of blocking MCMC methods. As we will consider models that contain elements of Bayesian variable selection, we will also explore computational tricks that utilise the variable selection to achieve reductions in scheme completion time.

Many investigations have been made to make various MCMC schemes run in parallel (Neiswanger et al., 2013; Casarin et al., 2015), including work to perform this on spatial latent Gaussian models (Whiley and Wilson, 2004), which will in practice make schemes run faster for users who now have easy access to multi-core processors and clusters. For the purposes of this thesis, we will not consider the use of parallel chains as we have an interest in ensuring each chain is as efficient as possible. Therefore, all measurements of computation time will be done based on CPU user time.

Not all Bayesian inference methods employ stochastic techniques. Rue et al. (2009) demonstrate that they are able to use a deterministic scheme to perform

approximate inference on latent Gaussian models. In many cases, the inference can be faster than an equivalent MCMC scheme, with evidence that the approximations are very close to inference performed by exact[1] MCMC methods. We therefore investigate the methods used in INLA as part of this thesis.

To test the methods that are explored, applications of the techniques will be made to some genetics datasets in Part II of this thesis.

1.1 Bayesian variable selection

As explained by O'Hara and Sillanpää (2009), Bayesian variable selection focuses on the problem of finding the posterior probability that each variable should be included in the model. It considers posterior probabilities for all possible considered models, as opposed to the frequentist equivalent of selecting a single optimum model.

In this thesis, we will feature two variable selection schemes which are used to help calculate the posterior probability that an associated effect should be included in the model. In both cases, these are treated as simple binary indicators, δ, that can take the values $\delta = 0$ or $\delta = 1$. If n of the variables, $x_1, \ldots, x_n$, are to be tested for inclusion in the model, then each variable is multiplied by a corresponding indicator, δ_l, $l = 1, \ldots, n$, so that the combined effect is $\delta_l x_l$. When an indicator states that a variable should be included in the model, then $\delta = 1$, causing $\delta_l x_l = x_l$. On the other hand, if the indicator states that a variable should be excluded, then $\delta = 0$, causing $\delta_l x_l = 0$; this essentially means x_l has no effect on the model.

The first variable selection scheme will be referred to as the Kuo and Mallick (1998) indicators. The indicator and the variable it acts upon are assumed to be independent, such that $\pi(\delta_l, x_l) = \pi(\delta_l)\pi(x_l)$. O'Hara and Sillanpää (2009) warns that if the prior distribution on x_l is too vague, the indicator will rarely flip from $\delta = 0$ to $\delta = 1$, since when x_l is sampled from a vague prior distribution when excluded from the model, it will often sample values with low posterior support.

The second variable selection method featured is Gibbs variable selection (GVS) (Dellaportas et al., 2002). This behaves like the Kuo & Mallick indicators, except the distribution of the variable depends on the value of the indicator. Now $\pi(\delta_l, x_l) = \pi(\delta_l)\pi(x_l \mid \delta_l)$ where $\pi(x_l \mid \delta_l) = (1 - \delta_l)\mathrm{N}(\tilde{\mu}_l, \tilde{\sigma}_l^2) + \delta_l \mathrm{N}(0, \sigma^2)$.

[1] An "exact" MCMC scheme is intended to define a method that samples from the exact target posterior distribution, but the stochastic nature of MCMC introduces random error itself.

Consequently, when the variable x_l is currently excluded from the model, it is sampled from a pseudo-prior, which has parameters $\tilde{\mu}_l$ and $\tilde{\sigma}_l$, designed to closely match the target posterior distribution. This makes the indicator more likely to swap from $\delta = 0$ to $\delta = 1$ for better mixing. When x_l is currently included, the pseudo-prior distribution has no effect, and x_l would follow the actual posterior distribution.

O'Hara and Sillanpää (2009) provides a good overview of other methods of variable selection exist, such as reversible-jump MCMC (Green, 1995) where the variables are randomly selected and proposed for inclusion or removal from the current model.

1.2 Novel contributions to the scientific community

Contributions have been made to statistical computing by rewriting existing software, known as `GDAGsim`, into a Java version, called `GDAGsimJ`, before enhancing the feature set of `GDAGsimJ` to work efficiently on larger datasets.

`GDAGsim` and `GDAGsimJ` can be used to perform inference on linear Gaussian models. We perform an investigation into its efficiency in comparison to popular existing software by testing their capabilities on a large-dimension linear Gaussian model that also includes an additional complexity from Bayesian variable selection. This allows us to establish which software is more efficient and to quantify how much more efficient the optimal software is. `GDAGsimJ` was then combined with deterministic approximation methods to explore its capabilities at performing inference on the more general class of latent Gaussian models, with the aim of establishing when this method will be effective and efficient. Throughout these investigations, we also check what strategies for performing Bayesian variable selection will provide more efficient inference, providing an overlapping contribution between statistical methodologies and computing.

Finally, research is performed on a new and ongoing experiment featured in Chapter 6. This requires a new statistical model to work at the end of an existing analytical pipeline, providing a contribution to the genetics community by identifying possible relationships between the functions of various genes.

1.3 Outline of thesis

Part I of this thesis presents the basic concepts for inference on latent Gaussian models, before the ideas are used on genetics-related case studies in Part II.

Both linear and latent Gaussian models are detailed in Chapter 2, where we demonstrate how models can be constructed by specifying each latent variable in turn based on its conditional dependencies. We also highlight how to condition on the model, with different techniques used between latent and linear Gaussian models. Sparsity among latent structures can also be found and exploited to reduce memory usage and computational demands in the sparse matrix operations also detailed towards the end of Chapter 2.

Integrated nested Laplace approximations are investigated in Chapter 3, where we describe the method as detailed by Rue et al. (2009) and we attempt to recreate their `INLA` software. We compare the accuracy of the software to both our own version of the software and long, high-quality MCMC chains from `JAGS`.

Chapter 4 provides an overview of various Markov chain Monte Carlo schemes that can be employed for performing Bayesian inference. These range from standard Gibbs and Metropolis-Hastings algorithms, to blocking methods designed to help improve mixing. The blocking methods are extended further to accommodate Bayesian variable selection, adding in the necessary steps to perform inference on a set of variable selection indicators.

To commence Part II of the thesis, Chapter 5 looks at a case study for an experiment to identify epistasis in telomere-defective chromosomes that have non-essential gene deletions. This experiment is named Quantitative Fitness Analysis (QFA). We provide a brief background in genetics before presenting the calculations that underpin the experiment. We then perform a comparison between blocking methods and conventional Gibbs sampling techniques for both linear and latent Gaussian versions of the model.

A new, ongoing follow-up study to QFA is featured in Chapter 6. A relatively small selection of genes are chosen, and the pairwise deletion of each of these genes is performed against different telomere defects. Despite using a small subset of possible gene deletions, the possible number of combinations creates a model of even greater dimension than that seen in the QFA experiment.

Conclusions are made in Chapter 7 for the content of this thesis, while also discussing future topics to be investigated for this work.

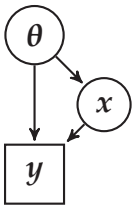
Chapter 2

Background

2.1 Linear Gaussian models

2.1.1 Model structure

Linear Gaussian models are highly versatile models with many applications. The structure of a typical linear Gaussian model can be represented by the Directed Acyclic Graph (DAG) shown in Figure 2.1. In the DAG, vertices are used to represent variables of the model, while edges are used to show conditional dependence between any of these variables. We see that the parameters of the model, $\boldsymbol{\theta}$, can have influence on the observations of the model, $\boldsymbol{y}$, and the latent variables, $\boldsymbol{x}$. In turn, the latent variables, which are not observed directly, can also influence the values of the observations. In the DAG, square nodes represent an observed value, while round nodes show an unobserved variable.



$\boldsymbol{\theta}$ are parameters of the model.

$\boldsymbol{x}$ are the latent Gaussian variables.

$\boldsymbol{y}$ are the observed variables.

Figure 2.1: A general Directed Acyclic Graph for a linear Gaussian system

The DAG shown in Figure 2.1 would correspond to the factorisation,

$$\pi(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}) = \pi(\boldsymbol{\theta})\ \pi(\boldsymbol{x} \mid \boldsymbol{\theta})\ \pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}). \tag{2.1}$$

For a linear Gaussian model we require that, conditional on the parameters, $\boldsymbol{\theta}$, all other variables in the model, $(\boldsymbol{x}, \boldsymbol{y})$, are jointly multivariate Normal in their distribution. For a p-dimensional model, the observations can have a mean equal to a linear combination of the latent variables:

$$\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta} \sim \mathrm{N}_p(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}(\boldsymbol{\theta})), \tag{2.2}$$

where $\boldsymbol{y}$, $\boldsymbol{x}$ and $\boldsymbol{b}$ are $1 \times p$ vectors; $\boldsymbol{A}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}(\boldsymbol{\theta})$ are $p \times p$ matrices. The latent values follow a distribution,

$$\boldsymbol{x} \mid \boldsymbol{\theta} \sim \mathrm{N}_p\big(\boldsymbol{\mu}, [\mathbf{Q}(\boldsymbol{\theta})]^{-1}\big),$$

where $\boldsymbol{\mu}$ is a $1 \times p$ vector, $\mathbf{Q}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}_{\boldsymbol{x}}(\boldsymbol{\theta})^{-1}$ is a $p \times p$ precision matrix quantifying the conditional dependence between latent variables, and $\boldsymbol{\Sigma}_{\boldsymbol{x}}(\boldsymbol{\theta})$ is the *variance* matrix (also known as a *covariance* matrix) of the latent variables.

2.1.2 Sparsity in latent structures

By adding more detail in to Figure 2.1, we can better understand the structures and dependencies between latent variables and observations. One such example could be the model structure represented by the DAG in Figure 2.2, where this sort of structure might be seen in an autoregressive walk of order 1, AR(1), or random walk of order 1, RW(1).

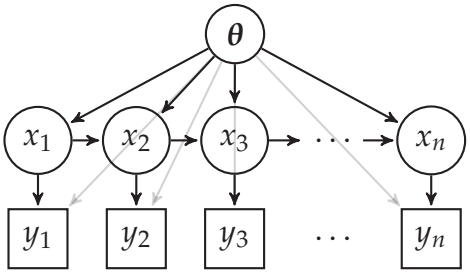


Figure 2.2: A more detailed example DAG for a particular linear Gaussian system. This particular DAG could be found in an AR(1) system as an example. Note that the edges denoting dependence between the parameters and each observation are faded only for legibility reasons.

The structure of the example in Figure 2.2 shows how the value of a latent variable can be dependent on a preceding latent variable and the parameters of the model, while also being a parent of, and therefore being able to influence, the value

of an observation. For example, latent variable x_2 of Figure 2.2 is dependent on x_1 and parameters $\boldsymbol{\theta}$. Each observation, $y_i, i = 1, \ldots, n$, in the Figure 2.2 example also shows dependence on the corresponding latent variable, x_i, and the parameters, $\boldsymbol{\theta}$.

Where a DAG shows no edge connecting any two of the vertices of the graph, we can see that those two variables are conditionally independent. For the example in Figure 2.2, we could say that the observations, $\boldsymbol{y}$, are conditionally independent given the latent variables, $\boldsymbol{x}$, and parameters, $\boldsymbol{\theta}$, i.e.

$$y_i \perp\!\!\!\perp y_j \mid (\boldsymbol{x}, \boldsymbol{\theta}) \; \forall \, i \neq j.$$

We also see a Markov property exhibited among the variables of the latent structure, such as $x_3 \perp\!\!\!\perp x_1 \mid (x_2, \boldsymbol{\theta})$—again, there is no edge that directly connects x_3 and x_1 in the DAG of Figure 2.2.

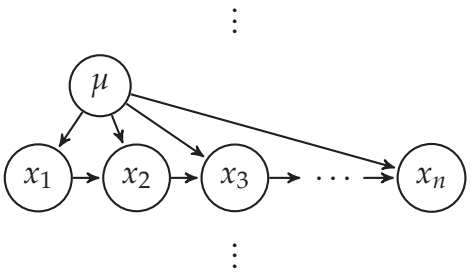
While a variance matrix shows the covariance between variables, a *precision* matrix shows the conditional dependence between variables. Consequently, models showing conditional independence due to any Markov properties will have correspondingly located zero-entries in its precision matrix. Using Figure 2.2 as an example, the precision matrix between latent variables $\boldsymbol{x}$ would be,

$$\mathbf{Q}_{\boldsymbol{x}} = \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & \ldots & x_n \end{array} \\ \begin{pmatrix} \tau_{1,1}(\boldsymbol{\theta}) & \tau_{1,2}(\boldsymbol{\theta}) & 0 & \ldots & 0 \\ \tau_{2,1}(\boldsymbol{\theta}) & \tau_{2,2}(\boldsymbol{\theta}) & \tau_{2,3}(\boldsymbol{\theta}) & \ddots & \vdots \\ 0 & \tau_{3,2}(\boldsymbol{\theta}) & \tau_{3,3}(\boldsymbol{\theta}) & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \tau_{n-1,n}(\boldsymbol{\theta}) \\ 0 & \ldots & 0 & \tau_{n,n-1}(\boldsymbol{\theta}) & \tau_{n,n}(\boldsymbol{\theta}) \end{pmatrix} \end{array}, \tag{2.3}$$

which shows a tri-diagonal structure, where $\tau_{i,j}$ denotes the conditional dependence between variables i and j.

We may often encounter the situation where multiple elements of the latent structure depend on the value of a single other latent variable; this commonly occurs when there is an underlying mean to a latent process. Suppose latent variable, μ, is an underlying mean value, and $x_i \sim \mathrm{N}(\mu, \sigma^2)$ for $i = 1, \ldots, n$, as shown in Figure 2.3a, then the precision matrix would feature a diagonal structure with a dense first row and column, and zero values in all remaining entries as

given in Figure 2.3b.



(a) DAG for latent variables in a linear Gaussian model containing latent variables, $x_1, \ldots, x_n$, that are all conditional on a common mean latent variable, μ. Parameters and observations are omitted from this diagram.

$$\mathbf{Q} = \begin{pmatrix} \times & \times & \times & \times & \times & \cdots & \times & \times & \times \\ \times & \times & \times & & & & & & \\ \times & \times & \times & \times & & & & & \\ \times & & \times & \times & \times & & & & \\ \times & & & \times & \times & \times & & & \\ \vdots & & & & \ddots & \ddots & \ddots & & \\ \times & & & & & \times & \times & \times & \\ \times & & & & & & \times & \times & \times \\ \times & & & & & & & \times & \times \end{pmatrix}$$

(b) Precision matrix where $\times$ denotes a non-zero entry. The first row/column represents the μ parameter, with all remaining parameters representing $\boldsymbol{x}$.

Figure 2.3: Example DAG and precision matrix for a latent structure with a common mean parameter, μ.

2.1.3 The benefit of sparsity

The quantity of latent variables and parameters in a model can often be quite large, leading to high dimensional matrices that represent the model. Working with such large dimension matrices can be expensive in terms of both computation cost and memory requirements.

For dense $n \times n$ matrices, the computational cost of a matrix-matrix multiplication or matrix inversion using a naive algorithm is $\mathcal{O}(n^3)$. There are more efficient algorithms, such as the Coppersmith and Winograd (1990) algorithm which can reduce this to $\mathcal{O}(n^{2.376})$, but this can still be time consuming as the dimension of the matrix increases.

We can exploit the sparsity in the precision matrices using special storage and calculation algorithms for sparse matrices. For sparse $n \times n$ matrices with no more than m non-zero elements, a multiplication algorithm by Yuster and Zwick (2005) performs in as little as $\mathcal{O}(m^{0.7}n^{1.2} + n^{2+\mathcal{O}(1)})$ operations and never more than the number of operations taken by the alternative dense operations. To save memory, the non-zero values of a sparse matrix can be stored with their corresponding indices in a map—this saves memory when the majority of the matrix values are

zero, with potential reductions to computational time and memory requirements as a direct consequence.

Implementing these methods for sparse matrices has been made convenient thanks to a number of software packages that perform these sparse operations for a variety of commonly used programming languages. Some examples include: `Matrix` for R (Bates and Mächler, 2014); `CSPARSE` for C (Davis, 2006); `PSPASES` (Joshi et al., 1999); `Parallel Colt` for Java (Wendykier and Nagy, 2010).

2.1.4 Canonical parameterisation of the Gaussian distribution

We begin with the *moment* parameterisation of a d-dimensional multivariate Gaussian distribution, specified by a mean vector, $\boldsymbol{\mu}$ and variance matrix, $\boldsymbol{\Sigma}$:

$$\begin{aligned} \boldsymbol{X} &\sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \pi(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \frac{1}{\sqrt{2\pi}^d |\boldsymbol{\Sigma}|^{1/2}} \exp\left\{ -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right\}. \end{aligned} \tag{2.4}$$

It will often be more convenient to consider this distribution in terms of its precision matrix, which will often be sparse for the models considered in this thesis. As previously mentioned in Section 2.1.1, we define the precision matrix as $\mathbf{Q} = \boldsymbol{\Sigma}^{-1}$:

$$\begin{aligned} \boldsymbol{X} &\sim \mathrm{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1}) \\ \pi(\boldsymbol{x}; \boldsymbol{\mu}, \mathbf{Q}^{-1}) &= \frac{|\mathbf{Q}|^{1/2}}{\sqrt{2\pi}^d} \exp\left\{ -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \mathbf{Q} (\boldsymbol{x} - \boldsymbol{\mu}) \right\}. \end{aligned} \tag{2.5}$$

A *canonical* parameterisation of the Gaussian distribution can be obtained by performing the substitutions, $\boldsymbol{h} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ and $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ into Equation (2.4) (Lauritzen, 1992; Wilkinson and Yeung, 2002). This gives a density of,

$$\begin{aligned} \boldsymbol{X} &\sim \mathcal{N}(\boldsymbol{h}, \boldsymbol{\Lambda}) \equiv \mathrm{N}(\boldsymbol{\Lambda}^{-1}\boldsymbol{h}, \boldsymbol{\Lambda}^{-1}) \\ \pi(\boldsymbol{x}; \boldsymbol{\Lambda}^{-1}\boldsymbol{h}, \boldsymbol{\Lambda}^{-1}) &\propto \exp\left\{ -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\Lambda}^{-1}\boldsymbol{h})^T \boldsymbol{\Lambda} (\boldsymbol{x} - \boldsymbol{\Lambda}^{-1}\boldsymbol{h}) \right\} \\ &\propto \exp\left\{ -\frac{1}{2} \boldsymbol{x}^T \boldsymbol{\Lambda} \boldsymbol{x} + \boldsymbol{h}^T \boldsymbol{x} \right\}. \end{aligned} \tag{2.6}$$

$\boldsymbol{h}$ and $\boldsymbol{\Lambda}$ are then called the canonical parameters, and $\boldsymbol{\Lambda}$ is equivalent to the

precision matrix $\mathbf{Q}$.

Taking the product of two canonically parameterised multivariate Gaussian densities gives,

$$\begin{aligned}
&\mathcal{N}(\boldsymbol{x}; \boldsymbol{h}_1, \boldsymbol{\Lambda}_1)\, \mathcal{N}(\boldsymbol{x}; \boldsymbol{h}_2, \boldsymbol{\Lambda}_2) \\
&\quad \propto \exp\left\{-\frac{1}{2}\boldsymbol{x}^T \boldsymbol{\Lambda}_1 \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{h}_1\right\} \exp\left\{-\frac{1}{2}\boldsymbol{x}^T \boldsymbol{\Lambda}_2 \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{h}_2\right\} \\
&\quad \propto \exp\left\{-\frac{1}{2}\boldsymbol{x}^T (\boldsymbol{\Lambda}_1 + \boldsymbol{\Lambda}_2) \boldsymbol{x} + \boldsymbol{x}^T (\boldsymbol{h}_1 + \boldsymbol{h}_2)\right\} \\
&\quad \equiv \mathcal{N}(\boldsymbol{x}; \boldsymbol{h}_1 + \boldsymbol{h}_2, \boldsymbol{\Lambda}_1 + \boldsymbol{\Lambda}_2),
\end{aligned} \tag{2.7}$$

up to a normalising constant. This implies the product of two Gaussian distributions is equivalent to summing the canonical parameters.

2.1.5 Prior model construction from DAG specification

The result from Equation (2.7) leads to a simplification of multivariate Normal theory which produces a method that allows the canonical parameter matrices to be easily constructed from the model (Wilkinson and Yeung, 2004), such that the model can be described by $\mathcal{N}(\boldsymbol{h}, \boldsymbol{\Lambda})$. Models can be constructed according to their DAG specification in a node-by-node fashion.

Roots in the latent structure are simplest to add to the model as they do not need to be conditioned on any parent nodes, excluding parameter nodes. For a root, $x_i \mid \boldsymbol{\theta} \sim \mathrm{N}\left(\mu_i, \tau_i^{-1}\right) \equiv \mathcal{N}(\tau_i \mu_i, \tau_i)$, $\{\mu_i, \tau_i\} \in \boldsymbol{\theta}$, the canonical parameter matrices can be updated as,

$$b_i = \tau_i \mu_i \quad \text{and} \quad \Lambda_{ii} = \tau_i.$$

Nodes which depend on other nodes or roots can then be specified. When we add the j-th node, we aim to find the values of the canonical parameters with j nodes specified, denoted $\boldsymbol{h}_{(j)}$ and $\boldsymbol{\Lambda}_{(j)}$. If we define the notation $\boldsymbol{x}_{<j} = x_1, \ldots, x_{j-1}$, then we wish to find the distribution of $\mathcal{N}\left(\boldsymbol{x}_{<j+1}; \boldsymbol{h}_{(j)}, \boldsymbol{\Lambda}_{(j)}\right)$, which has a density that can be factorised as,

$$\prod_{i=1}^{j} \pi(X_i \mid \mathrm{Par}(X_i), \boldsymbol{\theta}) = \pi(X_j \mid \mathrm{Par}(X_j), \boldsymbol{\theta}) \prod_{i=1}^{j-1} \pi(X_i \mid \mathrm{Par}(X_i), \boldsymbol{\theta}), \tag{2.8}$$

where $\mathrm{Par}(X_i)$ is an $(i-1)$-dimension vector defining the parents of node X_i, which will typically be sparse due to the sparse properties of the latent structure.

The density of $\pi(X_j \mid \mathrm{Par}(X_j), \boldsymbol{\theta})$ can be considered Gaussian where the mean parameter is a linear combination of dependencies ($\boldsymbol{a}_j^T \boldsymbol{X}_{<j} + b_j$ for constant b_j and a $(j-1)$-dimension sparse vector $\boldsymbol{a}_j^T$ denoting which nodes are parents of X_j) and with precision, τ:

$$\begin{aligned} X_j \mid \mathrm{Par}(X_j), \boldsymbol{\theta} &\sim \mathrm{N}(x_j; \boldsymbol{a}_j^T \boldsymbol{x}_{<j} + b_j, \tau_j) \\ \pi(X_j \mid \mathrm{Par}(X_j), \boldsymbol{\theta}) &\propto \exp\left\{ -\frac{1}{2}(x_j - \boldsymbol{a}_j^T \boldsymbol{x}_{<j} - b_j)\tau_j(x_j - \boldsymbol{a}_j^T \boldsymbol{x}_{<j} - b_j) \right\} \\ &\propto \exp\left\{ -\frac{1}{2}\left[\begin{pmatrix} \boldsymbol{x}_{<j}^T & x_j \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_j \tau_j \boldsymbol{a}_j^T & -\boldsymbol{a}_j \tau_j \\ -\tau_j \boldsymbol{a}_j^T & \tau_j \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_{<j} \\ x_j \end{pmatrix} \right.\right. \\ &\qquad \left.\left. - 2 \begin{pmatrix} \boldsymbol{x}_{<j}^T & x_j \end{pmatrix} \begin{pmatrix} -\boldsymbol{a}_j \tau_j b_j \\ \tau_j b_j \end{pmatrix} \right] \right\}, \end{aligned}$$

which matches the form of the canonically parameterised Gaussian density from Equation (2.6). Therefore:

$$X_j \mid \mathrm{Par}(X_j), \boldsymbol{\theta} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{x}_{<j} \\ x_j \end{pmatrix}; \begin{pmatrix} -\boldsymbol{a}_j \tau_j b_j \\ \tau_j b_j \end{pmatrix}, \begin{pmatrix} \boldsymbol{a}_j \tau_j \boldsymbol{a}_j^T & -\boldsymbol{a}_j \tau_j \\ -\tau_j \boldsymbol{a}_j^T & \tau_j \end{pmatrix} \right). \tag{2.9}$$

We can use Equation (2.9) to complete the calculation of Equation (2.8),

$$\begin{aligned} &\mathcal{N}\left(\boldsymbol{x}_{<j+1}; \boldsymbol{h}_{(j)}, \boldsymbol{\Lambda}_{(j)}\right) \\ &\quad \equiv \pi(X_j \mid \mathrm{Par}(X_j), \boldsymbol{\theta}) \prod_{i=1}^{j-1} \pi(X_i \mid \mathrm{Par}(X_i), \boldsymbol{\theta}) \\ &\quad \propto \mathcal{N}\left(\begin{pmatrix} \boldsymbol{x}_{<j} \\ x_j \end{pmatrix}; \begin{pmatrix} -\boldsymbol{a}_j \tau_j b_j \\ \tau_j b_j \end{pmatrix}, \begin{pmatrix} \boldsymbol{a}_j \tau_j \boldsymbol{a}_j^T & -\boldsymbol{a}_j \tau_j \\ -\tau_j \boldsymbol{a}_j^T & \tau_j \end{pmatrix} \right) \mathcal{N}\left(\boldsymbol{h}_{(j-1)}, \boldsymbol{\Lambda}_{(j-1)}\right) \\ &\quad \propto \mathcal{N}\left(\begin{pmatrix} \boldsymbol{x}_{<j} \\ x_j \end{pmatrix}; \begin{pmatrix} -\boldsymbol{a}_j \tau_j b_j \\ \tau_j b_j \end{pmatrix}, \begin{pmatrix} \boldsymbol{a}_j \tau_j \boldsymbol{a}_j^T & -\boldsymbol{a}_j \tau_j \\ -\tau_j \boldsymbol{a}_j^T & \tau_j \end{pmatrix} \right) \\ &\qquad \times \mathcal{N}\left(\begin{pmatrix} \boldsymbol{x}_{<j} \\ x_j \end{pmatrix}; \begin{pmatrix} \boldsymbol{h}_{(j-1)} \\ 0 \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Lambda}_{(j-1)} & 0 \\ 0 & 0 \end{pmatrix} \right). \end{aligned}$$

Using the result from Equation (2.7) allows for the parameters to be summed:

$$\propto \mathcal{N}\left(\begin{pmatrix}\boldsymbol{x}_{<j}\\ x_j\end{pmatrix}; \begin{pmatrix}\boldsymbol{h}_{(j-1)} - \boldsymbol{a}_j\tau_j b_j\\ \tau_j b_j\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Lambda}_{(j-1)} + \boldsymbol{a}_j\tau_j\boldsymbol{a}_j^T & -\boldsymbol{a}_j\tau_j\\ -\tau_j\boldsymbol{a}_j^T & \tau_j\end{pmatrix}\right) \tag{2.10}$$

This means we can construct the prior distribution of the model by specifying roots, followed by each node in turn based on its parents. The parameters of the canonically-parameterised distribution are iteratively updated as each node is added, such that the parameters after the j-th node is added will be,

$$\boldsymbol{h}_{(j)} = \begin{pmatrix}\boldsymbol{h}_{(j-1)} - \boldsymbol{a}_j\tau_j b_j\\ \tau_j b_j\end{pmatrix}; \quad \boldsymbol{\Lambda}_{(j)} = \begin{pmatrix}\boldsymbol{\Lambda}_{(j-1)} + \boldsymbol{a}_j\tau_j\boldsymbol{a}_j^T & -\boldsymbol{a}_j\tau_j\\ -\tau_j\boldsymbol{a}_j^T & \tau_j\end{pmatrix}.$$

The final matrix for $\boldsymbol{\Lambda}$ should be sparse as $\boldsymbol{a}$ is usually sparse as each node is added. Special care must be taken with the indexes of $\boldsymbol{h}$ and $\boldsymbol{\Lambda}$ when stored in a computer since the dimension of these parameters will increase as each node is added.

2.1.6 Conditioning on the observations

Wilkinson and Yeung (2004) also explain how the full conditional distribution for the latent variables of the model can be obtained once the model has been conditioned on the Gaussian observations. Recall that for a linear Gaussian model, the observations are Normally distributed with mean parameter equal to a linear combination of the latent variables, as defined in Equation (2.2). Since the observations are all conditionally independent given the latent variables and parameters, we can say,

$$y_i \mid \boldsymbol{x}, \boldsymbol{\theta} \sim \mathrm{N}((\boldsymbol{a}_i)^T\boldsymbol{x} + b_i, \tau^{-1}), \tag{2.11}$$

where $\boldsymbol{a}_i$ is the i-th column from matrix $\mathbf{A}$. The method will proceed in a similar fashion to how the prior structure was created in Section 2.1.5, and assumes that the prior distribution has already been constructed.

The intention is to find the posterior distribution for the latent values after the j-th observation is added:

$$\boldsymbol{x} \mid \boldsymbol{\theta}, y_1, \ldots, y_j \sim \mathcal{N}\big(\boldsymbol{h}_{(j)}^{(y)}, \boldsymbol{\Lambda}_{(j)}^{(y)}\big),$$

where $\boldsymbol{h}_{(j)}^{(y)}$ and $\boldsymbol{\Lambda}_{(j)}^{(y)}$ are the canonical parameters that have so far been conditioned on j observations.

Multiplying the current density by a Gaussian observation density is analogous to adding a Gaussian node to construct the prior distribution. By extension of the result in Equation (2.10), the joint-density for the current posterior and the j-th observation is,

$$\pi(\boldsymbol{x}, y_j \mid \boldsymbol{\theta}, y_{<j}) \propto \mathcal{N}\left(\begin{pmatrix}\boldsymbol{x}\\ y_j\end{pmatrix}; \begin{pmatrix}\boldsymbol{h}_{(j-1)}^{(y)} - \boldsymbol{a}_j\tau_j b_j\\ \tau_j b_j\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Lambda}_{(j-1)}^{(y)} + \boldsymbol{a}_j\tau_j\boldsymbol{a}_j^T & -\boldsymbol{a}_j\tau_j\\ -\tau_j\boldsymbol{a}_j^T & \tau_j\end{pmatrix}\right).$$

By dropping all dependence on y_j, we can find the full conditional distribution for $\boldsymbol{x}$:

$$\begin{aligned}
&\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, y_1, \ldots, y_j)\\
&\quad\propto \exp\left\{-\frac{1}{2}\left[\begin{pmatrix}\boldsymbol{x}^T & y_j\end{pmatrix}\begin{pmatrix}\boldsymbol{\Lambda}_{(j-1)}^{(y)} + \boldsymbol{a}_j\tau_j\boldsymbol{a}_j^T & -\boldsymbol{a}_j\tau_j\\ -\tau_j\boldsymbol{a}_j^T & \tau_j\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ y_j\end{pmatrix}\right.\right.\\
&\qquad\qquad\left.\left. - 2\begin{pmatrix}\boldsymbol{x}^T & y_j\end{pmatrix}\begin{pmatrix}\boldsymbol{h}_{(j-1)}^{(y)} - \boldsymbol{a}_j\tau_j b_j & \tau_j b_j\end{pmatrix}\right]\right\}\\
&\quad\propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{x}^T(\boldsymbol{\Lambda}_{(j-1)}^{(y)} + \boldsymbol{a}_j\tau_j\boldsymbol{a}_j^T)\boldsymbol{x} - 2\boldsymbol{x}^T(\boldsymbol{h}_{(j-1)}^{(y)} - \boldsymbol{a}_j\tau_j b_j)\right]\right\}\\
&\quad\propto \mathcal{N}(\boldsymbol{x}; \boldsymbol{h}_{(j-1)}^{(y)} - \boldsymbol{a}_j\tau_j b_j, \boldsymbol{\Lambda}_{(j-1)}^{(y)} + \boldsymbol{a}_j\tau_j\boldsymbol{a}_j^T)\\
&\quad\propto \mathcal{N}(\boldsymbol{x}; \boldsymbol{h}_{(j)}^{(y)}, \boldsymbol{\Lambda}_{(j)}^{(y)}).
\end{aligned} \tag{2.12}$$

We use this result to find that the canonical parameters can be iteratively updated upon each new observation. To condition on the j-th observation, the parameters are updated to,

$$\boldsymbol{h}_{(j)}^{(y)} = \boldsymbol{h}_{(j-1)}^{(y)} - \boldsymbol{a}_j\tau_j b_j, \quad \boldsymbol{\Lambda}_{(j)}^{(y)} = \boldsymbol{\Lambda}_{(j-1)}^{(y)} + \boldsymbol{a}_j\tau_j\boldsymbol{a}_j^T.$$

At each stage, $\boldsymbol{h}_{(j-1)}^{(y)}$ and $\boldsymbol{\Lambda}_{(j-1)}^{(y)}$ will be known, except when conditioning on the first observation, in which case the parameters are updated from the canonical parameters from the completed prior distribution, $\boldsymbol{h}$ and $\boldsymbol{\Lambda}$.

2.1.7 GDAGsim

`GDAGsim` is a software package that makes it easier to perform various computations on models with a Gaussian DAG structure. Based on the user's specification of the model DAG, the software can handle the construction of the underlying sparse precision matrix and simulating from the model among other tasks, following methods described by Wilkinson and Yeung (2004). This allows the user to perform various block sampling methods with the assistance of the software. The sparse matrix algorithms used by the software are crucial to the efficiency of the software.

The original `GDAGsim` package (Wilkinson, 2002) was written in C, using the Meschach matrix library (Stewart and Leyk, 1994) for sparse matrix operations and the GNU Scientific Library (Galassi et al., 2009). We have completely rewritten `GDAGsim` in Java, giving a new version called `GDAGsimJ` which uses Parallel Colt (Wendykier and Nagy, 2010) to handle the storage and operations for both dense and sparse matrices. Parallel Colt performs all operations as pure Java code, eliminating the dependency for external linear algebra packages to be installed.

The functionality of `GDAGsimJ` has a number of additional features over its predecessor, `GDAGsim`, such as the ability to permute the underlying sparse matrix, which will be explained in more detail in Section 2.3.2. Since `GDAGsimJ` can perform all tasks that `GDAGsim` can and more, we will usually refer to `GDAGsim` for tasks that could be performed using `GDAGsimJ` and `GDAGsim`. We will also make reference to "GDAG models" and "GDAG methods" to refer to models contructed in `GDAGsim` and inference methods performed using `GDAGsim`.

The package can be used to specify the dependencies of each node in turn to build up the structure of the model, using the methods described in Sections 2.1.5 and 2.1.6. Roots are defined first—these being nodes (variables) that have no parents to depend upon. Nodes are then added with their dependencies upon existing roots and nodes specified. With this structure prepared, the model can be pre-processed before observations are added to the model, before final processing is performed on a fully specified model. After processing, block simulations and various density calculations can then be obtained from the model using the software.

An outline of how a model is specified by the user in `GDAGsim` and `GDAGsimJ` is provided in Algorithm 5 of Appendix A.

2.2 Latent Gaussian models

We can relax an assumption from the linear Gaussian models introduced in Section 2.1, which allows for more versatile models to be considered. This is achieved by removing the restriction requiring observations $\boldsymbol{y}$ to follow a Gaussian distribution. Instead, we consider observations to come from a more general likelihood function, while maintaining the requirement for observations to be conditionally independent given latent values and model parameters. Now, we can define,

$$\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \prod_{i=1}^{n} \pi(y_i \mid x_i, \boldsymbol{\theta}) \tag{2.13}$$

for observation values y_i, $i = 1, \ldots, n$, and general likelihood function $\pi(\cdot)$. This causes Equation (2.2) to take the form,

$$y_i \mid x_i, \boldsymbol{\theta} \sim \text{Dist}(g(x_i), \sigma_\varepsilon(\boldsymbol{\theta})),$$

for suitable link function, $g(\cdot)$, and likelihood distribution $\text{Dist}(\cdot)$.

The facility to use general link functions permits more flexibility than a linear Gaussian model, in which y_i is required to have Gaussian distribution with mean μ_i provided by an identity link function to a single latent variable, i.e. $\mu_i = x_i$.

A consequence of permitting a general link function and observation distribution is that many of the convenient properties found in linear Gaussian models are invalidated. While the prior distribution can still be constructed by the same methods explained in Section 2.1.5, we can no longer use the methods described in Section 2.1.6 to condition the model on the observations. Therefore, `GDAGsim` is still useful for constructing the prior distribution of the model, but not for conditioning on the observations.

An alternative method for conditioning the model on the observations can be found by adopting techniques discussed by Rue and Held (2005). Performing a Gaussian approximation enables for modal values of the latent field to be approximated using an iterative optimisation, while simultaneously adjusting values of the precision matrix diagonal to account for increased precision from the data. The method for performing this will be discussed later in Section 3.2.2.

Gaussian approximations can fail to account for skew in the latent densities; such inaccuracy can be mitigated by using a Laplace approximation or simplified-

Laplace approximation as described by Rue et al. (2009). The Laplace approximation is most accurate of these approximations but comes with the greatest computational cost, while the simplified-Laplace approximation provides a middle-ground between the Laplace and Gaussian approximations in terms of cost and accuracy.

As part of a deterministic inference method such as INLA (Rue et al., 2009), the increased accuracy of the Laplace or simplified Laplace approximation is desirable despite the increased computational cost. When used as part of an MCMC algorithm, a Metropolis-Hastings step can be used to ensure the exact posterior distribution of the latent values are targeted despite the inaccuracy in the approximation of the latent variables. The minimal computational cost of the Gaussian approximation is desirable for use in a fast MCMC scheme, since this approximation would be performed at every iteration.

2.3 Numerical operations

2.3.1 Cholesky decomposition

The Cholesky decomposition provides an efficient way to factorise a square, Hermitian, positive-definite matrix, using fewer floating-point operations (*flop*[1]) than the QR decomposition. Brezinski and Tournès (2014) explain that the method was developed by André-Louis Cholesky and written in an unpublished manuscript[2] before the method was published by Benoît (1924) six years after the death of Cholesky.

For a square, Hermitian, positive-definite matrix $\mathbf{A}$, we can perform a Cholesky decomposition to find unique triangular factors such that $\mathbf{A} = \mathbf{L}\mathbf{L}^*$, where $\mathbf{L}$ is a lower-triangular matrix and $\mathbf{L}^*$ is the conjugate transpose of $\mathbf{L}$. In the context of precision matrices, which we will use Cholesky decompositions to factorise, these matrices will always contain real valued elements, meaning the Hermitian requirement simplifies to a symmetry requirement. Therefore, a p-dimensional square, real-valued symmetric matrix, $\mathbf{A}$, can be factorised as $\mathbf{A} = \mathbf{L}\mathbf{L}^T$.

[1] We define "flop" written in the singular form to be "floating-point operation(s)"—with emphasis that "operation(s)" may refer to both singular or plural form—to prevent confusion with the term "flops" which is commonly defined as "floating-point operations *per second*".

[2] Cholesky's handwritten manuscript has been scanned and translated from French to English in Brezinski and Tournès (2014).

The values for the elements in the lower-traingular factor $\mathbf{L}$ are calculated as,

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j} |l_{jk}|^2}, \quad l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}^T \right) / l_{jj}, \tag{2.14}$$

for $i = (j+1), \ldots, (j+p)$. Performing a Cholesky decomposition on a dense matrix would require $n^3/3$ flop, compared to $2n^3/3$ for a QR decomposition (Björck, 2014). Variations of the algorithm are available to perform the Cholesky decomposition on row-ordered and column-ordered matrices, and both will produce equivalent numerical factors regardless of which ordering the decomposition is performed in (Björck, 2014). Methods to compute a Cholesky decomposition in parallel on multi-core processors have been created with George et al. (1986) suggesting the most efficient method to be the 'column-Cholesky' method since this gives the least amount of processor idle time.

Aside from always being square and symmetric, a precision matrix will usually be positive-definite, but numerical instability can cause certain precision matrices to appear otherwise to a computer, usually where values close to zero feature on the diagonal of the matrix. In these situations, a square-root free Cholesky decomposition could be performed, such that matrix $\mathbf{A}$ is decomposed as $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$, where $\mathbf{D}$ is a diagonal matrix. This was a generalisation of the Cholesky method called the 'unsymmetrical Choleski [sic] method' by Turing (1948).

2.3.2 Permutation matrices for sparse Cholesky decompositions

The positioning of the non-zero elements in the sparse precision matrix, $\mathbf{Q}$, can lead to a problem known as *fill-in*, where values which were originally zero end up with non-zero values in those same positions after the Cholesky decomposition is performed. This is demonstrated in Figure 2.4. Where non-zero elements are in rows or columns towards the top and left of the matrix $\mathbf{Q}$, there is severe fill-in throughout the Cholesky factor $\mathbf{L}$, as shown in Figure 2.4a. Conversely, in Figure 2.4b, we see minimal fill in when non-zero elements in $\mathbf{Q}$ appear on the bottom/right rows and columns of the matrix.

For certain models where `GDAGsim` is employed, precision matrices will often end up in a form resembling that in Figure 2.4a. This occurs when the root of the model is specified early on to allow for dependent notes to later be conditioned

$$\mathbf{Q} = \begin{pmatrix} \times & \times & \times & \times & \times & \dots & \times & \times & \times \\ \times & \times & \times & & & & & & \\ \times & \times & \times & \times & & & & & \\ \times & & \times & \times & \times & & & & \\ \times & & & \times & \times & \times & & & \\ \vdots & & & \ddots & \ddots & \ddots & & & \\ \times & & & & \times & \times & \times & & \\ \times & & & & & \times & \times & \times & \\ \times & & & & & & \times & \times & \end{pmatrix}, \mathbf{L} = \begin{pmatrix} \times & & & & & & & & \\ \times & \times & & & & & & & \\ \times & \times & \times & & & & & & \\ \times & \times & \times & \times & & & & & \\ \times & \times & \times & \times & \times & & & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ \times & \times & \times & \times & \times & \times & \times & & \\ \times & \times & \times & \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \end{pmatrix}$$

(a) An example matrix $\mathbf{Q}$ that will suffer from substantial fill-in on Cholesky factorisation.

$$\mathbf{Q} = \begin{pmatrix} \times & \times & & & & & & \times \\ \times & \times & \times & & & & & \times \\ & \times & \times & \times & & & & \times \\ & & \ddots & \ddots & \ddots & & & \vdots \\ & & & \times & \times & \times & & \times \\ & & & & \times & \times & \times & \times \\ & & & & & \times & \times & \times \\ \times & \times & \times & \dots & \times & \times & \times & \times \end{pmatrix}, \mathbf{L} = \begin{pmatrix} \times & & & & & & & & \\ \times & \times & & & & & & & \\ & \times & \times & & & & & & \\ & & \times & \times & & & & & \\ & & & \times & \times & & & & \\ & & & & \ddots & \ddots & & & \\ & & & & & \times & \times & & \\ & & & & & & \times & \times & \\ \times & \times & \times & \times & \dots & \times & \times & \times & \times \end{pmatrix}$$

(b) An example matrix $\mathbf{Q}$ that will suffer from no fill-in on Cholesky factorisation.

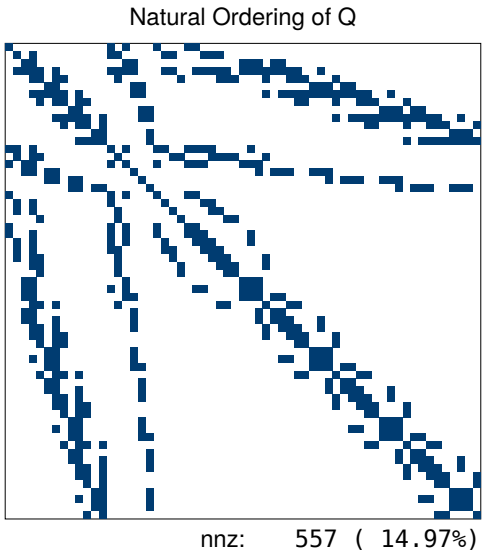
Figure 2.4: Comparison of how the positioning of the non-zero elements, represented by $\times$, in a sparse matrix, $\mathbf{Q}$, can affect the sparsity of its Cholesky factor, $\mathbf{L}$.

on these roots, resulting in multiple non-zero values towards the top-left of the precision matrix $\mathbf{Q}$.

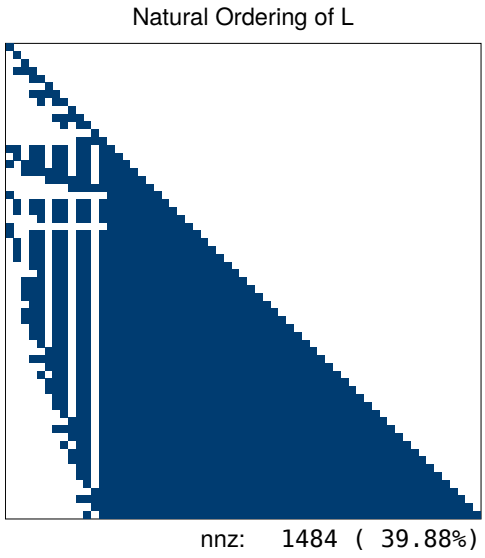
On models with a simple structure, variables can be specified in `GDAGsim` in a more optimal ordering, where such an ordering is found using some careful thought. But on more complicated examples, such an optimal ordering cannot be found trivially. Looking at the matrix in Figure 2.5a, an optimal re-ordering of rows and columns that will minimise Cholesky fill-in is not easy to spot. The location of the non-zero entries for Figure 2.5a come from the 'Can 61' dataset of the Harwell-Boeing collection[3] with values modified to produce a positive-definite matrix.

It is desirable that the user of the `GDAGsim` software should be free to specify nodes in an intuitive order, without being penalised with poor Cholesky decomposition performance caused by a bad ordering of variables. The solution is to use an algorithm that finds an optimal permutation matrix, which can be used to re-order the precision matrix $\mathbf{Q}$ before the Cholesky decomposition is taken. Therefore the aim is to find a permutation matrix $\mathbf{P}$, such that $\mathbf{C} = \mathbf{PQP}^T$ gives

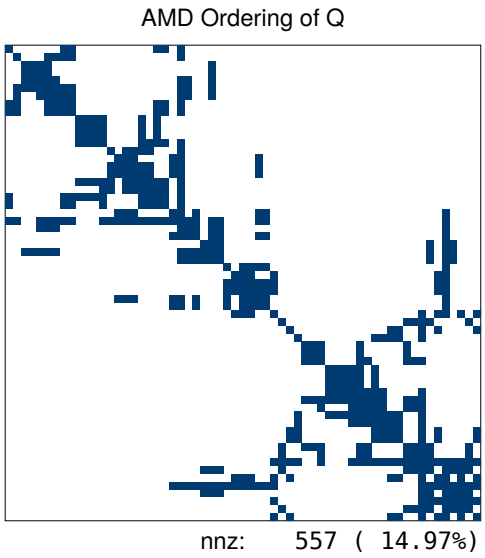
[3] `http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/cannes/can___61.html`



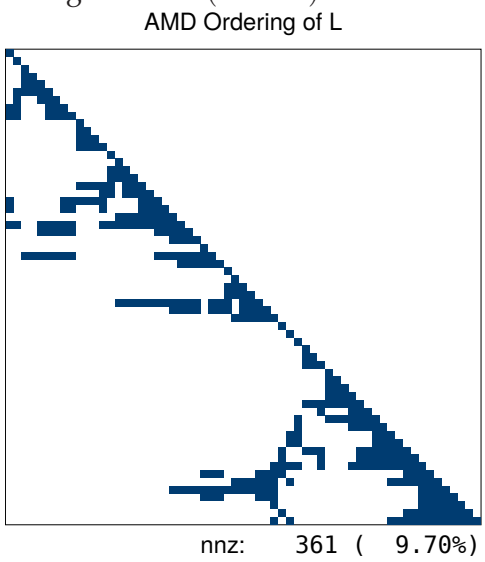
(a) The naturally ordered matrix **Q**. Number of non-zero entries in lower-triangle is 309 (16.34%)



(b) The Cholesky factor **L** of the naturally-ordered **Q** in Figure 2.5a. Number of non-zero entries in lower-triangle is 1484 (78.48%)



(c) The AMD ordered version of **Q** in Figure 2.5a. Number of non-zero entries in lower-triangle remains 309 (16.34%)



(d) The Cholesky factor **L** of the AMD-ordered **Q** in Figure 2.5c. Number of non-zero entries in lower-triangle is 361 (19.09%)

Figure 2.5: The location of non-zero entries in an example sparse, symmetric, positive-definite 61×61 matrix. In the full matrix, **Q**, there are 309 non-zero elements (16.34%). For a fairer comparison, each plot is captioned with the quantity and percentage of non-zero values in the *lower-triangle* only, since the Cholesky factors only occupy these elements.

the minimum amount of fill-in when taking the Cholesky factor of $\mathbf{C}$. Unfortunately, as demonstrated by Yannakakis (1981) this task is NP-complete, but there are heuristic algorithms that find an *approximately* optimum permutation.

The approximate minimum degree (AMD) algorithm (Amestoy et al., 1996, 2004) is a commonly used algorithm for determining the optimum permutation matrix before performing a Cholesky decomposition. As there is no algorithm that can find a 'perfect' permutation, plenty of attempts to optimise minimum degree algorithms have been contributed over the years (George and Liu, 1989).

As can be seen in Figure 2.5c, the algorithm attempts to reorder the rows and columns of the matrix such that: rows with the highest density are placed in the bottom/right of the matrix; the bandwidth is minimised by placing non-zero values close to the diagonal; large rectangular blocks of non-zero values appear in the permuted matrix, which will remain completely sparse in the Cholesky factor.

Figure 2.5b, which shows the non-zero values in the Cholesky factor of the naturally-ordered matrix in Figure 2.5a, demonstrates the substantial fill-in encountered on the Cholesky factor—over three-quarters of the lower-triangle of the matrix is now non-zero, compared to the original 16.34%. By performing a Cholesky decomposition on an AMD-permuted version of Figure 2.5c, an immediate improvement can be seen in Figure 2.5d with most of the sparsity being maintained and the number of non-zero values being less than one-third of what would be obtained without the permutation.

An increased number of non-zero values would carry a penalty in computation time and memory usage whenever the Cholesky factor is used in any computation. Consequently, the performance gains from using minimum degree ordering algorithms will generally outweigh the cost of running the algorithm to find an approximately optimal permutation.

With the optimum permutation found, linear systems can be solved as,

$$(\mathbf{PAP}^T)(\mathbf{Px}) = (\mathbf{Pb}),$$

which will be quicker to compute if the Cholesky factor of $(\mathbf{PAP}^T)$ can maintain most of the sparsity held in $\mathbf{A}$.

2.3.3 Density calculations

Many of the multivariate Gaussian density calculations can be performed efficiently once a sparse Cholesky factor of the precision matrix has been obtained. Recall from Equation (2.6) that the multivariate Gaussian density in its canonical parameterisation is given by

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{h}, \boldsymbol{\Lambda}) \propto \exp\left\{-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Lambda}\boldsymbol{x} + \boldsymbol{h}^T\boldsymbol{x}\right\}.$$

The most computationally demanding part of this calculation will be the calculation of $\boldsymbol{x}^T\boldsymbol{\Lambda}\boldsymbol{x}$. Take $\boldsymbol{\Lambda} = \mathbf{L}\mathbf{L}^T$, obtained by a Cholesky decomposition. Then we can simplify the calculation to,

$$\boldsymbol{x}^T\boldsymbol{\Lambda}\boldsymbol{x} = \boldsymbol{x}^T\mathbf{L}\mathbf{L}^T\boldsymbol{x} = \left(\mathbf{L}^T\boldsymbol{x}\right)^T\left(\mathbf{L}^T\boldsymbol{x}\right) = \boldsymbol{v}^T\boldsymbol{v},$$

where $\mathbf{L}^T\boldsymbol{x} = \boldsymbol{v}$. Multiplication by the triangular matrix is less demanding than multiplication by the full matrix.

The same method can be extended when calculating the density in the moment-precision parameterisation, from Equation (2.5). Taking the Cholesky decomposition of $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$ and defining $\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{\mu})$, we can complete the calculation in a similar manner:

$$(\boldsymbol{x} - \boldsymbol{\mu})^T\mathbf{Q}(\boldsymbol{x} - \boldsymbol{\mu}) = \boldsymbol{z}^T\mathbf{L}\mathbf{L}^T\boldsymbol{z} = \left(\mathbf{L}^T\boldsymbol{z}\right)^T\left(\mathbf{L}^T\boldsymbol{z}\right) = \boldsymbol{v}^T\boldsymbol{v},$$

where $\mathbf{L}^T\boldsymbol{z} = \boldsymbol{v}$.

The density also requires the calculation of $|\mathbf{Q}|^{1/2}$, which we can obtain at little extra cost once the Cholesky factor is known:

$$|\mathbf{Q}|^{1/2} = \left(|\mathbf{L}| \times |\mathbf{L}|^T\right)^{1/2} = \left(|\mathbf{L}|^2\right)^{1/2} = |\mathbf{L}|.$$

Since $|\mathbf{L}|$ is a triangular matrix, its determinant can be computed as the product of its diagonal elements. Hence,

$$|\mathbf{Q}|^{1/2} = \prod_i l_{ii}.$$

Algorithm 6 in Appendix A details how to calculate the log-density for a mul-

tivariate Gaussian density represented in the moment parameterisation.

2.3.4 Switching from canonical to moment parameterisation

We find it easiest to construct a Gaussian DAG model in `GDAGsim` by updating the canonical parameters as each node is added. However, we may wish to convert the model to a moment parameterisation in order to obtain a mean vector or precision matrix; this parameterisation may be used to condition the model on non-Gaussian observations in a latent Gaussian model. To do this, we prepare the Cholesky decomposition of the canonical parameter matrix, $\mathbf{\Lambda} = \mathbf{L}\mathbf{L}^T$.

Computation of the mean (moment) parameter from the canonical parameters, $\boldsymbol{h}$ and $\mathbf{\Lambda}$, is given by,

$$\begin{aligned} \boldsymbol{\mu} &= \mathbf{\Lambda}^{-1}\boldsymbol{h} \\ \mathbf{\Lambda}\boldsymbol{\mu} &= \boldsymbol{h} \\ \mathbf{L}\mathbf{L}^T\boldsymbol{\mu} &= \boldsymbol{h}. \end{aligned} \tag{2.15}$$

Let $\mathbf{L}^T\boldsymbol{\mu} = \boldsymbol{v}$. Then (2.15) simplifies to,

$$\mathbf{L}\boldsymbol{v} = \boldsymbol{h}.$$

Forward-solving for $\boldsymbol{v}$ allows us to return to finding $\boldsymbol{\mu}$ by back-solving the original substitution, $\mathbf{L}^T\boldsymbol{\mu} = \boldsymbol{v}$. Solving a set of triangular problems provides a more efficient way to obtain $\boldsymbol{\mu}$ than attempting to directly invert $\mathbf{\Lambda}$.

2.3.5 Sampling from a multivariate Gaussian density

Suppose we have a d-dimensional multivariate Gaussian density with mean vector, $\boldsymbol{\mu}$, and precision matrix, $\mathbf{Q}$, and we wish to generate a random sample from this distribution. We already have a Cholesky decomposition of the precision matrix, $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$.

Begin by generating d independent realisations from a standard univariate normal distribution, $Z_i \sim \mathrm{N}(0,1)$, $i = 1, \ldots, d$; methods to generate these values are well-established (Box and Muller, 1958). When normalising $\boldsymbol{x}$ to be on the

standard normal scale like the generated values of z, we find,

$$(x-\mu)^T\mathbf{Q}(x-\mu) = (x-\mu)^T\mathbf{L}\mathbf{L}^T(x-\mu) = [\mathbf{L}^T(x-\mu)]^T[\mathbf{L}^T(x-\mu)] \equiv z^T z.$$

Using this result, we know to perform the following steps to create each sample:

$$\begin{aligned} \mathbf{L}^T(x-\mu) &= z && [\text{set } w = (x-\mu)] \\ \mathbf{L}^T w &= z && [\text{Backsolve for } w] \\ x &= w + \mu. \end{aligned}$$

Therefore, realisations can be generated for the cost of a single sparse backward solve if we have already computed the Cholesky factor. A summary of this method is provided in Algorithm 7 of Appendix A.

Chapter 3

Integrated nested Laplace approximations

3.1 Introduction

Performing Bayesian inference will often present integrals which cannot be performed analytically. As a solution to this, Rue et al. (2009) describe the use of Laplace approximations for the problematic densities to be integrated. The method they propose is known as *integrated nested Laplace approximations* (INLA).

Rue et al. (2009) demonstrated that INLA can perform approximate Bayesian inference on latent Gaussian models with a sparse latent structure, often achieving results very close to the densities obtained from long Markov chain Monte Carlo runs in a much quicker time. This has led to its adoption as a means of performing Bayesian modelling in a number of fields. Applications in Ecology have used INLA for Bayesian modelling to identify that increased forest species diversity can reduce the transmission risk of pathogens (Haas et al., 2011), but also that insects and pathogens have a large role to play in maintaining tropical plant diversity (Bagchi et al., 2014). Spatio-temporal models are also popular applications for INLA as they often satisfy the requirement for a sparse latent field, such as the modelling of veterinary data in Switzerland (Schrödle et al., 2011), digital mapping of soil properties in Scotland (Poggio et al., 2016), and investigating links between air pollution and socio-economic status (Hajat et al., 2013). The INLA method was extended by van de Wiel et al. (2012) to allow for joint shrinkage of priors for analysis of RNA sequencing data.

There are restrictions on what models can be handled effectively by the INLA method, with one of the most notable being that the dimension of the model parameters must be small, i.e. $\dim(\boldsymbol{\theta}) \leq 6$, since the numerical integration strategies will struggle in high-dimension spaces. The latent structure can be of very high-dimension, provided it demonstrates a considerable degree of sparsity between latent variables.

3.2 Method

We will present the method for performing integrated nested Laplace approximations as described by Rue et al. (2009). We then use an example later on in Section 3.3 to help demonstrate the calculations in more detail.

There are two main posterior marginals which we are aiming to find. The first are posterior marginals for the elements of the latent field, which can be obtained as follows,

$$\pi(x_j|\boldsymbol{y}) = \int \pi(x_j \mid \boldsymbol{\theta}, \boldsymbol{y})\, \pi(\boldsymbol{\theta} \mid \boldsymbol{y})\, d\boldsymbol{\theta}. \tag{3.1}$$

We are also interested in the posterior marginals for each of the parameters. These are obtained from the joint distribution of all parameters by integrating out all other parameters:

$$\pi(\theta_k \mid \boldsymbol{y}) = \int \pi(\boldsymbol{\theta} \mid \boldsymbol{y})\, d\boldsymbol{\theta}_{\setminus k}, \tag{3.2}$$

where $\boldsymbol{\theta}_{\setminus k}$ denotes the set of parameters with the k-th parameter removed.

In order to obtain these, nested approximations are used:

$$\tilde{\pi}(x_j \mid \boldsymbol{y}) = \int \tilde{\pi}(x_j \mid \boldsymbol{\theta}, \boldsymbol{y}) \tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})\, d\boldsymbol{\theta} \tag{3.3}$$

$$\tilde{\pi}(\theta_k \mid \boldsymbol{y}) = \int \tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})\, d\boldsymbol{\theta}_{\setminus k}, \tag{3.4}$$

where $\tilde{\pi}(\cdot \mid \cdot)$ denotes an approximated distribution conditional on its arguments. This will require obtaining the approximated distributions $\tilde{\pi}(x_j \mid \boldsymbol{\theta}, \boldsymbol{y})$ and $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$.

An approximation for $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$ can be derived by factorising the joint distribution as,

$$\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y}) = \pi(\boldsymbol{y})\pi(\boldsymbol{\theta} \mid \boldsymbol{y})\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y}),$$

and rearranging for $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$:

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})\pi(\boldsymbol{y})}. \tag{3.5}$$

The equality of Equation (3.5) highlights that since the left-hand side of this equation does not depend on the latent variables $\boldsymbol{x}$, the right hand side of this equation must also not depend on $\boldsymbol{x}$. This is demonstrated by the factorisation performed in Appendix B.1. As a result, Equation (3.5) can be evaluated at any convenient value of $\boldsymbol{x}$ we choose. The most stable approximation can then be obtained where the density is well-represented by the modal configuration of $\boldsymbol{x}^\star(\boldsymbol{\theta})$. So $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$, the approximation for $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$ in Equation (3.5), is given by,

$$\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \left. \frac{\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y})}{\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})} \right|_{\boldsymbol{x}=\boldsymbol{x}^\star(\boldsymbol{\theta})}, \tag{3.6}$$

where $\boldsymbol{x}^\star(\boldsymbol{\theta})$ is the modal configuration of the latent elements, $\boldsymbol{x}$, in the latent field conditional on a given $\boldsymbol{\theta}$, and $\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})$ is a Gaussian approximation of the full conditional of $\boldsymbol{x}$. Producing the Gaussian approximation is a method described in Section 3.2.2.

3.2.1 Approximating marginal densities of hyperparameters

We will need an approximation of $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ for use in the integrals of Equations (3.3) and (3.4). The numerator of Equation (3.6) can be easily computed using an alternative factorisation of the joint distribution:

$$\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y}) = \pi(\boldsymbol{\theta})\, \pi(\boldsymbol{x} \mid \boldsymbol{\theta})\, \pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) \tag{3.7}$$

where we have that

- $\pi(\boldsymbol{\theta})$ is the hyperparameter distribution.
- $\pi(\boldsymbol{x} \mid \boldsymbol{\theta})$ is the Multivariate Gaussian distribution of all elements of our latent field.
- $\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$ is the likelihood distribution which our observations follow.

During the calculation of Equation (3.6), we will also need to find the modal

configuration of the latent field for the Gaussian approximations. This process is explained in Section 3.2.2.

The density of the approximate posterior marginal of all parameters $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ can be explored using a grid integration method as follows:

1. Maximise the log-density, $\log[\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})]$, with respect to $\boldsymbol{\theta}$, so that a modal value $\boldsymbol{\theta}^\star$ is found. A quasi-Newton method, such as the BFGS algorithm (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970), is useful for this and a gradient function can be found using finite differencing.

2. Calculate the negative Hessian, $\mathbf{H}$, at the parameter density mode, $\boldsymbol{\theta}^\star$. Set $\boldsymbol{\Sigma} = \mathbf{H}^{-1}$ and take an eigendecomposition of $\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$. Use this eigendecomposition to reparameterise for $\mathbf{z}$ as follows,

$$\boldsymbol{\theta}(\mathbf{z}) = \boldsymbol{\theta}^\star + \mathbf{V}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{z}. \tag{3.8}$$

3. Use the new $\mathbf{z}$ parameterisation to create a new set of coordinate axes. Select points along each of these new axes as long as the density is above a threshold of interest, creating a regular lattice of points throughout the density according to the directions of the new coordinate axes for $\mathbf{z}$. Smith et al. (1987) use this reparameterisation as "an appropriate linear transformation, to a new, approximately orthogonal, set of parameters", allowing densities to be efficiently explored according to their shape.

4. A type of kernel density estimation or an interpolant can be created from these points which can be used to perform numerical integration for the posterior marginal distributions of the parameters.

More detail to steps 1–3 are given in Section 3.3.5, while a detailed description of the kernel density estimation can be seen in Section 3.3.6.

3.2.2 Gaussian approximations

In order to obtain the density of $\tilde{\pi}(x_j \mid \boldsymbol{y})$ from Equation (3.4), we need the approximation $\tilde{\pi}(x_j \mid \boldsymbol{\theta}, \boldsymbol{y})$, and the simplest approximation for this is a Gaussian approximation. One such Gaussian approximation would have already been performed when exploring $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$.

We need to find the modal configuration of the elements in the latent field, which we find iteratively using a Newton-Raphson method. Assuming the latent structure is a replacedmMultivariate Gaussian of zero mean, the Gaussian density we are approximating is of the form,

$$\begin{aligned}\tilde{\pi}(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y}) &\propto \exp\left[-\frac{1}{2}\boldsymbol{x}^T\mathbf{Q}(\boldsymbol{\theta})\boldsymbol{x} + \sum_i \log\left\{\pi\left(y_i \mid x_i, \boldsymbol{\theta}\right)\right\}\right] \\ &\propto \exp\left[-\frac{1}{2}\boldsymbol{x}^T\mathbf{Q}(\boldsymbol{\theta})\boldsymbol{x} + \sum_i g_i(x_i)\right], \end{aligned} \tag{3.9}$$

where $g_i(x_i) = \log\left\{\pi\left(y_i \mid x_i, \boldsymbol{\theta}\right)\right\}$.

An initial guess at the modal configuration is made, denoted as $\boldsymbol{\mu}^{(0)}$. Then $g_i(x_i)$ is then expanded around the guess $\boldsymbol{\mu}$ using a Taylor series up to the second order terms,

$$g_i(x_i) \approx g_i\left(\mu_i^{(0)}\right) + b_i x_i - \frac{1}{2}c_i x_i^2 + o\left(x_i^3\right), \tag{3.10}$$

with b_i and c_i both dependent on $\boldsymbol{\theta}$. The aim will be to iteratively solve the equation $[\mathbf{Q} + \text{diag}(\boldsymbol{c})]\boldsymbol{\mu} = \boldsymbol{b}$ for $\boldsymbol{\mu}$ and recomputing the values of b_i and c_i until $\boldsymbol{\mu}$ converges to the modal configuration of the latent variables, $\boldsymbol{x}^\star$. This gives a Gaussian distribution with mean $\boldsymbol{x}^\star$ and precision matrix $\mathbf{Q}^\star = \mathbf{Q} + \text{diag}(\boldsymbol{c})$.

A demonstration of this method applied to an example model is given in Section 3.3.4.

It is useful to find a good initial guess $\boldsymbol{\mu}^{(0)}$ of the modal configuration to reduce the number of iterative steps needed to converge on the modal configuration. Yoon and Wilson (2011) propose a method to get close to the solution of,

$$-\frac{d\,\tilde{\pi}(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})}{d\,\boldsymbol{x}} = 0.$$

This is often not possible, especially where the derivative contains an $\exp\{\boldsymbol{x}\}$ term, so this cannot usually be solved directly for $\boldsymbol{x}$. Their solution is to transform the problem from vector space into the diagonals of a matrix space. This allows for the equation to be solved, before the diagonal values of the matrices are transformed back into vectors and used as the initial guess $\boldsymbol{\mu}^{(0)}$.

Better estimates can be obtained by using a Laplace approximation, where third-order terms of the Taylor expansion are also considered (Gamerman and Lopes, 2006) to account for skew in the latent densities, or simplified-Laplace

approximation instead of the Gaussian approximation. These improved estimates do come at a slight computational cost.

3.3 Application to traffic 'near-miss' example

3.3.1 Near-misses on the Place Charles de Gaulle

To demonstrate the calculations used by INLA, we will apply them to a simple, fictional model.

Place Charles de Gaulle (informally known as the Arc de Triomphe roundabout) is a major road junction in central Paris where 12 roads meet and priority is given to vehicles *entering* the roundabout; this causes congestion and makes it notoriously hazardous to navigate safely. With an interest in minimising hazards on the road, we choose to observe the number of collisions and near-misses[1] that occur every 60 seconds, treating this as Poisson count data. We believe that the rate of this Poisson distribution changes over time depending on an unobserved latent process—which we assume follows an autoregressive process—influenced by some unmeasured mixture of the prevailing traffic flow levels, road surface wetness and average vehicle speed.

We have n observations, $y_i, i = 1, \ldots, n$ from $\boldsymbol{Y} \sim \text{Pois}(\boldsymbol{\lambda})$. We then have a link function $\lambda_i = E_i \exp(\eta_i)$, $i = 1, \ldots, n$, where η_i is the linear predictor and $E_i > 0$ is a known constant (or $\log(E_i)$ is the offset of η_i). The linear predictor is,

$$\eta_i = \alpha + x_i, \qquad i = 1, \ldots, n,$$

where α is an intercept term and $\boldsymbol{x}$ is an AR(1) process with mean 0 and autoregressive constant $|\phi| < 1$:

$$x_i = \phi x_{i-1} + \epsilon_i; \quad |\phi| < 1; \quad \epsilon_i \sim \text{N}(0, \sigma^2); \quad i = 2, \ldots, n. \tag{3.11}$$

[1] we may choose to define a near-miss as an event where a driver has to brake or turn suddenly in reaction to another driver's actions.

3.3.2 Construction of sparse precision matrix

It can be shown that for the AR(1) process given in Equation (3.11), the variance of each term is

$$\text{Var}(x_i) = \text{Var}(\phi x_{i-1} + \epsilon_i) = \frac{\sigma^2}{1-\phi^2}, \tag{3.12}$$

while for covariance between values of $\boldsymbol{x}$ is,

$$\text{Cov}(x_{i-i}, x_i) = \text{Cov}(x_i, x_{i-1}) = \text{Cov}(\phi x_{i-1} + \epsilon_i, x_{i-1}) = \phi \frac{\sigma^2}{1-\phi^2},$$

and it can more generally be shown that,

$$\text{Cov}(x_{i-k}, x_i) = \text{Cov}(x_i, x_{i-k}) = \phi^k \frac{\sigma^2}{1-\phi^2}, \tag{3.13}$$

for $k = 1, \ldots, i-1$. Derivations for Equations (3.12) and (3.13) are provided in Appendix B.2.
The values of Equations (3.12) and (3.13) allow us to find the dense covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ for $\boldsymbol{x}$:

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \frac{\sigma^2}{1-\phi^2} \begin{pmatrix} 1 & \phi & \phi^2 & \cdots & \phi^k \\ \phi & 1 & \phi & \ddots & \phi^{k-1} \\ \phi^2 & \phi & 1 & \ddots & \phi^{k-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \phi^k & \phi^{k-1} & \phi^{k-2} & \cdots & 1 \end{pmatrix}$$

It is more convenient to work with the precision matrix $\mathbf{Q}(\boldsymbol{\theta})$, which shows conditional dependence between variables as opposed to the covariance, since this may produce a sparse matrix, allowing for efficient computations. The precision

matrix for this AR(1) process is,

$$\mathbf{Q}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}) = \frac{1}{\sigma^2}\begin{pmatrix} 1 & -\phi & 0 & \cdots & 0 & 0 & 0 \\ -\phi & \phi^2+1 & -\phi & \ddots & 0 & 0 & 0 \\ 0 & -\phi & \phi^2+1 & \ddots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & \phi^2+1 & -\phi & 0 \\ 0 & 0 & 0 & \ddots & -\phi & \phi^2+1 & -\phi \\ 0 & 0 & 0 & \cdots & 0 & -\phi & 1 \end{pmatrix} \tag{3.14}$$

We will later find it useful to work with the Cholesky decomposition of this matrix. We define $\mathbf{U}$ to be the upper (right) Cholesky factor of $\mathbf{Q}$ so that $\mathbf{Q} = \mathbf{U}^T\mathbf{U}$, and the lower Cholesky factor to be $\mathbf{L} = \mathbf{U}^T$. The upper Cholesky factor for an AR(1) precision matrix as shown in Equation (3.14) is

$$\mathbf{U}(\boldsymbol{\theta}) = \frac{1}{\sigma}\begin{pmatrix} 1 & -\phi & & & \\ & 1 & -\phi & & \\ & & \ddots & \ddots & \\ & & & 1 & -\phi \\ & & & & \sqrt{1-\phi^2} \end{pmatrix} \tag{3.15}$$

Recall from Section 2.3.3 that having the Cholesky factor, either $\mathbf{L}$ or $\mathbf{U}$, allows computation of $|\mathbf{Q}|$ and $\log|\mathbf{Q}|$ with little additional computational cost:

$$|\mathbf{Q}| = \left[\prod_{i=1}^{n} l_{ii}\right]^2 \quad \Longrightarrow \quad \log|\mathbf{Q}| = 2\sum_{i=1}^{n}\log(l_{ii}). \tag{3.16}$$

The prior distributions are placed on $(\theta_1, \theta_2, \theta_3)$ as opposed to being directly placed on (σ, ϕ, μ). In line with a reparameterisation made by Rue et al. (2009), we link θ_1 to σ using,

$$\theta_1 = \log(\kappa) = \log\left[\tau(1-\phi^2)\right] = \log\left(\frac{1-\phi^2}{\sigma^2}\right), \tag{3.17}$$

θ_2 to ϕ using,

$$\theta_2 = \log\left(\frac{1+\phi}{1-\phi}\right), \tag{3.18}$$

and θ_3 is linked directly to μ as $\theta_3 = \mu$.

So when the precision matrix $\mathbf{Q}(\boldsymbol{\theta})$ is constructed for a given (θ_1, θ_2), we convert the values to ϕ and σ before using them in the precision matrix. The values of ϕ and σ are obtained by inverting Equations (3.17) and (3.18):

$$\phi = \frac{2\exp(\theta_2)}{1+\exp(\theta_2)} - 1; \qquad \sigma = \sqrt{\frac{1-\phi^2}{\exp(\theta_1)}}.$$

We place prior distributions on the parameters as follows:

$$\begin{aligned}
\theta_1 &\sim \text{logGamma}(\alpha, \beta), \\
\theta_2 &\sim \text{N}(\mu_\phi, \sigma_\phi^2), \\
\theta_3 &\sim \text{N}(\mu_\mu, \sigma_\mu^2).
\end{aligned}$$

For these examples, we will use vague priors as used by the `INLA` software, for the purposes of verification.

3.3.3 Finding the marginal hyperparameter distribution

With the precision matrix created, we can now begin to construct the approximation $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ as given in Equation (3.6). When including Equation (3.7) for the numerator of this expression we have,

$$\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \left. \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{x} \mid \boldsymbol{\theta})\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})}{\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})} \right|_{\boldsymbol{x}=\boldsymbol{x}^\star(\boldsymbol{\theta})} \tag{3.19}$$

We can calculate each part to this in turn:

- $\pi(\boldsymbol{\theta})$ is the distribution of our parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$. With θ_1 following a log-Gamma distribution and θ_2 and θ_3 following a Gaussian distribution, the density of $\pi(\boldsymbol{\theta})$ simply becomes,

$$\begin{aligned}
\pi(\boldsymbol{\theta}) &= \pi(\theta_1)\pi(\theta_2)\pi(\theta_3) \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)} \exp\left\{\alpha\theta_1 - \beta e^{\theta_1}\right\} \frac{1}{\sigma_\phi\sqrt{2\pi}} \exp\left\{-\frac{(\theta_2-\mu_\phi)^2}{2\sigma_\phi^2}\right\} \\
&\quad \times \frac{1}{\sigma_\mu\sqrt{2\pi}} \exp\left\{-\frac{(\theta_3-\mu_\mu)^2}{2\sigma_\mu^2}\right\}.
\end{aligned}$$

The log of this density is

$$\begin{aligned}\log[\pi(\boldsymbol{\theta})] = \alpha\log(\beta) - \log[\Gamma(\alpha)] + \alpha\theta_1 - \beta e^{\theta_1} \\ - \log(\sigma_\phi) - \frac{1}{2}\log(2\pi) - \frac{(\theta_2-\mu_\phi)^2}{2\sigma_\phi^2} \\ - \log(\sigma_\mu) - \frac{1}{2}\log(2\pi) - \frac{(\theta_3-\mu_\mu)^2}{2\sigma_\mu^2}.\end{aligned} \tag{3.20}$$

- $\pi(\boldsymbol{x} \mid \boldsymbol{\theta})$ is the multivariate Gaussian density of the latent field, conditional on the current value of $\boldsymbol{\theta}$. Assuming a zero-mean latent Gaussian structure ($\boldsymbol{\mu} = \mathbf{0}$) and using a precision matrix $\mathbf{Q}(\boldsymbol{\theta})$ constructed from the current $\boldsymbol{\theta}$, this has density,

$$\pi(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{|\mathbf{Q}(\boldsymbol{\theta})|^{1/2}}{\sqrt{2\pi}^d} \exp\left\{-\frac{1}{2}\boldsymbol{x}^T\mathbf{Q}(\boldsymbol{\theta})\boldsymbol{x}\right\}, \tag{3.21}$$

  which we will evaluate at the mode of $\boldsymbol{x} = \boldsymbol{x}^\star(\boldsymbol{\theta})$. Recalling the log density of $|\mathbf{Q}|$ from Equation (3.16), the log-density of Equation (3.21) is,

$$\log[\pi(\boldsymbol{x} \mid \boldsymbol{\theta})] = \sum_i [\log u_{ii}(\boldsymbol{\theta})] - \frac{d}{2}\log(2\pi) - \frac{1}{2}\boldsymbol{x}^T\mathbf{Q}(\boldsymbol{\theta})\boldsymbol{x}. \tag{3.22}$$

- The likelihood of $\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$ in the case where observations y_i follow a Poisson distribution with parameter $E_i \exp(x_i)$ is,

$$\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \prod_i \frac{(E_i e^{x_i})^{y_i} \exp\left(-E_i e^{x_i}\right)}{y_i!}.$$

  The log of this distribution is

$$\log\left[\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})\right] = \sum_i y_i \log(E_i) + \sum_i y_i x_i - \sum_i E_i e^{x_i} - \sum_i \log(y_i!). \tag{3.23}$$

- The denominator of Equation (3.19) contains the distribution of the Gaussian approximation, $\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})$, which has density,

$$\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y}) = \frac{|\mathbf{Q}^\star(\boldsymbol{\theta})|^{\frac{1}{2}}}{\sqrt{2\pi}^d} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\mathbf{Q}^\star(\boldsymbol{\theta})(\boldsymbol{x}-\boldsymbol{\mu})\right\}, \tag{3.24}$$

where $\mathbf{Q}^\star(\boldsymbol{\theta}) = \mathbf{Q}(\boldsymbol{\theta}) + \text{diag}(\boldsymbol{c})$ which is calculated from the Gaussian approximation. The log of the density in Equation (3.24) is

$$\log[\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})] = \sum_i [\log u^*_{ii}(\boldsymbol{\theta})] - \frac{d}{2}\log(2\pi) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\mathbf{Q}^\star(\boldsymbol{\theta})(\boldsymbol{x}-\boldsymbol{\mu}). \tag{3.25}$$

The log-density of our approximation therefore becomes

$$\begin{aligned}
\log[\tilde{\pi}_G(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{y})] = {} & \alpha\log(\beta) - \log[\Gamma(\alpha)] + \alpha\theta_1 - \beta e^{\theta_1} \\
& - \log(\sigma_\phi) - \frac{1}{2}\log(2\pi) - \frac{(\theta_2-\mu_\phi)^2}{2\sigma_\phi^2} \\
& - \log(\sigma_\mu) - \frac{1}{2}\log(2\pi) - \frac{(\theta_3-\mu_\mu)^2}{2\sigma_\mu^2} \\
& + \sum_i [\log u_{ii}(\boldsymbol{\theta})] - \frac{1}{2}\boldsymbol{x}^T\mathbf{Q}(\boldsymbol{\theta})\boldsymbol{x} \\
& + \sum_i y_i\log(E_i) + \sum_i y_i x_i - \sum_i E_i e^{x_i} - \sum_i \log(y_i!) \\
& - \sum_i [\log u^*_{ii}(\boldsymbol{\theta})] + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\mathbf{Q}^*(\boldsymbol{\theta})(\boldsymbol{x}-\boldsymbol{\mu}),
\end{aligned}$$

which we evaluate at $\boldsymbol{x} = \boldsymbol{x}^\star(\boldsymbol{\theta})$ that we obtain from a Gaussian approximation.

3.3.4 Performing the Gaussian approximation

For the Gaussian approximation, we need to know $g_i(x_i) = \log\{\pi(y_i \mid x_i, \boldsymbol{\theta})\}$ which we can obtain from Equation (3.23):

$$g_i(x_i) = \log[\pi(y_i|x_i,\boldsymbol{\theta})] = y_i\log(E_i) + y_i x_i - E_i e^{x_i} - \log(y_i!).$$

With $\boldsymbol{\mu}^{(i)}$ being the i-th guess of the mode, we expand $g_i(x_i)$ around $\boldsymbol{\mu}$,

$$\begin{aligned}
g_i(x_i) &\approx g_i(\mu_i) + g_i'(\mu_i)(x_i-\mu_i) + \frac{g_i''(\mu_i)}{2}(x_i-\mu_i)^2 \\
&\approx g_i(\mu_i) + g_i'(\mu_i)x_i - g_i'(\mu_i)\mu_i + \frac{1}{2}x_i^2 g_i''(\mu_i) - x_i\mu_i g_i''(\mu_i) + \frac{1}{2}\mu_i^2 g_i''(\mu_i),
\end{aligned}$$

and combining all constant terms,

$$\approx g_i(\mu_i) + x_i \left[g_i'(\mu_i) - \mu_i g_i''(\mu_i)\right] - \frac{1}{2}x_i^2 \left[-g_i''(\mu_i)\right] + \text{const.} \tag{3.26}$$

We now need the values of $g_i'(x_i)$ and $g_i''(x_i)$:

$$g_i'(x_i) = y_i - E_i e^{x_i}, \quad \text{and} \quad g_i''(x_i) = -E_i e^{x_i}.$$

When applying Equation (3.26) to Equation (3.10) we can derive that,

$$\begin{aligned} b_i(\mu_i) &= g_i'(\mu_i) - \mu_i g_i''(\mu_i) \\ &= y_i - E_i e^{\mu_i} + \mu_i E_i e^{\mu_i} \\ &= y_i + (\mu_i - 1) E_i e^{\mu_i}, \end{aligned} \qquad \text{and} \qquad \begin{aligned} c_i(\mu_i) &= -g_i''(\mu_i) \\ &= E_i e^{\mu_i}. \end{aligned}$$

Starting with the initial guess $\boldsymbol{\mu}^{(0)}$, we can solve $[\mathbf{Q} + \text{diag}(\boldsymbol{c})]\,\boldsymbol{\mu}^{(1)} = \boldsymbol{b}$ to find $\boldsymbol{\mu}^{(1)}$. This is repeated until $\boldsymbol{\mu}$ converges to the mode $\boldsymbol{x}^\star$ of a Gaussian distribution with precision matrix $\mathbf{Q}^\star = \mathbf{Q} + \text{diag}(\boldsymbol{c}^\star)$.

An example of the result from a Gaussian approximation for $\boldsymbol{x}$ in this situation is shown in Figure 3.1. When the latent values take low values (generally below zero), the confidence intervals are very wide. This occurs since negative values of x will often result in observations y taking values of zero. Consequently, we can only make a vague guess as to where the actual latent value is, with only the knowledge that x is likely to be negative.

3.3.5 Exploring the density of the parameters

The first stages of this process are described briefly in Section 3.2.1. Having performed a Gaussian approximation at the initial value of $\boldsymbol{\theta}$, which obtains the guess of the modal configuration of the latent field $\boldsymbol{x}^\star(\boldsymbol{\theta})$ along with its precision matrix $\boldsymbol{Q}^\star(\boldsymbol{\theta})$, we now need to find the maximum value of the log-density $\log[\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})]$.

Quasi-Newton optimisers are readily available in several programming languages, but the optimiser may need to be repeatedly called since every time we have optimised our value of $\boldsymbol{\theta}$, we need to repeat the Gaussian approximation at this new value of $\boldsymbol{\theta}$ for updated values of $\boldsymbol{x}^\star(\boldsymbol{\theta})$ and $\boldsymbol{Q}^\star(\boldsymbol{\theta})$. We then repeat the cycle of optimisation on $\boldsymbol{\theta}$ and $\boldsymbol{x}^\star(\boldsymbol{\theta})$ until both have converged to the maximum of the $\log[\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})]$ density. At this point, we say we have found the modal value

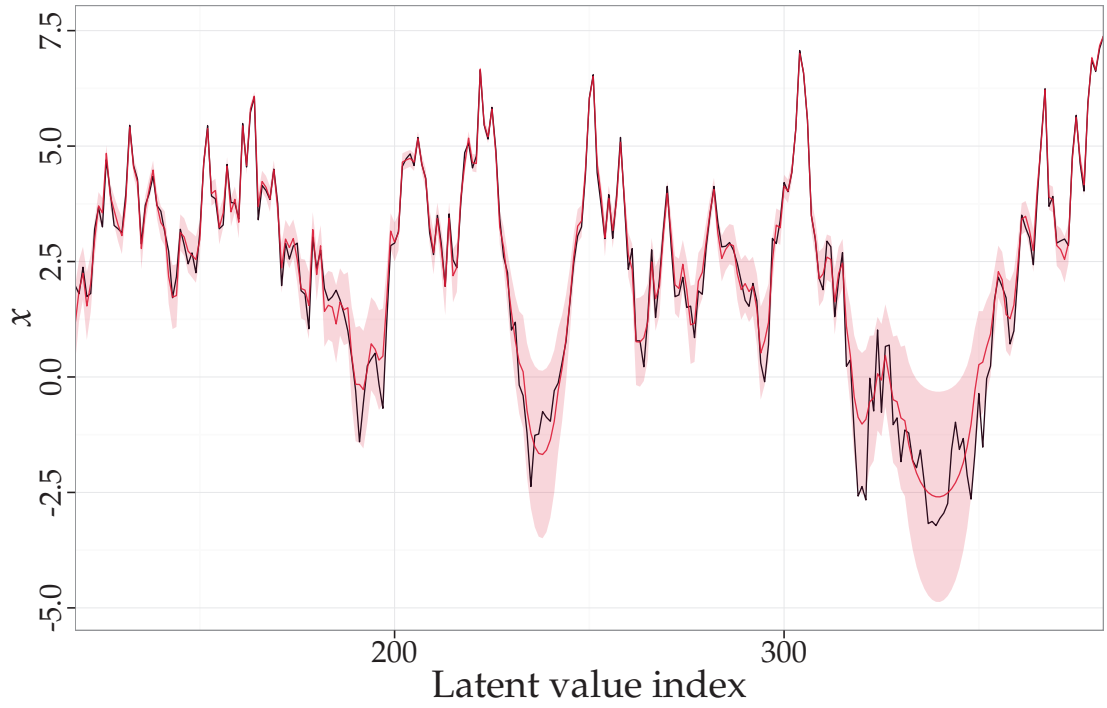


Figure 3.1: An example of the Gaussian approximation guessing the simulated latent field. The "true" latent values, x, are shown by the black line; the mode of the Gaussian approximation, $x^\star$, is shown by the solid red line, with 95% probability intervals given by the shaded red area.

$\boldsymbol{\theta}^\star$.

We find a Hessian at the modal value $\boldsymbol{\theta}^\star$ using finite-differencing. We need to take the negative of this Hessian, so that result is a Hessian $\mathbf{H} > 0$. Let $\boldsymbol{\Sigma} = \mathbf{H}^{-1}$ and take the eigendecomposition of this to get $\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$. To save the computational cost of inverting the Hessian $\mathbf{H}$ before taking the resulting eigendecomposition, simply note that the eigendecomposition of $\mathbf{H} = \boldsymbol{\Sigma}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^T$. We now create a new parametrisation for the model parameters, which we call $\mathbf{z}$ and specify this parameterisation as,

$$\boldsymbol{\theta}(\mathbf{z}) = \boldsymbol{\theta}^* + \mathbf{V}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{z}$$

By exploring around $\mathbf{z}$-space in regular step sizes δ_z, we can efficiently explore around the $\boldsymbol{\theta}$-space. We start by exploring from the mode along each axis in $\mathbf{z}$, until the density drops below a pre-detemined threshold δ_π in each direction. Once this has been done for each $\mathbf{z}$ axis, we then fill in all the remaining points

between the axes following the same grid, but only while these points stay above the threshold.

Once this is done, we will have explored the density to obtain a list of coordinates in $\mathbf{z}$, which we can easily convert back to $\boldsymbol{\theta}$, and the density of each of these points. We can use these $\boldsymbol{\theta}$ coordinates along with their associated densities to perform the numerical integration specified in Equations (3.3) and (3.4).

3.3.6 Performing numerical integration

Recall from Equation (3.4) that we wish to find the marginal of each hyperparameter, which is obtained from performing this integration,

$$\tilde{\pi}(\theta_k \mid \boldsymbol{y}) = \int \tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})\, d\boldsymbol{\theta}_{\setminus k}.$$

Having explored the density of $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ as described in Section 3.3.5, we now have a grid of coordinates where the density is non-negligable, along with the corresponding densities at those locations. As we have explored the densities in $\mathbf{z}$-space, our grid of exploration coordinates are very unlikely to align with the axes in $\boldsymbol{\theta}$ space.

Attempting to sum densities within fixed-width intervals along $\boldsymbol{\theta}$-axes can be unreliable when the regular grid of explored coordinates does not align with the $\boldsymbol{\theta}$-axes. This happens because some of the intervals used may not necessarily contain a consistent amount of evaluated density points, especially where narrow intervals are used.

In order to be able to easily and reliably construct posterior marginals $\tilde{\pi}(\theta_k \mid \boldsymbol{y})$, we will remove the dependency of the coordinates having to align with the $\boldsymbol{\theta}$-axes and the need for selecting fixed interval widths, by using a variation of kernel density estimation.

Kernel density estimation would normally construct a density by assigning a Gaussian density to each individual value of a random sample from a density. Being from a random sample of size N, each Gaussian density constructed around a sampled value would be given equal weighting, $w_i = 1/N$, $i = 1, \ldots, N$. By adding the Gaussian densities, the sample can be used to build a representation of the overall density. The variance of each Gaussian density would also need to be carefully selected, which raises the issue of bandwidth selection—an excessively

large bandwidth will lose too much information about the density, while a bandwidth that is too small will cause a density that is too "spiky" and not smooth enough.

However, in our situation we have a regular grid of points with their own corresponding densities, as opposed to a random scatter of points where each point can be considered equally important. This means adjustments will be needed to the usual method of kernel density estimation, with the main changes being to the weighting applied to each point's Gaussian density.

For simplicity, we describe this method in a two-dimensional context for $\boldsymbol{\theta}$, noting that the methodology extends trivially to higher dimensions. For the example described in Section 3.3.1, we can easily extend the method to work on 3-dimensions. Note that the computational cost of this method does increase dramatically as the dimension of $\boldsymbol{\theta}$ increases.

Consider a two-dimensional density of $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ which is explored in $\mathbf{z}$-space, where each coordinate from the exploration can be written as $(z_{1,p}, z_{2,q})$. Using the transformation from Equation (3.8), we can convert from $\mathbf{z}$-space to $\boldsymbol{\theta}$-space:

$$(z_{1,p}, z_{2,q}) \mapsto (\theta_{1,i}, \theta_{2,j}).$$

We then apply a weight $w_{i,j}$ to each point we explored in the density. The weight assigned to each point is taken from the density at that coordinate:

$$w_{i,j} = \tilde{\pi}(\theta_{1,i}, \theta_{2,j} \mid \boldsymbol{y}).$$

Then the weights are all normalised by the total sum of all weights,

$$w^*_{i,j} = \frac{w_{i,j}}{\sum_i \sum_j w_{i,j}},$$

such that $\sum_i \sum_j w^*_{i,j} = 1$.

A Gaussian distribution is centered around each coordinate, $(\theta_{1,i}, \theta_{2,j})$, from the exploration, with each scaled by their corresponding weights $w^*_{i,j}$. By summing every Gaussian density created, we have constructed the posterior marginals:

$$\tilde{\pi}(\theta_1, \theta_2 \mid \boldsymbol{y}) = \sum_i \sum_j w^*_{i,j} \, \mathrm{N}\Big((\theta_1, \theta_2); (\theta_{1,i}, \theta_{2,j}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\Big),$$

where $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ is the kernel bandwidth matrix. This method can be generalised in the obvious way for models where there is only one hyperparameter or more than two parameters.

We can marginalise this to obtain the marginal posterior distributions of each hyperparameter. For example, the posterior marginal density of $\theta_1 \mid \boldsymbol{y}$ is,

$$\tilde{\pi}(\theta_1 \mid \boldsymbol{y}) = \sum_i \sum_j w^*_{i,j} \, \mathrm{N}\Big(\theta_1; \theta_{1,i}, [\boldsymbol{\Sigma}_{\boldsymbol{\theta}}]_{11}\Big).$$

As mentioned earlier, selection of the correct bandwidth is important in kernel density estimation, and this is still true in this variation of the method. We can gain an idea of what bandwidth to use thanks to Rue et al. (2009), who explain that the eigendecomposition that forms the basis of Equation (3.8), $\boldsymbol{\Sigma} = \mathbf{H}^{-1} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$, would be the covariance matrix for $\boldsymbol{\theta}$ if the density were Gaussian, and also that $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$ whenever this posterior is Gaussian.

Let $\boldsymbol{\Sigma}_{\mathbf{z}}$ be the covariance (bandwidth) matrix for a Gaussian distribution used if the kernel density estimation were to be performed in $\mathbf{z}$-space, and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ be the corresponding standard deviation that would be needed to match this in $\boldsymbol{\theta}$-space. Since we explore $\mathbf{z}$-space in equal step sizes along each direction of $\mathbf{z}$, we intuitively set the standard deviation σ of each Gaussian distribution to be equal to the step size, δ_z. This provides us with a kernel bandwidth matrix of $\boldsymbol{\Sigma}^2_{\mathbf{z}} = \sigma^2\mathbf{I} = \delta_z^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

We now need to convert this bandwidth into something appropriate for $\boldsymbol{\theta}$-space. Recalling that $\boldsymbol{\Sigma}$ would be the covariance matrix in $\boldsymbol{\theta}$-space, we scale the covariance matrix from $\mathbf{z}$-space appropriately:

$$\begin{aligned}
\boldsymbol{\Sigma}_{\boldsymbol{\theta}} &= \boldsymbol{\Sigma}\boldsymbol{\Sigma}_{\mathbf{z}} \\
&= \boldsymbol{\Sigma}\sigma^2\mathbf{I} = \boldsymbol{\Sigma}\delta_z^2\mathbf{I} \\
&= \delta_z^2\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T.
\end{aligned}$$

If we wish to consider bandwidth in terms of the standard deviations, we can obtain,

$$\boldsymbol{\Sigma}^{1/2}_{\boldsymbol{\theta}} = \delta_z\mathbf{V}\boldsymbol{\Lambda}^{1/2},$$

meaning that the kernel density bandwidth for the integration in $\boldsymbol{\theta}$ depends on

the step size, δ_z, used to explore the $\mathbf{z}$-space and the eigendecomposition of the Hessian, $\mathbf{H}$.

3.3.7 Results

For this example, the AR(1) walk of dimension 1000 and its corresponding observations were simulated according to the model with, $\alpha = 2$, $\phi = 0.9$ and $\sigma = 1$. When converted into $\boldsymbol{\theta}$ values using Equations (3.17) and (3.18), they become,

$$\theta_1 = \log\left(\frac{1-\phi^2}{\sigma^2}\right) = \log\left(\frac{1-0.9^2}{1^2}\right) = -1.66$$
$$\theta_2 = \log\left(\frac{1+\phi}{1-\phi}\right) = \log\left(\frac{1+0.9}{1-0.9}\right) = 2.94$$
$$\theta_3 = \alpha = 2$$

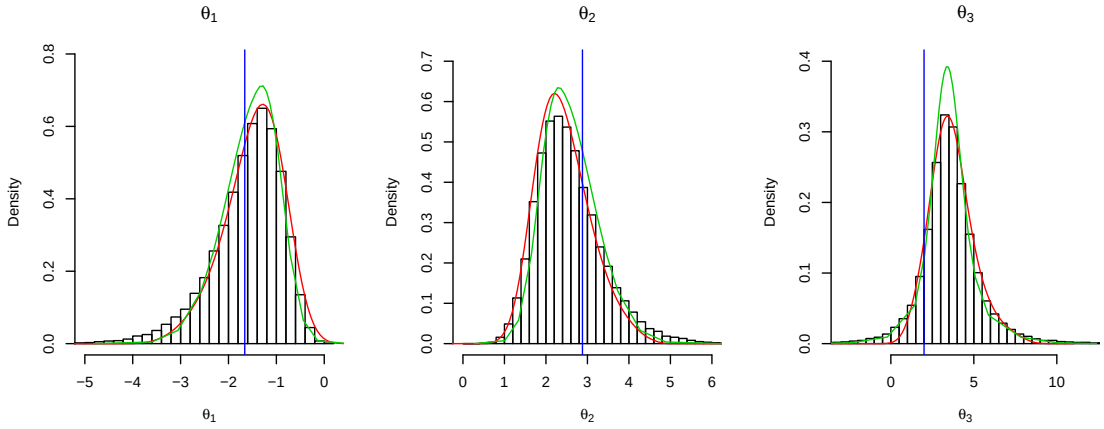


Figure 3.2: Comparison of MCMC results for $\boldsymbol{\theta} = \theta_1, \theta_2, \theta_3$ displayed as histograms, along with "true" parameter values as a blue line, the estimates from the R-INLA package displayed as a red line and a recreation of the INLA method using the weighted kernel-density estimation shown as a green line. The MCMC output consists of 10000 iterations (thinned by 30) produced by JAGS.

Figure 3.2 demonstrates that the methods are able to produce similar inference for the parameter values, with the "true" parameter values well represented by all of the schemes.

The R-INLA package has used its most accurate integration strategy: grid-exploration with the step-size $\delta_z = 0.2$. Our own INLA method attempts to recreate the results of the INLA package, except while the grid-exploration used

here uses the same step-size $\delta_z = 0.2$, this uses a greater log-density threshold δ_π to reduce computational efforts, and used the kernel density method for constructing the marginal density. The fit of the R-INLA package appears to be better than our own attempt due to the greater threshold δ_π allowing more of the density to be explored.

The R-INLA package has arguably produced a better fit to the densities of $\boldsymbol{\theta}$ than our own attempts, which here is a direct consequence of the greater threshold δ_π that has been used. When the R-INLA package is restricted to the same threshold δ_π as our INLA code, the results seem less satisfactory. The R-INLA package fails to represent the density in the tails of some distributions and does not evaluate the θ_2 density above values of 3.3; the cause of this is unclear. Use of the kernel density estimation avoids this and produces a satisfactory fit, despite the compromise lower δ_π threshold.

Chapter 4

Markov chain Monte Carlo methods for Bayesian inference

4.1 Introduction

Under a Bayesian framework, we wish to understand the uncertainty around unseen parameters of the model, $\boldsymbol{\theta}$, where we have observed data, $\boldsymbol{y}$. We may hold prior views about the possible parameter values for $\boldsymbol{\theta}$, which can be described by a probability density function, $\pi(\boldsymbol{\theta})$. Where a suitable model is used, the observed data will depend on the model parameters according to its density function, $f_y(\boldsymbol{y} \mid \boldsymbol{\theta})$. In the case where observations are conditionally independent given any parameters in the model, as will be the case in the latent Gaussian models considered in this thesis, the likelihood function of the data can be computed as,

$$\pi(\boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_i f_y(y_i \mid \boldsymbol{\theta}).$$

The prior beliefs about the parameter distribution can be combined with the observations using Bayes' Theorem to obtain a posterior distribution for the parameters:

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{\pi(\boldsymbol{\theta})\, \pi(\boldsymbol{y} \mid \boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta})\, \pi(\boldsymbol{y} \mid \boldsymbol{\theta})\, d\boldsymbol{\theta}}. \tag{4.1}$$

By integrating over the possible parameter values, the denominator of Equation (4.1) no longer depends on $\boldsymbol{\theta}$ and simply becomes a normalising constant.

We can therefore simplify this to,

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\, \pi(\boldsymbol{y} \mid \boldsymbol{\theta}),$$

which is to say that the posterior density is proportional to the product of the prior and the likelihood density.

Performing the integrals analytically is typically unfeasible if non-conjugate distributions are used, so Markov chain Monte Carlo (MCMC) methods are often employed to do this.

4.2 Using MCMC for Bayesian inference

Monte Carlo integration uses random independent samples to evaluate a definite integral. This can be performed for a multi-dimensional integral, but the increased dimension makes integration more time-consuming as areas of low density are more likely to be explored. In practice, the integration does not necessarily need to be performed using independent samples, but can instead be performed by constructing a Markov chain to explore areas of support for the target distribution (Gilks et al., 1995; Gamerman and Lopes, 2006).

This is due to the memoryless property of Markov chains, which for a stochastic process, $\{X_0, X_1, \ldots\}$, describes that the next value X_{t+1}, given the current and all previous states in the process, will only depend on the current state X_t. Formally, this is noted as,

$$P(X_{t+1} \in A \mid X_0, X_1, \ldots, X_t) = P(X_{t+1} \in A \mid X_t)$$

for any set A and conditional probability $P(\cdot \mid \cdot)$ (Gilks et al., 1995). Under certain conditions, the Markov chain is then able to converge to a unique *stationary* distribution, $\phi(x)$, if the Markov chain satisfies detailed balance, that is,

$$\phi(x)\phi(x, y) = \phi(y)\phi(y, x),$$

for transition kernel $\phi(x, y)$, and any x, y. For Bayesian inference, the stationary distribution of the chain is the target posterior distribution for which inference is desired.

The starting value of the chain will eventually be forgotten thanks to the mem-

oryless property, but all states should be discarded until the chain converges to the stationary distribution; the period discarded prior to convergence is known as a *burn-in period*.

4.3 Gibbs sampler

The Gibbs sampler was first demonstrated by Geman and Geman (1984) on a Markov random field model used for image processing. Gelfand and Smith (1990) later noticed a relationship between the Geman and Geman (1984) concept of a Gibbs sampler, and both the Metropolis-Hastings algorithm (Hastings, 1970) and data augmentation method by Tanner and Wong (1987), before describing how to extend this to general statistical distributions.

To construct the target samples from the joint posterior distribution, a set of full-conditional distributions are produced where each one can be sampled from. Each of these distributions is then repeatedly sampled from based on the current state of all other parameters, until the desired number of samples have been generated from the target distribution. A fixed-sweep or deterministic-sweep Gibbs sampler refers to the case where each parameter is sampled sequentially, which is the method described in Algorithm 1. The random-sweep Gibbs sampler selects a random order in which to update each parameter at each iteration, which creates a reversible MCMC chain that is easier to analyse (Roberts and Sahu, 1997; Brooks, 1998).

Once the chain has converged to the stationary distribution, all samples come from the distribution $\pi(\boldsymbol{\theta})$. Since each parameter value is only dependent on the last parameter values to be generated, the Markov chain is homogeneous.

4.4 Metropolis-Hastings

The ability to perform Gibbs sampling methods from Section 4.3 relies on the ability to know the full conditional distribution and sample from this. This is generally only feasible if conditionally-conjugate priors are used, where the product of certain prior distributions with the likelihood distribution produces tractable conditional posterior distributions that can be sampled from (Gelman et al., 2013). Where this is not possible, the Metropolis-Hastings (MH) algorithm—created by

Algorithm 1: A fixed-sweep Gibbs sampler algorithm.

Initialise $\boldsymbol{\theta}^{(0)} = \left(\theta_1^{(0)}, \ldots, \theta_d^{(0)}\right)$;

for $i := 1$ **to** N **do**

Sample new $\boldsymbol{\theta}^{(i)}$ values from each full-conditional distribution:

$$\begin{aligned}
\theta_1^{(i)} &\sim \pi\left(\theta_1 \mid \theta_2^{(i-1)}, \ldots, \theta_d^{(i-1)}\right) \\
\theta_2^{(i)} &\sim \pi\left(\theta_2 \mid \theta_1^{(i)}, \theta_3^{(i-1)}, \ldots, \theta_d^{(i-1)}\right) \\
\theta_3^{(i)} &\sim \pi\left(\theta_3 \mid \theta_1^{(i)}, \theta_2^{(i)}, \theta_4^{(i-1)}, \ldots, \theta_d^{(i-1)}\right) \\
&\vdots \\
\theta_d^{(i)} &\sim \pi\left(\theta_d \mid \theta_1^{(i)}, \ldots, \theta_{d-1}^{(i)}\right)
\end{aligned}$$

end

Result: N samples from the target distribution for each of the d parameters.

Hastings (1970) as an extension to work by Metropolis et al. (1953)—allows for samples to be generated from a distribution where the density can only be calculated up to a constant of proportionality.

To demonstrate a step of the Metropolis-Hastings method, suppose we have a single parameter, θ, to update; multiple parameters can be updated trivially by repeating the MH-step for each parameter. A proposal distribution, $q(\theta^* \mid \theta)$, is selected for the purpose of generating proposal values where the chain should next move to. The probability of accepting the move based on the proposed value is given by $\alpha(\theta^* \mid \theta) = \min\{1, A\}$, where,

$$A = \frac{\pi(\theta^*)}{\pi(\theta)} \frac{q(\theta \mid \theta^*)}{q(\theta^* \mid \theta)}.$$

If the proposed move is rejected, then θ remains at its existing value and the MCMC routine continues as before. Algorithm 2 shows these steps in more detail.

Where a proposal comes from a symmetric proposal distribution, we find that $q(\theta \mid \theta^*) = q(\theta^* \mid \theta)$, resulting in a simplification of Equation (4.4):

$$A = \frac{\pi(\theta^*)}{\pi(\theta)}.$$

Algorithm 2: A Metropolis-Hastings method for drawing samples from the univariate density, $\pi(\theta)$.

Initialise $\theta^{(0)}$;
for $i := 1$ **to** N **do**
  Generate proposal: $\theta^* \sim q(\theta^* \mid \theta^{(i-1)})$;
  Compute acceptance probability:

$$\alpha\left(\theta^* \mid \theta^{(i-1)}\right) = \min\left\{1, \frac{\pi(\theta^*)}{\pi\left(\theta^{(i-1)}\right)} \frac{q\left(\theta^{(i-1)} \mid \theta^*\right)}{q\left(\theta^* \mid \theta^{(i-1)}\right)}\right\};$$

  Sample $u \sim \mathrm{U}(0,1)$;
  if $u < \alpha(\theta^* \mid \theta^{(i-1)})$ **then**
    $\theta^{(i)} := \theta^*$;
  else
    $\theta^{(i)} := \theta^{(i-1)}$;
  end
end
Result: N samples from the target distribution, $\pi(\theta)$.

This is known as the Metropolis sampler; a special case of the MH method. In this situation, we find that the proposal will always be accepted if the proposed move is to a location of higher density.

A development of using the symmetric proposals in a Metropolis scheme is the use of a random walk sampler. Here we define 'innovations', ω, to be i.i.d. perturbations generated from the symmetric proposal distribution, f_Ω, so that proposals become $\theta^* = \theta + \omega$ and have transition density $q(\theta^* \mid \theta) = f_\Omega(\theta^* - \theta)$ (Tierney, 1994). Common choices for a proposal distribution here include a bounded uniform distribution or Gaussian with zero mean.

The variance of the proposal distribution in the symmetric random walk will influence how rapidly the chain converges. Large innovations will attempt to explore the target density quickly, but the more ambitious moves proposed will be less likely to be accepted, leading to the chain being too "hot". Conversely, small innovations on the proposals lead to higher acceptance probabilities, resulting in a "cold" chain that frequently moves to new values but is slow to explore the entire density. Roberts et al. (1997) suggest methods for tuning the optimum proposal variance on a Gaussian random walk and provide the theoretically optimum acceptance ratio based on the dimension of the problem. Since this requires

additional simulations to be performed and user interaction, Gibbs schemes are considered more convenient to execute when possible.

The acceptance ratio of the Metropolis-Hastings scheme will depend on the choice of proposal distribution—the acceptance ratio is higher when the proposal distribution is similar to the target distribution. In fact, for the special case where the proposal distribution comes from the target density up to a constant term, we discover that Gibbs sampling is a special case of the Metropolis-Hastings step where proposals are always accepted. To demonstrate this, suppose that $q(\theta^* \mid \theta) \propto \pi(\theta^*)$, then the acceptance probability based on Equation (4.4) becomes $\alpha(\theta^* \mid \theta) = \min\{1, A\}$ where,

$$A = \frac{\pi(\theta^*)}{\pi(\theta)} \frac{q(\theta \mid \theta^*)}{q(\theta^* \mid \theta)} \propto \frac{\pi(\theta^*)}{\pi(\theta)} \frac{\pi(\theta)}{\pi(\theta^*)} = 1,$$

resulting in all moves being accepted.

4.5 Block samplers

Poor mixing of MCMC chains can occur where models have a large number of parameters. The problem is exacerbated when the many parameters in the model are highly correlated, leading to a posterior distribution that is difficult to explore when each parameter is updated in turn. This effect can be reduced by reparameterising the model, but there are many situations where this will still fail to yield a satisfactory result.

An alternative method for improving the problematic mixing of a MCMC scheme is to employ blocking—the concept of simultaneously updating a group, or 'block', of similar parameters as a single move during the MCMC scheme. The ability to do this requires knowledge of the joint conditional distribution of all parameters in the block to be updated. The block of parameters can be of high-dimension with conditional dependence between these parameters, which in the linear Gaussian case can be represented in the precision matrix. As a consequence of working with this matrix, performing a block update is more computationally expensive than performing a Gibbs update.

Roberts and Sahu (1997), which provides a nice example of how using an equivalent reparameterisation of a model can improve mixing, also summarises the ideas and benefits of the blocking method nicely with two quotes. The first

describes a general belief about the positive correlation between block size and the rate of chain's convergence,

> "the larger the blocks that are updated simultaneously – the faster the convergence." (Amit and Grenander, 1991)

The second attempts to explain why blocking helps combat the poor mixing caused by highly correlated variables:

> "[Blocking] moves any high correlation [between variables] from the Gibbs sampler over to the random vector generator." (Seewald, 1992)

This highlights how simulating multiple parameters at once, while incorporating the dependence between variables, leads to MCMC chains which can explore in the directions of the correlated density. In cases where there are highly correlated variables in the model, the improved mixing offered by blocking methods can considerably outweigh the additional computational overheads associated with its implementation.

In many of the models, like that featured in Figure 2.2, there will be a correlation between variables. While the precision matrix, like in Equation (2.3), can be very sparse, the covariance matrix would be dense, indicating correlation between those $\boldsymbol{x}$. In these cases, we may find that simulating $\boldsymbol{x}$ as a block will improve mixing. It is also worth considering that the parameters of the model $\boldsymbol{\theta}$ may also be highly correlated, so it may be beneficial to also simulate these as a block.

Where a block of parameters to be updated are independent, blocking schemes will demonstrate no advantage over a standard Gibbs sampler, as the marginal distribution of the block is equivalent to the conditional distribution for each update in the Gibbs sampler. In this situation, we may find a Gibbs sampler is more efficient to compute due to the removal of more expensive matrix computations.

Blocking methods are known to perform well in the literature (Roberts and Sahu, 1997; Rue, 2001), including applications to MCMC for spatial disease mapping (Knorr-Held and Rue, 2002). More recent developments by Chib and Ramamurthy (2010) allow block updating on irregular densities by tailoring the proposal to the curvature of the posterior at that location, while also randomising the size and content of each block.

4.6 Data augmentation

In the case of a linear Gaussian model, the target distribution that we wish to sample from is $\pi(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y})$. Samples from this target can be obtained using the method of data augmentation described by Tanner and Wong (1987) where samples are drawn from $\pi(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})$ and $\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})$ in turn.

In many cases, including the case studies to be explored in Chapters 5 and 6, there will be no conditional dependence between any of the parameters, $\boldsymbol{\theta}$, meaning that samples from $\pi(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})$ can often be simulated one-at-a-time without issues of poor mixing between themselves. The latent variables, $\boldsymbol{x}$, are likely to be highly correlated with each other so samples from $\pi(\boldsymbol{x} \mid \boldsymbol{\theta})$ should be sampled as a multivariate Gaussian block using the sparse matrix algorithm described in Section 2.3.5 which is implemented in `GDAGsim` (Wilkinson, 2002).

Generating a sample of values for the j-th iteration of the scheme is done in two stages. Firstly, each parameter is simulated in-turn from its full conditional distribution:

$$\theta_k^{(j)} \mid \theta_{\setminus k}^{(j-1)}, \boldsymbol{x}^{(j-1)}, \boldsymbol{y},$$

but since θ_k is unlikely to be dependent on other parameters this will often simplify to,

$$\theta_k^{(j)} \mid \boldsymbol{x}^{(j-1)}, \boldsymbol{y}.$$

A `GDAGsim` model is created using $\boldsymbol{\theta}^{(j)}$ and conditioned on the observations before a sample of the latent variables is generated as a block (Wilkinson and Yeung, 2002):

$$\boldsymbol{x}^{(j)} \mid \boldsymbol{\theta}^{(j)}, \boldsymbol{y}.$$

4.7 Marginal updating scheme

The target distribution we wish to sample from can be factorised as,

$$\pi(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\, \pi(\boldsymbol{x} \mid \boldsymbol{\theta})\, \pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}). \tag{4.2}$$

We might only be interested in performing inference for the parameters $\boldsymbol{\theta}$,

with little or no interest in obtaining simulations from the latent field $\boldsymbol{x}$. If this is the case, we will be interested in simulating from $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$. In order to target this posterior distribution, we would simulate proposal values of $\boldsymbol{\theta}^*$ from a proposal distribution $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$, and then accept $\boldsymbol{\theta}^*$ with acceptance probability $\alpha(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) = \min\{1, A\}$, where A is given by,

$$A = \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{y})}{\pi(\boldsymbol{\theta} \mid \boldsymbol{y})} \frac{q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})}. \tag{4.3}$$

The success of such a method depends on being able to evaluate $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$, which will not be possible for the majority of models. Instead, we may seek an approximation to the this density, $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$. Use of an approximation means we are now targeting an approximation of the posterior distribution as opposed to the exact posterior distribution. If we combine this approximation with a symmetric proposal distribution $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$, the value of A in Equation (4.3) simply becomes

$$A = \frac{\tilde{\pi}(\boldsymbol{\theta}^* \mid \boldsymbol{y})}{\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})},$$

where the approximation $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ aims to integrate out all dependence on $\boldsymbol{x}$:

$$\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y}) = \left. \frac{\pi(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y})}{\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})} \right|_{\boldsymbol{x}=\boldsymbol{x}^\star(\boldsymbol{\theta})}$$

and the numerator can be factorised as per Equation (4.2):

$$\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \left. \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{x} \mid \boldsymbol{\theta})\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})}{\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})} \right|_{\boldsymbol{x}=\boldsymbol{x}^\star(\boldsymbol{\theta})} \tag{4.4}$$

where $\boldsymbol{x} = \boldsymbol{x}^*(\boldsymbol{\theta})$ is the modal configuration of the latent field. As the LHS of Equation (4.4) does not depend on $\boldsymbol{x}$, it follows that the RHS does not either, so this can be evaluated at any $\boldsymbol{x}$ desired. Rue et al. (2009) recommends evaluating the density at the modal configuration $\boldsymbol{x}^*(\boldsymbol{\theta})$, which is obtained Gaussian approximation is performed at $\boldsymbol{\theta}$, for greatest accuracy in the approximation. A detailed version of this method is given in Algorithm 3.

While this method does not perform inference on the latent variables, Liu (1994) describes a method where these values could be obtained later using a collapsed Gibbs sampler. Once the chain for $\boldsymbol{\theta} \mid \boldsymbol{y}$ has converged, we can use the

Algorithm 3: A marginal update scheme for drawing samples from the univariate density, $\tilde{\pi}(\boldsymbol{\theta})$.

Initialise $\boldsymbol{\theta}^{(0)}$;
for $i := 1$ **to** N **do**
  Generate proposal: $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)})$;
  Perform Gaussian approximation to find $\boldsymbol{x}^\star(\boldsymbol{\theta}^*)$;
  Compute acceptance probability:

$$\alpha\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)}\right) = \min\left\{1, \frac{\tilde{\pi}(\boldsymbol{\theta}^*)}{\tilde{\pi}\left(\boldsymbol{\theta}^{(i-1)}\right)} \frac{q\left(\boldsymbol{\theta}^{(i-1)} \mid \boldsymbol{\theta}^*\right)}{q\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)}\right)}\right\}$$

  where

$$\tilde{\pi}\left(\boldsymbol{\theta}^{(i-1)}\right) = \left.\frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{x} \mid \boldsymbol{\theta})\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})}{\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})}\right|_{\boldsymbol{x}=\boldsymbol{x}^\star(\boldsymbol{\theta})};$$

  Sample $u \sim \mathrm{U}(0,1)$;
  if $u < \alpha(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)})$ **then**
    $\boldsymbol{\theta}^{(i)} := \boldsymbol{\theta}^*$;
  else
    $\boldsymbol{\theta}^{(i)} := \boldsymbol{\theta}^{(i-1)}$;
  end
end
Result: N samples from the target distribution, $\tilde{\pi}(\boldsymbol{\theta})$.

samples obtained to generate values of $\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y}$, either during the main monitoring run or as a separate run afterwards.

4.8 Two block sampler

Where the Marginal updating scheme in Section 4.7 targets an approximation of the posterior distribution, the two-block sampler can be used to target the *exact* posterior distribution. In addition, where the Marginal scheme defaults to only drawing samples of the parameters, $\boldsymbol{\theta}$, this two-block method also produces samples for the latent field, $\boldsymbol{x}$.

The aim of this method is to alternate between updating from $\pi(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})$ and $\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})$. Symmetric random walks can be used for each new update of $\boldsymbol{\theta}$. A Metropolis step is used to determine whether each proposed new value, $\boldsymbol{\theta}^*$ from proposal distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$ and $\boldsymbol{x}^*$ from proposal distribution $q_{\boldsymbol{x}}(\boldsymbol{x}^* \mid \boldsymbol{x})$,

should be accepted. The acceptance probabilities for each simulation will be:

1. For $\pi(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})$, the acceptance probability would be,

$$\alpha_{\boldsymbol{\theta}}\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}\right) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{x}, \boldsymbol{y})}{\pi(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})} \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{q_{\boldsymbol{\theta}}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})}\right\}.$$

2. For $\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})$, the acceptance probability would be,

$$\alpha_{\boldsymbol{x}}\left(\boldsymbol{x}^* \mid \boldsymbol{x}\right) = \min\left\{1, \frac{\pi(\boldsymbol{x}^* \mid \boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})} \frac{q_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{x}^*)}{q_{\boldsymbol{x}}(\boldsymbol{x}^* \mid \boldsymbol{x})}\right\}.$$

A suitable proposal distribution, $q_{\boldsymbol{x}}$, for $\boldsymbol{x}$ can be the optimised Gaussian approximation of the latent field, $\mathrm{N}\big(\boldsymbol{x}^\star, \boldsymbol{Q}^\star(\boldsymbol{\theta})\big)$.

4.9 Single block sampler

As with the two-block sampler in Section 4.8, this method for the single-block sampler can also produce simulations from the *exact* posterior distributions for parameters, $\boldsymbol{\theta}$, and the latent field, $\boldsymbol{x}$. This targets the same posterior distribution as is given in Equation (4.2).

Rather than alternate between sampling new values of $\boldsymbol{\theta}^*$ and $\boldsymbol{x}^*$ using the method in Section 4.8, a new set of values $(\boldsymbol{\theta}^*, \boldsymbol{x}^*)$ can be created in two steps: A new $\boldsymbol{\theta}^*$ is proposed from a distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$, then a new $\boldsymbol{x}^*$ is proposed from a distribution $q_{\boldsymbol{x}}(\boldsymbol{x}^* \mid \boldsymbol{\theta}^*)$ conditional on the proposed $\boldsymbol{\theta}^*$. The proposed $(\boldsymbol{\theta}^*, \boldsymbol{x}^*)$ are then either jointly accepted or rejected using a Metropolis step, with acceptance probability $\alpha\big((\boldsymbol{\theta}^*, \boldsymbol{x}^*) \mid (\boldsymbol{\theta}, \boldsymbol{x})\big) = \min\{1, A\}$ where

$$A = \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \frac{\pi(\boldsymbol{x}^* \mid \boldsymbol{\theta}^*)}{\pi(\boldsymbol{x} \mid \boldsymbol{\theta})} \frac{\pi(\boldsymbol{y} \mid \boldsymbol{x}^*, \boldsymbol{\theta}^*)}{\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})} \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{q_{\boldsymbol{\theta}}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})} \frac{q_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{\theta})}{q_{\boldsymbol{x}}(\boldsymbol{x}^* \mid \boldsymbol{\theta}^*)}. \tag{4.5}$$

The first three terms of Equation (4.5) are given by Equation (4.2). A possible proposal for $q_{\boldsymbol{\theta}}$ is a symmetric random walk, meaning A would be simplified since $q_{\boldsymbol{\theta}}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*) = q_{\boldsymbol{\theta}}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$. Meanwhile, the proposal distribution for $q_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{\theta}^*)$ comes from the Gaussian approximation meaning $q_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{\theta}^*) = \tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}^*, \boldsymbol{y})$.

Hence A from Equation (4.5) would now simplify to,

$$A = \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \frac{\pi(\boldsymbol{x}^* \mid \boldsymbol{\theta}^*)}{\pi(\boldsymbol{x} \mid \boldsymbol{\theta})} \frac{\pi(\boldsymbol{y} \mid \boldsymbol{x}^*, \boldsymbol{\theta}^*)}{\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})} \frac{\tilde{\pi}_G(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})}{\tilde{\pi}_G(\boldsymbol{x}^* \mid \boldsymbol{\theta}^*, \boldsymbol{y})}. \tag{4.6}$$

This scheme proves to be useful when there is a strong dependence between the parameters, $\boldsymbol{\theta}$, and the latent variables, $\boldsymbol{x}$. This is in contrast to the two-block sampler where the mixing of a scheme can be affected if there is a strong correlation between the $\boldsymbol{\theta}$ and $\boldsymbol{x}$.

4.10 Blocking methods for Bayesian variable selection models

4.10.1 Including indicators for inference

We can expand the blocking methods shown in this Chapter to allow for Bayesian variable selection to be included in these models. We demonstrate this by supposing there are binary indicators, $\boldsymbol{I}$, such as those suggested by Kuo and Mallick (1998) which act upon some latent variables. Developing on Equation (4.2), the new target distribution would be,

$$\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I} \mid \boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\, \pi(\boldsymbol{I} \mid \boldsymbol{\theta})\, \pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{I})\, \pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I}). \tag{4.7}$$

The full conditional distribution for I can be calculated in these models by normalising the probability mass of being in each of the two possible states. In this section, we explore how existing block-MCMC algorithms are adjusted to account for this.

4.10.2 Data augmentation

Use of data augmentation already requires the ability to simulate from the full conditional distributions for parameters and latent variables. As the full conditional distribution for the indicator variables are also known, we simply include this as an extra step in the existing method. Like the data augmentation scheme seen in Section 4.6, this produces samples from the exact posterior distribution.

The method for generating samples for the j-th iteration of the scheme is now done in three stages:

1. Sample each parameter in turn,

$$\theta_k^{(j)} \mid \theta_{\setminus k}^{(j-1)}, \boldsymbol{x}^{(j-1)}, \boldsymbol{I}^{(j-1)}, \boldsymbol{y}.$$

2. Sample each indicator,

$$\boldsymbol{I}^{(j)} \mid \boldsymbol{\theta}^{(j)}, \boldsymbol{x}^{(j-1)}, \boldsymbol{y}.$$

   Since the indicators are conditionally independent from each other, these can be sampled using a standard one-at-a-time Gibbs sampler.

3. Sample the latent variables as a block using the latest parameters and indicators,

$$\boldsymbol{x}^{(j)} \mid \boldsymbol{\theta}^{(j)}, \boldsymbol{I}^{(j)}, \boldsymbol{y}.$$

This process is repeated for the desired number of iterations.

4.10.3 Augmented block in data augmentation

Data augmentation methods can be used where all the full conditional distributions are tractable, such as in linear Gaussian models. Where latent Gaussian models are used, we may often find some of these full conditional distributions to be intractable. To work around this issue, a hybrid of the data augmentation and single block schemes are used, where the indicators are sampled directly from their full conditional distribution before the parameters and latent variables are sampled using a single-block-style method; this is detailed in Algorithm 4.

This method can be modified to accommodate a two-block (Section 4.8) scheme to generate samples for the parameters and latent values. An option is to expand further by splitting latent variables to be sampled in multiple distinct blocks is possible, but this requires the model to be reconstructed for each accepted block, leading to multiple model constructions per iteration.

4.10.4 Marginal schemes

As marginal schemes do not provide inference for the latent variables, inference cannot be performed for the indicators. Attempts to use a collapsed sampler (Liu,

Algorithm 4: Augmented block in data augmentation scheme drawing samples from $\pi(\boldsymbol{\theta}, \boldsymbol{I}, \boldsymbol{x} \mid \boldsymbol{y})$.

Initialise $\left(\boldsymbol{\theta}^{(0)}, \boldsymbol{I}^{(0)}, \boldsymbol{x}^{(0)}\right)$;
for $i := 1$ **to** N **do**
  Sample new indicators from: $\boldsymbol{I}^{(i)} \mid \boldsymbol{\theta}^{(i-1)}, \boldsymbol{x}^{(i-1)}, \boldsymbol{y}$;
  Generate proposal: $\boldsymbol{\theta}^* \sim q\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)}\right)$;
  Perform Gaussian approximation to find $\boldsymbol{x}^\star\left(\boldsymbol{\theta}^*, \boldsymbol{I}^{(i)}\right)$ and $\boldsymbol{Q}^\star\left(\boldsymbol{\theta}^*, \boldsymbol{I}^{(i)}\right)$;
  Generate proposal: $\boldsymbol{x}^* \sim \mathrm{N}(\boldsymbol{x}^\star, \boldsymbol{Q}^\star)$;
  Compute acceptance probability $\alpha\left(\boldsymbol{\theta}^*, \boldsymbol{x}^*\right) \mid (\boldsymbol{\theta}, \boldsymbol{x})\big) = \min\{1, A\}$, where:

$$A = \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{I}^{(i)})}{\pi(\boldsymbol{\theta}^{(i-1)} \mid \boldsymbol{I}^{(i)})} \frac{\pi(\boldsymbol{x}^* \mid \boldsymbol{\theta}^*, \boldsymbol{I}^{(i)})}{\pi(\boldsymbol{x}^{(i-1)} \mid \boldsymbol{\theta}^{(i-1)}, \boldsymbol{I}^{(i)})} \frac{\pi(\boldsymbol{y} \mid \boldsymbol{x}^*, \boldsymbol{\theta}^*, \boldsymbol{I}^{(i)})}{\pi(\boldsymbol{y} \mid \boldsymbol{x}^{(i-1)}, \boldsymbol{\theta}^{(i-1)}, \boldsymbol{I}^{(i)})} \times \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{(i-1)} \mid \boldsymbol{\theta}^*)}{q_{\boldsymbol{\theta}}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)})} \frac{\tilde{\pi}_G(\boldsymbol{x}^{(i-1)} \mid \boldsymbol{\theta}^{(i-1)}, \boldsymbol{I}^{(i)}, \boldsymbol{y})}{\tilde{\pi}_G(\boldsymbol{x}^* \mid \boldsymbol{\theta}^*, \boldsymbol{I}^{(i)}, \boldsymbol{y})}; \tag{4.8}$$

  Sample $u \sim \mathrm{U}(0, 1)$;
  if $u < \alpha\left((\boldsymbol{\theta}^*, \boldsymbol{x}^*) \mid (\boldsymbol{\theta}, \boldsymbol{x})\right)$ **then**
    $\left(\boldsymbol{\theta}^{(i)}, \boldsymbol{x}^{(i)}\right) := \left(\boldsymbol{\theta}^*, \boldsymbol{x}^*\right)$;
  else
    $\left(\boldsymbol{\theta}^{(i)}, \boldsymbol{x}^{(i)}\right) := \left(\boldsymbol{\theta}^{(i-1)}, \boldsymbol{x}^{(i-1)}\right)$;
  end
end
Result: N samples from the target distribution, $\pi(\boldsymbol{\theta}, \boldsymbol{I}, \boldsymbol{x} \mid \boldsymbol{y})$.

1994) are unlikely to be successful since the indicators that would be generated are dependent on the current latent variables (which aren't initially inferred), and the current latent variables depend on the current indicators, creating a cyclic dependency that can't be resolved.

4.10.5 Gibbs variable selection

Gibbs variable selection (GVS), as described by Dellaportas et al. (2002), also provides a binary indicator to determine if an associated variable should be included in the model. Under GVS, the variable associated with a currently excluded indicator is sampled from a pseudo-prior rather than the originally specified prior distribution. This does not affect the posterior distribution of the variable because samples from the pseudo-prior are never included in the model if the indicator's

current value is zero.

Values for $\tilde{\mu}_\gamma$ and $\tilde{\tau}_\gamma$ are selected to approximately match the posterior distribution for γ. This is achieved by performing a short trial-run where all indicator variables are fixed to equal 1, before the main MCMC run is performed. The mean and variance of each γ_l variable is recorded from the trial-run to create a pseudo-prior in the MCMC run, being sampled from when the corresponding δ_l takes the value of zero. Allowing the pseudo-prior to create samples closer to the underlying posterior improves the mixing of the scheme by causing the indicators to switch states more frequently, without affecting the proportion of time that the indicator spends in each state.

Performing each iteration of this method works in the same way as the data augmentation method described in Section 4.10.2, with the additional step at the end of each iteration where variables excluded by their indicator are replaced by a new value simulated from their pseudo-prior. As we have seen with the data augmentation methods, this scheme will target the exact posterior distribution.

4.10.6 Dynamic resizing of GDAG models

All MCMC algorithms in this section involve using the `GDAGsim` software to create a precision matrix and mean vector for the prior distribution of the latent variables; in the case of linear Gaussian models, the software will also condition the model on the observations. The high dimension of the latent variables mean the matrices become large and expensive to work with. We can exploit the fact that many of these latent variables may be excluded from the model by the indicator variables, by not including these variables at all when constructing the precision matrix.

While matrix operations are expensive and we will gain performance benefits by reducing the matrix size, there are additional overheads to model resizing, mostly relating to ensuring that new samples of the latent variables are correctly indexed to account for the variables which are missed out. There can also be additional memory allocation overheads since the size of the model often changes, resulting in extra allocation and garbage collection steps which negate some of the computational efficiency benefits. As a result, implementing this technique will only prove beneficial when a relatively small proportion of the variables are likely to be included in the model at any given time.

No new values will be simulated for the latent variables which are excluded

from the construction and processing of the `GDAGsim` model. Values are created for these variables by creating independent samples from the prior distribution, generated separately from the main `GDAGsim` model. Model resizing is especially useful on schemes using Gibbs Variable Selection since excluded variables have to be simulated from their pseudo-prior anyway.

4.11 Evaluation of method performance

Thinning can be used to decrease autocorrelation between samples in an MCMC chain by creating a subsample of every k-th value where a thinning step size of k is used. Use of thinning has drawbacks as it essentially discards useful information from the chain. As a result, Geyer (1992) and Link and Eaton (2012) argue that all thinning is bad—except where memory is limited for storing or analysing the output chains—since a chain which has thinning applied will have greater error in estimating parameter means compared to its unthinned equivalent. MacEachern and Berliner (1994) take this view further by attempting to justify a ban on thinning for the same reasons. All authors highlight that estimating variance of a parameter can be affected by an autocorrelated chain, but suggest this can be estimated on a subsampled chain or preferably using suitable time-series-based methods.

As a measure of efficiency, we are interested in the amount of time it takes to achieve a suitably thinned run with each scheme under investigation. A key measurement for efficiency is obtained by determining the *effective sample size* (ESS) generated per second of CPU[1] user time (ESS/s). For a sample generated over N iterations, the ESS is calculated as,

$$\text{ESS} = \frac{N}{1 + 2\sum_{k=1}^{\infty} \rho_k(\theta_i)}, \tag{4.9}$$

where $\rho_k(\theta_i)$ is the autocorrelation at lag k for parameter θ_i (Gong and Flegal, 2016; Ripley, 1987). This indicates the number of independent iterations the chain would represent once any autocorrelation between the samples has been accounted for. Kass et al. (1998) explain that in practice, the summation should only be performed to an appropriate value of k as including autocorrelation estimates for higher lags produces a noisy estimate of ESS, so the summation in

[1]Simulations performed on a 3.40GHz Intel® Core™ i7-3770 CPU with 8GB RAM.

Equation (4.9) may typically only be computed while $\rho_k(\theta_i) > 0.05$. In cases where there is no (or negligible) autocorrelation between samples, ESS $= N$.

We consider a MCMC scheme to be as efficient overall as its least efficient part; we concern ourselves with the minimum ESS/s value for any variable inferred as part of the scheme as this is related to the geometric rate of convergence of the MCMC chain. Some exceptions will apply to which parameters are included as part of the minimum ESS except for indicator variables which will have an ESS of zero if they stay in the same state for the entire chain.

Software implementations for computing ESS are freely available in the `coda` (Plummer et al., 2006) and `mcmcse` (Flegal et al., 2016) packages for R; throughout this thesis, we use the `coda` implementation to compute ESS values.

4.12 Application to the traffic 'near-miss' model

4.12.1 Constructing the necessary factors

We revisit the model for near-misses at Place Charles de Gaulle (introduced in Section 3.3.1) to investigate some MCMC methods that can be applied to this example.

We are interested in performing inference for the posterior of interest given in Equation (4.2). We will look at each of the factorised terms in turn:

- We have $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ which represent the hyperparameters, where θ_1 is a log-Gamma distribution and θ_2, θ_3 are Normally distributed. This gives $\pi(\boldsymbol{\theta})$ the following distribution:

$$\begin{aligned}\pi(\boldsymbol{\theta}) &= \pi(\theta_1)\pi(\theta_2)\pi(\theta_3) \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)} \exp\left\{\alpha\theta_1 - \beta e^{\theta_1}\right\} \times \frac{1}{\sigma_\phi\sqrt{2\pi}} \exp\left\{-\frac{(\theta_2 - \mu_\phi)^2}{2\sigma_\phi^2}\right\} \\ &\quad \times \frac{1}{\sigma_\mu\sqrt{2\pi}} \exp\left\{-\frac{(\theta_3 - \mu_\mu)^2}{2\sigma_\mu^2}\right\}. \end{aligned} \tag{4.10}$$

- $\pi(\boldsymbol{x} \mid \boldsymbol{\theta})$ is a multivariate Gaussian density of the latent variables, with precision matrix, $\boldsymbol{Q}(\boldsymbol{\theta})$, constructed from the current parameters, $\boldsymbol{\theta}$. This has a

density of,

$$\pi(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{|\boldsymbol{Q}(\boldsymbol{\theta})|^{1/2}}{\sqrt{2\pi}^{d}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{T}\boldsymbol{Q}(\boldsymbol{\theta})(\boldsymbol{x}-\boldsymbol{\mu})\right\}. \tag{4.11}$$

- The likelihood of $\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$ in the case where observations, y_i, follow a Poisson distribution with rate parameter, $E_i \exp(x_i)$, is,

$$\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \prod_{i \in \mathcal{I}} \frac{(E_i e^{x_i})^{y_i} \exp\left(-E_i e^{x_i}\right)}{y_i!}. \tag{4.12}$$

By combining the terms from Equations (4.10), (4.11) and (4.12), we can write the target posterior distribution as,

$$\begin{aligned}
\pi(\boldsymbol{\theta}, \boldsymbol{x} \mid \boldsymbol{y}) &\propto \pi(\boldsymbol{\theta})\pi(\boldsymbol{x} \mid \boldsymbol{\theta})\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) \\
&\propto \pi(\boldsymbol{\theta})\pi(\boldsymbol{x} \mid \boldsymbol{\theta}) \prod_{i} \pi(y_i \mid x_i, \boldsymbol{\theta}) \\
&\propto \frac{\beta^{\alpha}}{\Gamma(\alpha)} \exp\left\{\alpha\theta_1 - \beta e^{\theta_1}\right\} \frac{1}{\sigma_\phi\sqrt{2\pi}} \exp\left\{-\frac{(\theta_2 - \mu_\phi)^2}{2\sigma_\phi^2}\right\} \\
&\quad \times \frac{1}{\sigma_\mu\sqrt{2\pi}} \exp\left\{-\frac{(\theta_3 - \mu_\mu)^2}{2\sigma_\mu^2}\right\} \\
&\quad \times \frac{|\boldsymbol{Q}(\boldsymbol{\theta})|^{1/2}}{\sqrt{2\pi}^{d}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{T}\boldsymbol{Q}(\boldsymbol{\theta})(\boldsymbol{x}-\boldsymbol{\mu})\right\} \\
&\quad \times \prod_{i \in \mathcal{I}} \left[\frac{(E_i e^{x_i})^{y_i} \exp\left(-E_i e^{x_i}\right)}{y_i!}\right].
\end{aligned} \tag{4.13}$$

This forms the basis of finding acceptance probabilities for a number of MCMC methods discussed in this chapter. Some methods, such as the data augmentation method in Section 4.6 cannot be applied to this example since samples cannot be directly sampled from the full conditional distributions.

4.12.2 Marginal method

The marginal MCMC method was initially described in Section 4.7. In order to perform approximate inference, we need to be able to calculate the density of $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ as shown in Equation (4.4). The numerator of Equation (4.4) is given by

Equation (4.13), while its denominator is given by the density of the Gaussian approximation as given in Equation (3.24). The whole density is then evaluated at the modal value of $\boldsymbol{x} = \boldsymbol{x}^\star(\boldsymbol{\theta})$ for reasons discussed in Section 3.2.

With the density of $\tilde{\pi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ now known, we are then able to calculate the acceptance probability for each proposed value of $\boldsymbol{\theta}$ by following the method described in Section 4.7.

4.12.3 Double-block method

We perform the double-block method described in Section 4.8, but we note that we need to calculate two different acceptance ratio terms for this method. Taking relevant terms from Equation (4.13), we have ways to determine these acceptance ratio components:

- For $\pi(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})$,

$$\begin{aligned}\pi(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y}) \propto \frac{\beta^\alpha}{\Gamma(\alpha)} \exp\left\{\alpha\theta_1 - \beta e^{\theta_1}\right\} \frac{1}{\sigma_\phi\sqrt{2\pi}} \exp\left\{-\frac{(\theta_2 - \mu_\phi)^2}{2\sigma_\phi^2}\right\} \\ \times \frac{1}{\sigma_\mu\sqrt{2\pi}} \exp\left\{-\frac{(\theta_3 - \mu_\mu)^2}{2\sigma_\mu^2}\right\} \\ \times \frac{|\boldsymbol{Q}(\boldsymbol{\theta})|^{1/2}}{\sqrt{2\pi}^d} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{Q}(\boldsymbol{\theta})(\boldsymbol{x} - \boldsymbol{\mu})\right\}.\end{aligned}$$

- While for $\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})$,

$$\begin{aligned}\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y}) \propto \frac{|\boldsymbol{Q}(\boldsymbol{\theta})|^{1/2}}{\sqrt{2\pi}^d} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{Q}(\boldsymbol{\theta})(\boldsymbol{x} - \boldsymbol{\mu})\right\} \\ \times \prod_{i \in \mathcal{I}} \left[\frac{(E_i e^{x_i})^{y_i} \exp\left(-E_i e^{x_i}\right)}{y_i!}\right].\end{aligned}$$

With these terms known, it is simple to follow the algorithm presented in Section 4.8 to perform the double-block method.

4.12.4 Single-block method

For the single-block algorithm introduced in Section 4.9, we need to find the value of A specified in Equation (4.5). The first three terms are given by Equation (4.13),

while the final term is based on the density from a Gaussian approximation given in Equation (3.24). These are then evaluated at the current and proposed values of $\boldsymbol{\theta}$ and $\boldsymbol{x}$, following the algorithm given in Section 4.9.

4.12.5 Results

We initially implement these MCMC methods to a 21 observation subset of the same data used in Section 3.3.7 The parameter values from which the data was generated remain unchanged: $(\theta_1 = -1.66, \theta_2 = 2.94, \theta_3 = 2)$. Non-informative priors were selected to match the vague defaults used by Rue et al. (2009) in their `INLA` software. These priors are,

$$\begin{aligned} \theta_1 &\sim \text{LogGamma}(1, 0.00005) \\ \theta_2, \theta_3 &\sim \text{N}(0, 0.001^{-1}). \end{aligned}$$

As parameters θ_1 and θ_2 are correlated, we select the proposal distribution for parameters, $q(\boldsymbol{\theta}^\star \mid \boldsymbol{\theta})$, to be a zero-mean multivariate Gaussian random walk. The variance matrix for the proposal distribution, $\boldsymbol{\Sigma}_q$, is based on a multivariate extension of the Roberts et al. (1997) suggestion for proposals:

$$\boldsymbol{\Sigma}_q = \frac{2.38^2 \times \boldsymbol{\Sigma}_{\pi(\boldsymbol{\theta})}}{d},$$

where $\Sigma_{\pi(\boldsymbol{\theta})}$ is the d-dimensional variance matrix of the target distribution of $\pi(\boldsymbol{\theta})$ which could be estimated by a short trial run of the MCMC scheme.

For this small model, we find that standard Gibbs schemes are able to compete with the marginal and blocking methods, largely thanks to the low-dimension restricting the correlation between latent variables.

Trace plots in Figure 4.1 demonstrate that the single and two-block methods are the slowest to explore the target density compared to the approximate marginal and `JAGS` methods. A major reason for this is that most proposed moves will be rejected as part of the MH-step. The Marginal scheme also suffers this issue but to a lesser extent since latent variables are not part of the proposal, leading to a smaller dimension proposal with higher acceptance probability.

Autocorrelation plots given in Figure 4.2 reinforce traits identified in the trace plots. `JAGS` and Marginal schemes have the lowest ACF values as lag increases

thanks to their ability to explore the posterior density quicker than the Single and Two block methods.

Kernel density estimates of the posterior distributions, generated from 30000 well-thinned realisations, are shown in Figure 4.3 to demonstrate how all schemes are able to equivalently target the same posterior density, with the "true" parameter values well-represented in the posterior density for each of the model parameter. Trace and ACF plots for the chains used to create these densities are provided as a supplement in Appendix C.1. For the model shown here, the Marginal scheme, which only produces *approximate* inference, is able to closely match the other schemes which are designed to target the exact posterior distribution.

CPU user time and Effective Sample Size measurements for running these long schemes are provided in Table 4.1. The marginal method proves to be quickest in producing a long and suitably thinned run with minimal autocorrelation between samples—while the `JAGS` scheme manages to produce the least overall autocorrelation, and therefore the highest Effective Sample Size, the Marginal scheme produces the highest ESS/s due to its faster completion time. Despite the apparent speed of the Marginal schemes, it is worth remembering that this scheme only provides *approximate* inference for the posterior distribution and does not provide inference for the latent values by default—latent values can be inferred later as part of a collapsed Gibbs sampler (Liu, 1994).

As the dimension of the latent structure increases, the single and two block schemes begin to struggle to perform inference for the latent values. Attempts to perform inference on the full 1000 observations used in Section 3.3.7 typically causes chains for the latent values to become very slow at mixing. The high dimension of the block being sampled produces a very low acceptance probability. In the case of the single-block sampler, this is enough to cripple the entire scheme as movement of the parameters will be held back by poorly mixing latent values. A way to mitigate this issue involves dividing the latent variables into a greater number of smaller blocks, so each block has a higher acceptance probability.

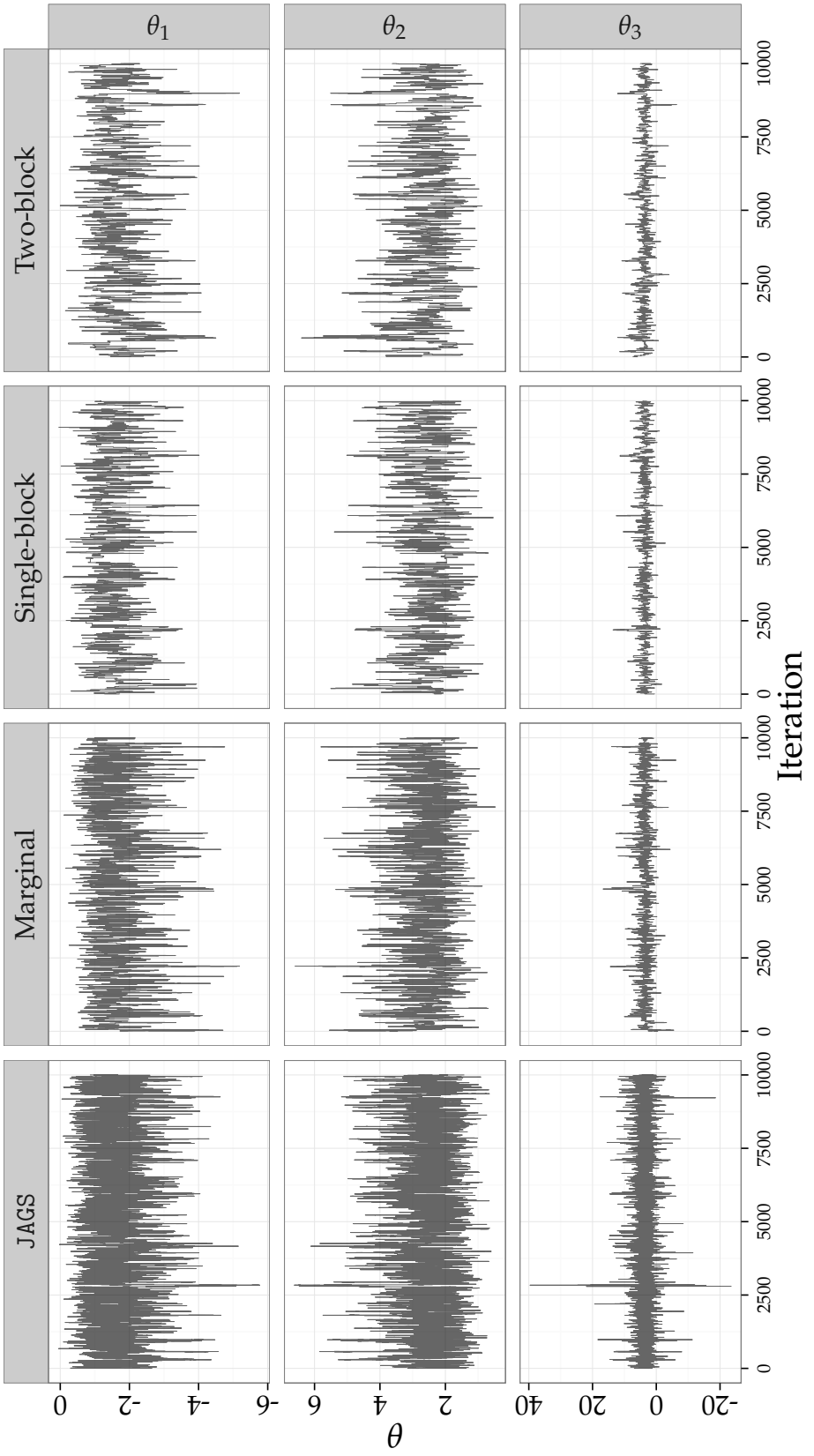


Figure 4.1: Trace plots over 10000 unthinned iterations for the parameters (from top row to bottom row) θ_1, θ_2 and θ_3, for each of the MCMC schemes used (from left column to right column): JAGS, Marginal, Single Block and Two block methods.

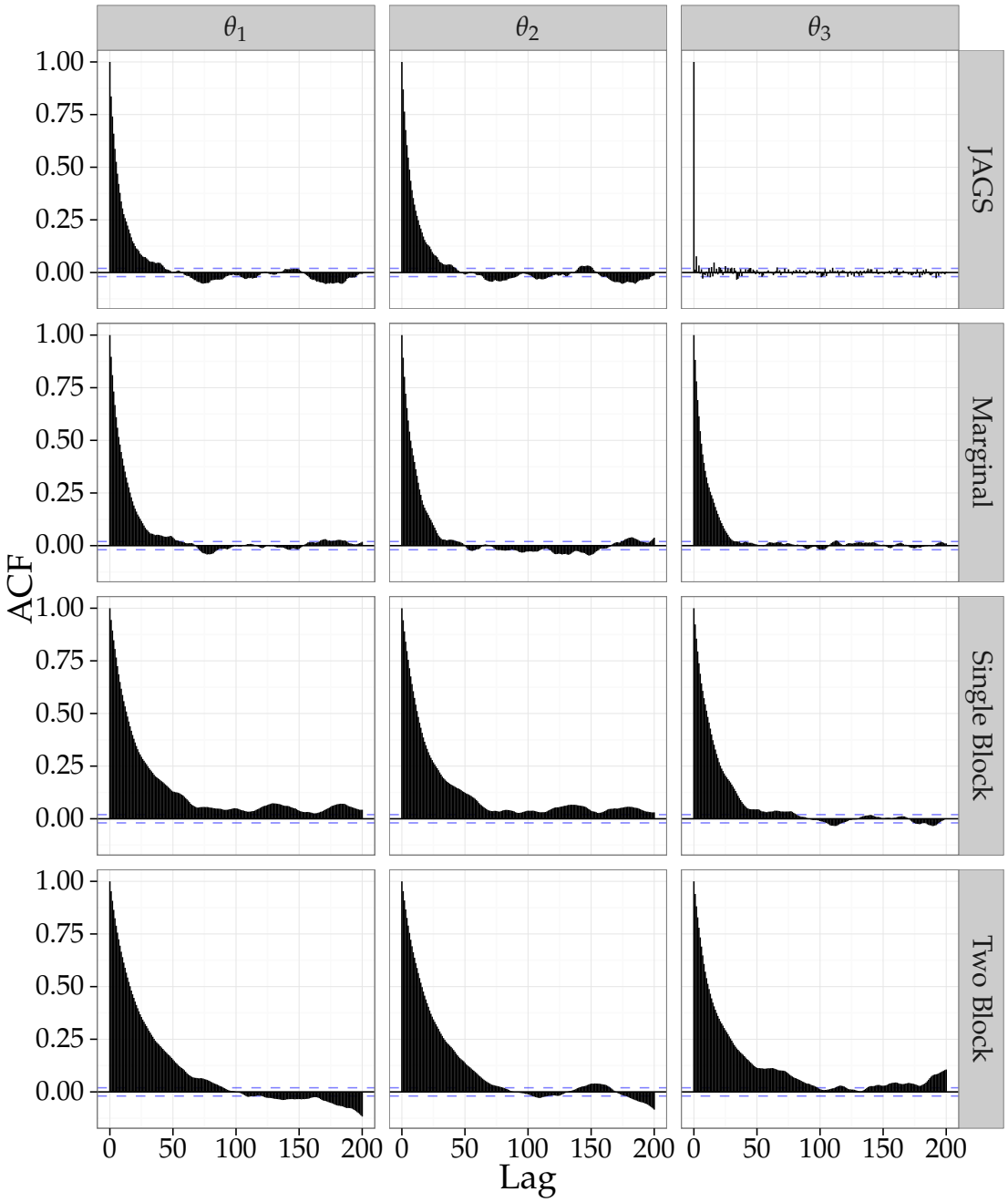


Figure 4.2: Autocorrelation plots with corresponding 95% intervals for the 10000 unthinned iterations shown in Figure 4.1, for the parameters (from left column to right column) θ_1, θ_2 and θ_3, for each of the MCMC schemes used (from top row to bottom row): `JAGS`, Marginal, Single Block and Two block methods.

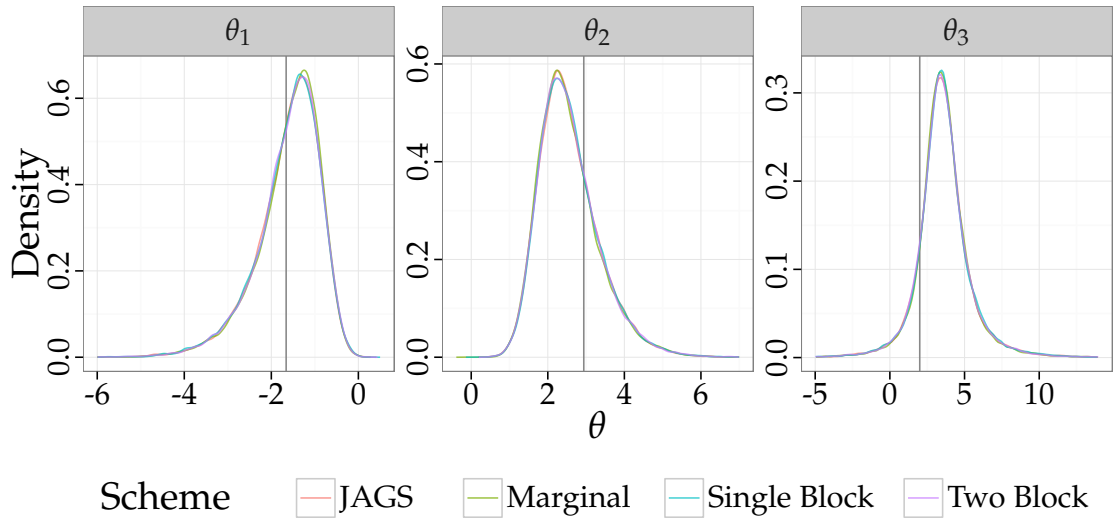


Figure 4.3: Kernel density plots of the posterior parameter densities (from left to right) θ_1, θ_2 and θ_3, provided by 30000 iterations (thinned by steps of 50) from various MCMC schemes. Vertical grey bars represent the "true" parameter values.

Implementation	Time (s)	ESS	Var.	ESS/s.
JAGS	105	29140	θ_1	278
		29231	θ_2	278
		30000	θ_3	286
Marginal	77	27708	θ_1	360
		28197	θ_2	366
		26583	θ_3	345
Single Block	106	25735	θ_1	243
		27267	θ_2	257
		23519	θ_3	222
Two Block	109	23192	θ_1	213
		24849	θ_2	228
		21637	θ_3	199

Table 4.1: CPU User time taken to obtain 30000 iterations (50 thinning steps between iteration) of $\boldsymbol{\theta}$ in the small dimension model for the example AR(1) model.

Part II

Case studies in genetic interactions

Chapter 5

Quantitative Fitness Analysis

5.1 A brief introduction to genetics

5.1.1 DNA

The molecular structure of *deoxyribonucleic acid* (DNA) encodes the essential genetic processes that are required for cells to function, grow and reproduce. Consequently, DNA is considered essential for all living organisms and some viruses to survive.

Watson and Crick (1953) demonstrated that DNA comprises of two strands of nucleotides, each containing one of 4 nucleobases (or *bases* for short): adenine (A), cytosine (C), guanine (G) or thymine (T). The nucleotide in any location on one strand is always paired to its complementary nucleotide on the other strand, such that adenine (A) pairs to thymine (T), while cytosine (C) pairs with guanine (G).

To identify the direction of each strand, each end of the strand is denoted by 3′ and 5′ (pronounced "three-prime" and "five-prime")[1]. The strands run antiparallel to each other, such that the 5′ end of one strand is at the same end as the 3′ end of the other strand. It is important to note the sequence of bases depends on which way round they are read, so denoting the ends can be necessary to remove this potential ambiguity. The sequence 5′ GAT 3′ would be equivalent to 3′ TAG 5′ when the same strand is view from the other side. When the 3′ and 5′ ends are not denoted, the convention is assume bases are being listed in the

[1] The 3′ refers to the end which terminates at the hydroxyl group and the 5′ refers to the end terminating at the phosphate group. The numbers 3 and 5 refer to the 3rd and 5th carbons (respectively) in the deoxyribose sugar-ring where the hydroxyl and phosphate groups are found.

$5' \rightarrow 3'$ direction. Since bases appear as complements, only the sequence of one strand needs to be recorded to know the overall sequence of both strands. An example sequence is given in Figure 5.1.

```
5′ ...GTTCGACTCAAT... 3′
3′ ...CAAGCTGAGTTA... 5′
```

Figure 5.1: Example sequence of a DNA double-strand. Nucleotides in the top strand are complemented by nucleotides in the bottom strand, so that `C` complements `G` and `A` complements `T`.

To remove ambiguity in how the nucleobases are recorded, Cartwright and Graur (2011) propose that all DNA sequences should follow the conventions used in the *Saccharomyces Genome Database* (SGD) (Cherry et al., 2012). A centromere is used to divide the chromosome into two unequal halves. The Watson strand is then defined to have its $5'$-end of the shorter half of the chromosome, while the Crick strand has its $5'$-end on the longer half. The sequence is then recorded in the $5' \rightarrow 3'$ direction on the Watson strand.

Ribonucleic acid (RNA) is a molecule comprised from a chain of nucleotides, much like DNA but with two notable differences in its structure that we will consider. Firstly, uracil (`U`) appears as one of the four nucleobases in RNA as a replacement for thymine (`T`) which appears in DNA. Secondly, while DNA appears as two strands of nucleotides running antiparallel to each other, RNA appears as a single strand that folds back on itself.

5.1.2 Open reading frames

When reading the sequence of nucleotides in the $5' \rightarrow 3'$ direction, the nucleotides can be grouped into triplets. There are three possible reading frames for the same sequence depending on which nucleotide the reading frame begins on; Figure 5.2 demonstrates the three possible reading frames you can have for a given sequence. Each reading frame has the grouping of each triplet offset by one compared to the other possible reading frames.

Certain triplets of bases translate to amino acids, or signals to start or stop genetic translation—such triplets are known as *codons*. A triplet of `ATG` often serves as a *start codon* in DNA, while any triplets from `TAA`, `TAG` or `TGA` behave as

Reading frame 1: 5′ . . . GTTCGACTCAATGACGTG . . . 3′
Reading frame 2: 5′ . . . GTTCGACTCAATGACGTG . . . 3′
Reading frame 3: 5′ . . . GTTCGACTCAATGACGTG . . . 3′

Figure 5.2: Three possible reading frames for a given sequence of nucleotides in DNA. The triplets in each reading frame are denoted by alternating blue and orange colouring.

stop codons[2].

A sequence of DNA between a start codon and an end codon that could be translated into RNA is called an *open reading frame* (ORF) (Pagon et al., 2017). The sequence of one particular ORF in *Saccharomyces cerevisiae*, better known as *baker's yeast*, is listed in Figure 5.3.

```
ATG GTC AAA TTA ACT TCA ATC GCT GCC GGT GTC GCC GCC ATT GCT GCT
GGT GCC TCC GCC GCA GCA ACC ACT ACA TTA TCT CAA TCT GAC GAA AGA
GTT AAT TTG GTT GAA TTA GGT GTT TAT GTT TCC GAT ATC AGA GCT CAT
TTG GCT GAA TAC TAC TCT TTC TAA
```

Figure 5.3: The nucleotides of open reading frame YAR020C (known by the gene name PAU7) in *Saccharomyces cerevisiae* with a length of 168 base-pairs. The start codon is shown in blue while the end codon is shown in red. The first occurrence of a `TAA` sequence is highlighted in pink, but since this is not in the same reading frame, it does not act as a stop codon.

In this thesis, we will refer to ORF that appear in *Saccharomyces cerevisiae* by their ORF names or gene names as defined by SGD naming scheme. For the ORF featured in Figure 5.3, YAR020C, the name can be decoded as follows:

- 'Y' stands for 'yeast'.
- 'A' refers to which of the chromosomes this ORF is found on. There are 16 different chromosomes in *Saccharomyces cerevisiae*, labelled 'A'–'P'.
- 'R' shows that this gene is found to the right of the centromere. A gene to the left of the centromere would have 'L' instead.
- '020' indicates that this is the 20^{th} gene from the centromere.

[2] The equivalent start codon in RNA would be `AUG` and the stop codons would be `UAA`, `UAG` and `UGA`.

- 'C' denotes a gene that appears on the 'Crick' strand, while an ORF name ending in 'W' would appear on the 'Watson' strand.

Some ORF are given gene names (also known as genetic names), which relate to their known genetic function. Example gene names include CDC13 (cell division control protein 13, ORF name: YDL220C) and EXO1 (exonuclease protein 1, ORF name: YOR033C). In this thesis, we will refer to genes using their gene name if it has one, otherwise we will use the ORF name. We will also use fictional gene names, such as YFG1 (which stands for "your favourite gene") and XYZ2, as placeholders for other genes when describing some genetic processes.

5.1.3 Synthetic genetic array analysis

Synthetic genetic array analysis (SGA) is a method described by Tong and Boone (2006) where a gene can be targeted for deletion using cassettes that contain a barcode and an antibiotic resistance strain. In Figure 5.4, we see a cassette containing barcodes, which help identify the target location in the sequence to be replaced, along with NATMX which provides antibiotic resistance to nourseothricin. When these strains are allowed to culture for a while, a mixture of strains will develop where some have remained unmodified and still contain the original ORF, while other gene-deleted mutants contain the NATMX cassette. These strains are then transferred to a new agar plate containing nourseothricin and allowed to culture further. Any unmodified strains will die in the presence of nourseothricin, leaving only the strains which had the ORF removed in exchange for the nourseothricin resistance. The notation *yfg1*Δ is standard notation for denoting that the ORF with the gene name YFG1 has been deleted from the strain of interest.

Tong et al. (2001) explains how that this can be extended so that multiple genes can be removed from the same strain. To perform this, a second cassette is required which targets the second ORF to be deleted and provides a different antibiotic resistance. In the example illustrated in Figure 5.5, the KANMX antibiotic resistance displaces the second target gene and provides the strain resistance to kanamycin.

Allowing the strain to culture around the NATMX and KANMX markers produces a mixture of four different strains: unmodified strains that have both original genes to be deleted; strains that accepted NATMX only; strains that accepted KANMX only; strains that accepted both NATMX and KANMX and therefore had

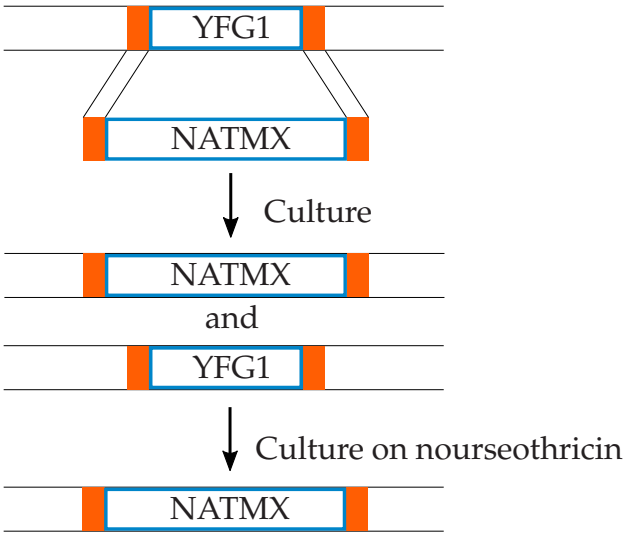


Figure 5.4: Gene displacement by a NATMX cassette, causing the ORF for gene YFG1 to be lost but gaining a resistance to nourseothricin. The orange blocks represent barcodes which identify the target location of the strain. This diagram is based on Figure 1a of Tong and Boone (2006).

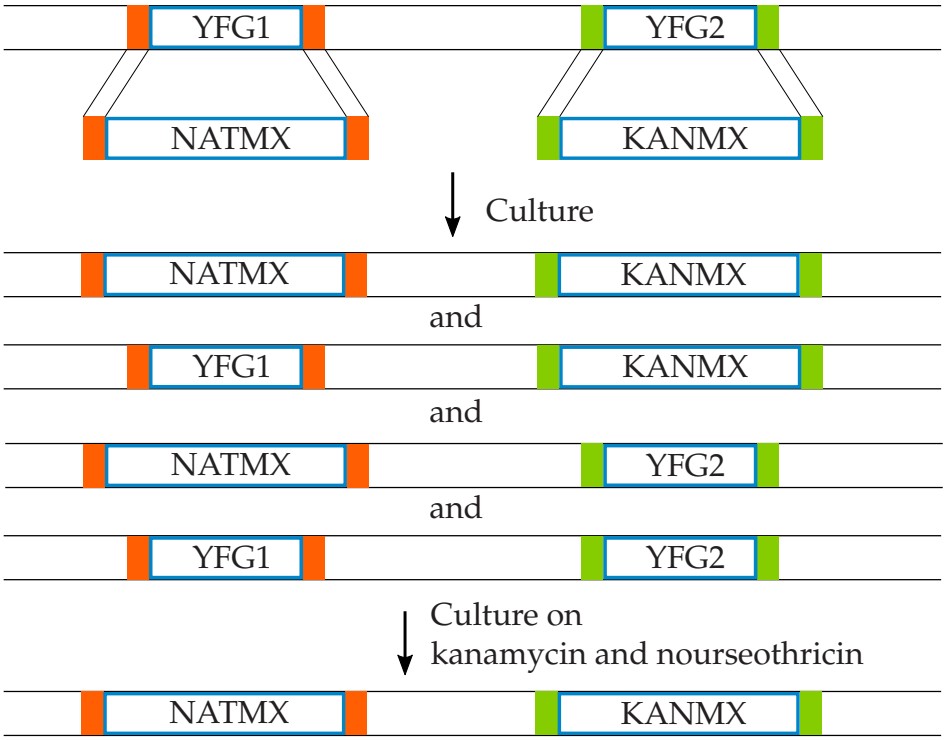


Figure 5.5: Displacement of two genes by providing kanamycin and nourseothricin antibiotic resistance cassettes to two different locations.

both of the targeted genes displaced. When this is cultured on plates containing both nourseothricin and kanamycin, the only strains that will survive are those which collected both the NATMX and KANMX markers.

An interesting situation arises when two cassettes are designed to target the same ORF of the same strain, which is illustrated by Figure 5.6. Only one of the cassettes is able to replace the targeted ORF; if one cassette has successfully displaced the ORF of interest, the second cassette would have to displace the first cassette in order to be included in the strain. After being allowed to grow under these conditions, three possible strains could develop, with none of them containing both NATMX and KANMX resistance markers needed to grow on a plate containing both kanamycin and nourseothricin. In this situation, there would be no surviving strains and all strains would be synthetically dead.

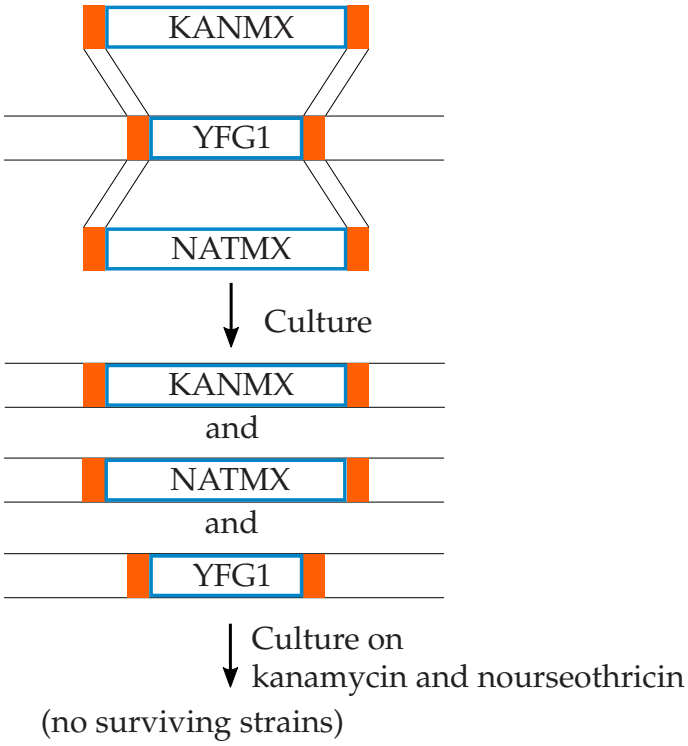


Figure 5.6: Attempting gene displacement of the same gene 'twice' from the same strain using two different antibiotic resistance markers.

It may not appear logical to attempt to delete the same ORF twice from the same strain, but it must be remembered that due to the large number of gene-deletion combinations to perform, these strains are often created by robots spotting multiple cultures simultaneously on to agar plates. Consequently, there will

be a few cultures where the same gene is targeted by multiple cassettes and don't gain all necessary antibiotic resistance markers needed to survive, but still need to be grown under the same conditions as other strains which have all the antibiotic resistance needed to survive.

5.1.4 Genetic epistasis

Genetic *epistasis* is the phenomenon where a gene will interact with one or more other genes. This often occurs when a genetic process relies on two genes to perform the same task or when one gene inhibits the process of another gene. Where there is no epistasis between two genes, the processes of one gene will not be affected by the presence or absence of the other gene; the two genes will work independently.

Epistasis can be explained using a known example which is originally mentioned by Vallen et al. (2000), where "deletions of BNI1 and BNR1 are synthetically lethal at 23 °C". This point is visually demonstrated by Buttery et al. (2012, Figure 3), where growth images of an unmodified wildtype strain is compared to those for three other strains, each containing different deletions: one *bnr1*Δ strain, one *bni1-1*Δ strain, one strain with both deletions, *bnr1*Δ *bni1-1*Δ. In Figure 3 of Buttery et al. (2012), the wildtype strain has the highest fitness, while the *bnr1*Δ and *bni1-1*Δ single-deletion strains nearly match the fitness levels of the wildtype. While we might expect the double-deletion strain *bnr1*Δ *bni1-1*Δ to grow at a fitness level reasonably close to the single-deletion strands, at higher temperatures it does not grow at all, indicating epistasis is present between the two genes.

Epistasis can be quantified under different models which are explored later in Section 5.2.2.

5.2 Introduction to QFA

5.2.1 Background

A *telomere* is a structure found at the end of a linear chromosome in the majority of eukaryotes. The telomere's structure prevents the end of the chromosomes being mistaken for a DNA double-strand break and protects the ends of chromosomes from deterioration. Problems or defects in the telomere caps cause them to be-

have like double-strand-breaks (DSB), which in turn prompts a damage response. Lydall (2009) summarises a number of genes and processes currently known to be involved in telomere maintenance and DSB response.

Research by Cawthon et al. (2003) suggests that in people aged 60 and over, shortened or defective telomeres are linked to an increased incidence of cancer and greater effects of ageing. This brings obvious interest into identifying the mechanisms of telomere defects and searching for ways to minimise their negative health effects.

Certain non-essential genes along the chromosome can cause an interaction with the strength of the telomere cap defect. We can investigate which non-essential genes have an interaction by deleting those open reading frames, denoted as *orf*Δ, and measuring whether the telomere capping defect has been suppressed or enhanced, indicating genetic epistasis.

Quantitative Fitness Analysis (QFA) as used by Addinall et al. (2011) provides a high-throughput methodology for measuring genetic interactions between various telomere defects and genetic deletions created using the SGA method described in Section 5.1.3. This can give better understanding of the role each gene plays. QFA has been demonstrated on *Saccharomyces cerevisiae*, which can be grown on defined media and for which the full genome sequence is known and freely available (Goffeau et al., 1996).

Data collection for QFA involves the inoculation of cell cultures on solid agar plates. Robotic equipment regularly removes each plate and to be photographed at a number of time intervals, before being returned to incubate further. Image analysis is performed on the photographs using Colonyzer (Lawless et al., 2010) to provide quantitative estimates of the cell densities, which can then be used to estimate the colony doubling rate and capacity parameters for logistic growth models. Based on these parameters, several measures of culture fitness can be calculated—such as Maximum Doubling Rate (MDR; doublings/day), Maximum Doubling Potential (MDP; doublings), numerical area under [growth] curve (nAUC), or the product of MDR and MDP (MDRMDP; doublings^2/day)—as detailed by Addinall et al. (2011).

5.2.2 Modelling genetic interactions

The objective is to detect a deviation in the expected fitness of a query strain (containing a telomere defect) compared to the fitness of a control strain (a wildtype strain) with the same *orf*Δ in both strains. For the contents of this chapter, we will use *ura3*Δ for the control strain, and *cdc13-1* mutants for the query strain. We search for these deviations assuming Fisher's multiplicative model of epistasis (Cordell, 2002).

Where the multiplicative model of epistasis holds, the fitness of a query strain crossed with *orf*Δ should be,

$$\text{fit}(\textit{cdc13-1 orf}\Delta) = \frac{\text{fit}(\textit{cdc13-1})}{\text{fit}(\textit{ura3}\Delta)} \times \text{fit}(\textit{ura3}\Delta\ \textit{orf}\Delta), \quad \forall\ \textit{orf}\Delta, \tag{5.1}$$

where *orf*Δ is a gene deletion that does not genetically interact with the query strain, *cdc13-1*. The ratio of fitness for the query strain to the control strain remains constant, so we expect a linear dependence between the fitness of the control and query strain where an *orf*Δ is also present. Any such deviation between the observed fitness and the expected fitness from Equation (5.1) implies a genetic interaction between the defective telomere cap and the *orf*Δ, while the size of this deviation is the genetic interaction strength.

Alternative models of epistasis can be used, such as additive, log and minimum, which are evaluated by Mani et al. (2008). When fitness is converted to a log-scale for use in the models of Section 5.3, the additive model on the log-scale will be equivalent to the multiplicative model (Cordell, 2002) that is applied to the MDRMDP fitness metric, as used in Addinall et al. (2011) and Heydari et al. (2016). The minimum model of epistasis suggests that the fitness of a non-interacting double-mutant will be the same as whichever corresponding single-mutant has the lower fitness, but this not suitable for detecting interactions where a mutation affects the whole pathway and is not considered optimal for identifying certain interactions (Mani et al., 2008).

5.2.3 Previous Bayesian modelling for QFA

Heydari et al. (2016) uses a Bayesian Joint Hierarchical model (JHM) to detect genetic interactions between the deletion of non-essential genes and defective telomere caps. Unfortunately, the Joint Hierarchical model (JHM) which performs

full inference, from inferring growth rates and maximal colony capacities through to detecting significant epistasis, can take about a month of CPU time to complete. The Interaction Hierarchical Model (IHM) focuses mainly on detecting epistasis, which Heydari quotes as taking approximately a day to perform inference for. Part of the relatively large time requirement to perform inference could be attributed to the use of Gibbs sampling to update a large number of parameters in the model, with a complex structure between the variables. This can lead to poor mixing which requires thinning to reduce autocorrelation in the samples.

Previous models for measuring such interaction have been created by Heydari et al. (2016), which followed on from prior frequentist analysis by Addinall et al. (2011).

5.3 Linear Gaussian models

5.3.1 Frequentist random effects model

The simplest model for measuring genetic interactions mentioned by Heydari et al. (2016) is the frequentist random effects model. To maintain the linear Gaussian structure, this model was simplified to assume that for observed fitnesses, $\boldsymbol{F}$, the log-fitness $\boldsymbol{f} = \log(\boldsymbol{F} + 1)$ followed a Gaussian distribution with mean parameter equal to a linear combination of latent variables. Furthermore, latent variables were given Gaussian distributions whereas the JHM and IHM by Heydari et al. (2016) used t-distributions for some of these variables—this potentially makes our inference less robust if heavy-tailed distributions are necessary, but this compromise in accuracy enables improvements to the scheme's efficiency to be explored. The random effects model is specified as,

$$
\begin{aligned}
f_{clm} &= \mu_c + Z_l + \gamma_{cl} + \varepsilon_{clm} \\
\mu_c &= \begin{cases} \mu + \alpha & \text{if } c = 0 \\ \mu & \text{if } c = 1 \end{cases} \qquad \gamma_{cl} = \begin{cases} 0 & \text{if } c = 0 \\ \gamma_l & \text{if } c = 1 \end{cases} \\
Z_l &\sim \mathrm{N}(0, \sigma_Z^2) \qquad \varepsilon_{clm} \sim \mathrm{N}(0, \sigma^2)
\end{aligned}
\tag{5.2}
$$

where

- f is a measure of log-fitness,
- $c = 0$ for control strain and $c = 1$ for query strain,
- $l = 1, \ldots, L$, is an index for each *orf*Δ,
- m is an index for each replicate,
- γ_l is an estimate of genetic interaction strength for each *orf*Δ.

The fitness F for the model can be calculated by fitting a logistic growth model as described by Addinall et al. (2011), with fitness F being defined as the product of the maximal doubling rate and the maximal doubling potential of the colony. The log-fitness is then defined as $f = \log(F + 1)$ for the model.

The fitness of the query strain is represented in the model as μ, while $(\mu + \alpha)$ applies to the control strain. We can therefore consider α to be the expected increase in log-fitness for a control strain compared to its query strain. This corresponds to the inverse of the fit(*cdc13-1*)/fit(*ura3*Δ) ratio seen in Equation (5.1). The ORF fitness, a variation in the fitness caused by the deletion of a particular ORF, is represented by a random effect, $\mathbf{Z}$. Deviations from this linear model are accounted for by the genetic interaction strength term, $\boldsymbol{\gamma}$, once errors are accounted for.

Significant interactions are determined by testing each of the γ_l interactions for each *orf*Δ, and finding which of these have a false discovery rate (FDR) corrected p-value of less than 0.05.

Significant interactions with a positive γ_l interaction strength suggest that the query-strain fitness is better than expected compared to the control-strain fitness when the same gene is deleted from both strains, hence the telomere capping defect has been *suppressed* in the query-strain. Conversely, negative γ_l interaction strength suggest an *enhanced* defect.

5.3.2 Bayesian linear Gaussian model with no indicators

The random effects model of Section 5.3.1 provides a good basis for building a Bayesian model. We can easily account for other effects, such as plate effects, using a hierarchical structure. However, the large quantity of parameters in the model will require us to be more efficient, or inference may be time-consuming to perform. Using fitness on the log-scale, as was used in the random effects model in Section 5.3.1, and an identity link function on the linear predictor, gives

a linear Gaussian structure to this model. This provides the potential for blocking methods to be used which can improve mixing and overall efficiency.

Using this as a basis for a simple Bayesian model, the model can be specified as follows:

$$
\begin{aligned}
f_{clm} &= \mu + \alpha_c + Z_l + \gamma_{cl} + \varepsilon_{clm} \qquad (5.3)\\
\mu &\sim \mathrm{N}(0, 0.001^{-1}) & Z_l &\sim \mathrm{N}(0, \tau_z^{-1})\\
\alpha_c &\begin{cases} \sim \mathrm{N}(0, 0.001^{-1}) & \text{if } c = 0 \\ = 0 & \text{if } c = 1 \end{cases} & \gamma_{cl} &\begin{cases} = 0 & \text{if } c = 0 \\ \sim \mathrm{N}(0, \tau_\gamma^{-1}) & \text{if } c = 1 \end{cases}\\
\varepsilon_{clm} &\sim \mathrm{N}(0, \tau_\varepsilon^2) & \tau_\varepsilon, \tau_z, \tau_\gamma &\sim Ga(1, 0.00005)
\end{aligned}
$$

where terms of the model are equivalent to that in the random effects model of Section 5.3.1. Vague prior distributions are in use for this model, designed to replicate those used in the `INLA` software.

A DAG representation of this model is shown in Figure 5.7.

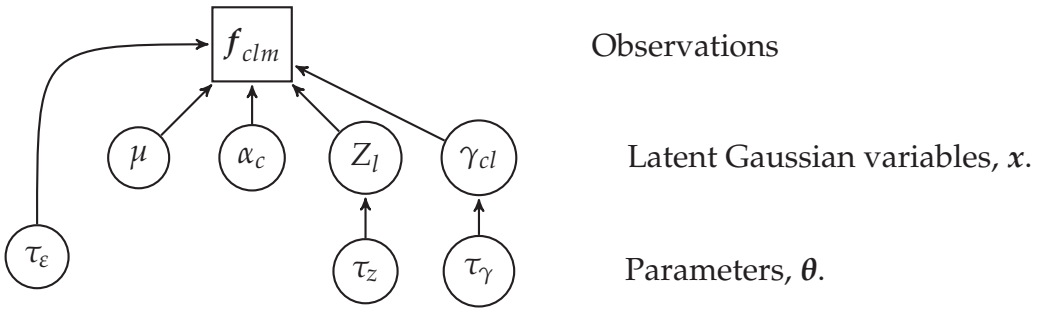


Figure 5.7: DAG for the basic model containing no indicator variables.

5.3.3 Bayesian linear Gaussian model with indicators

In the context of the QFA experiment, it is of great interest to infer the genetic interaction strengths for each possible *orf* Δ and of even greater interest to identify which of these interaction strength values contribute *significantly* to the model, since these would denote that the corresponding ORF exhibits epistasis when deleted. Identifying the significant interactions which should be included in the model can be achieved by incorporating variable selection as part of the model, such as Kuo and Mallick (1998) binary indicator variables. For our model, δ_l

represents the binary indicator for the l-th *orf*Δ, where $\delta_l = 1$ if the corresponding interaction strength is significant and should be included in the model, and $\delta_l = 0$ if not. The indicator will be incorporated into Equation (5.3) of the linear Gaussian model in Section 5.3.2:

$$f_{clm} = \mu_c + Z_l + \delta_l \gamma_{cl} + \epsilon_{clm}, \tag{5.4}$$

where $\delta_l \sim \text{Bern}(0.05)$ as it is expected that only about 5% of the ORF deletions will interact significantly. Significant interactions exist where $\text{P}(\delta_l = 1) > 0.5$, hence MCMC simulations can identify such interactions where $\bar{\delta}_l > 0.5$ and the corresponding interaction strength is the average value of $\gamma_{cl}\delta_l$ for each *orf*Δ.

The model with the Kuo & Mallick indicator is specified as,

$$f_{clm} = \mu + \alpha_c + Z_l + \delta_l \times \gamma_{cl} + \varepsilon_{clm} \tag{5.5}$$

$$\begin{aligned} \mu &\sim \text{N}(0, 0.001^{-1}) & Z_l &\sim \text{N}(0, \tau_z^{-1}) \\ \alpha_c &\begin{cases} \sim \text{N}(0, 0.001^{-1}) & \text{if } c = 0 \\ = 0 & \text{if } c = 1 \end{cases} & \gamma_{cl} &\begin{cases} = 0 & \text{if } c = 0 \\ \sim \text{N}(0, \tau_\gamma^{-1}) & \text{if } c = 1 \end{cases} \\ \varepsilon_{clm} &\sim \text{N}(0, \tau_\varepsilon^2) & \tau_\varepsilon, \tau_z, \tau_\gamma &\sim Ga(1, 0.00005) \\ \delta_l &\sim \text{Bern}(p_\delta) \end{aligned} \tag{5.6}$$

A DAG representation of this model including Kuo & Mallick indicators is shown in Figure 5.8.

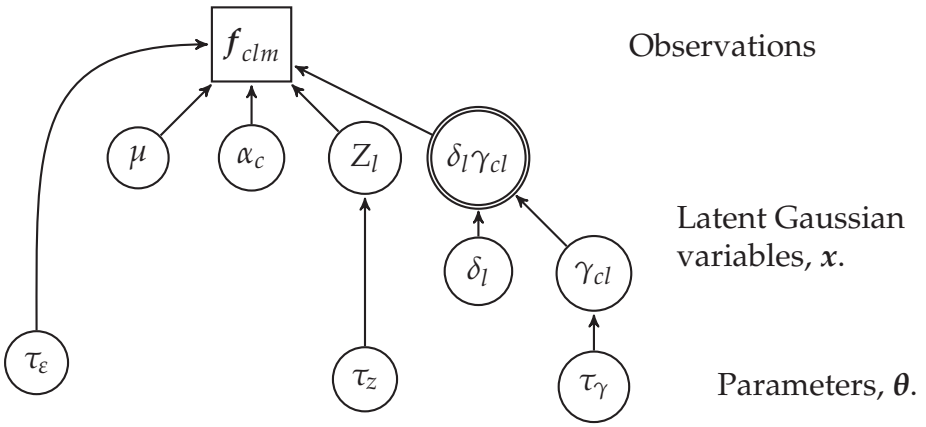


Figure 5.8: DAG for the model now including binary indicators to select γ_{1l} variables. The node with a double circle border represents a deterministic node.

5.3.4 Reparameterising with a sum-to-zero contrast

The representation of γ_{cl} given in Equation (5.6), in combination with the binary indicator δ_l, will lead to a bi-modal distribution for the parameter Z_l under certain circumstances. This happens whenever the corresponding indicator δ_l does not spend the vast majority of its time in only one of the include/exclude states of $\delta_l = 1$ or $\delta_l = 0$, respectively—the closer the mean value $\bar{\delta}_l$ is to 0.5, the more pronounced the bi-modal shape of the density of Z_l will become, leading to a poor effective sample size. This will be explored in more detail in Section 5.6.4.

To combat the bi-modal density of Z_l created by the indicator variables, an alternative parameterisation will be explored. This will match the model given in Section 5.3.3 along with all definitions, except for the definition of γ_{cl} in Equation (5.6), where the offset interaction strength is replaced by a sum-to-zero contrast:

$$\gamma_{cl} = \begin{cases} -\frac{1}{2}\gamma_l & \text{if } c = 0 \\ +\frac{1}{2}\gamma_l & \text{if } c = 1 \end{cases}, \quad \text{where } \gamma_l \sim \mathrm{N}(0, \tau_\gamma^{-1}) \tag{5.7}$$

Note that under this parameterisation, the interpretation and magnitude of the genetic interaction strength γ_l remains the same as the definition for γ_{1l} in Equation (5.6).

5.4 Joint distributions for linear Gaussian models

5.4.1 Model without indicators

Using Bayes' theorem, the target distribution can be factorised as,

$$\begin{aligned} \pi(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y}) &= \frac{\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})\pi(\boldsymbol{x}, \boldsymbol{\theta})}{\pi(\boldsymbol{y})} \\ &\propto \pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})\pi(\boldsymbol{x} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta}). \end{aligned} \tag{5.8}$$

We will calculate each of these terms in turn, starting with the observation likelihood distribution, $\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$:

$$\begin{aligned}
\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) &= \prod_{c=0}^{1} \prod_{l=1}^{p} \prod_{m=1}^{r_l} \pi(f_{clm} \mid \boldsymbol{x}, \boldsymbol{\theta}) \\
&= \prod_{l=1}^{p} \prod_{m=1}^{r_l} \pi(f_{0lm} \mid \boldsymbol{x}, \boldsymbol{\theta}) \pi(f_{1lm} \mid \boldsymbol{x}, \boldsymbol{\theta}) \\
&= \prod_{l=1}^{p} \prod_{m=1}^{r_l} \sqrt{\frac{\tau_\varepsilon}{2\pi}} \exp\left\{-\frac{\tau_\varepsilon}{2}(f_{0lm} - \hat{f}_{0l})^2\right\} \sqrt{\frac{\tau_\varepsilon}{2\pi}} \exp\left\{-\frac{\tau_\varepsilon}{2}(f_{1lm} - \hat{f}_{1l})^2\right\} \\
&= \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l} \exp\left\{-\frac{\tau_\varepsilon}{2} \sum_{l=1}^{p} \sum_{m=1}^{r_l} \left[(f_{0lm} - \hat{f}_{0l})^2 + (f_{1lm} - \hat{f}_{1l})^2\right]\right\},
\end{aligned} \tag{5.9}$$

where r_l denotes the number of replicates for the l-th *orf*Δ in the control (or query) strain. As *orf*Δ are performed on the plates to an indentical layout for both control and query strains, there are the same p *orf*Δ and number of replicates, r_l, $l = 1, \ldots, p$ when $c = 0$ and $c = 1$.

Note fitted values $\hat{f}_{cl}$ have different equations based on whether they are control ($c = 0$) or query ($c = 1$). Each is defined as follows:

$$\hat{f}_{0l} = \mu + \alpha + Z_l; \qquad \hat{f}_{1l} = \mu + Z_l + \gamma_{1l}.$$

This provides the distribution,

$$\begin{aligned}
\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) &= \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l} \\
&\times \exp\left\{-\frac{\tau_\varepsilon}{2} \sum_{l=1}^{p} \sum_{m=1}^{r_l} \left[(f_{0lm} - \mu - \alpha - Z_l)^2 + (f_{1lm} - \mu - Z_l - \gamma_{1l})^2\right]\right\}.
\end{aligned} \tag{5.10}$$

The prior distribution of the latent variables, $\pi(\boldsymbol{x} \mid \boldsymbol{\theta})$, is given by,

$$\begin{aligned}\pi(\boldsymbol{x} \mid \boldsymbol{\theta}) &= \pi(\mu)\pi(\alpha_0)\prod_{l=1}^{p}\left[\pi(z_l \mid \tau_z)\pi(\gamma_{1l} \mid \tau_\gamma)\right] \\ &= \sqrt{\frac{\tau_\mu}{2\pi}}\exp\left\{-\frac{\tau_\mu}{2}(\mu-\mu_\mu)^2\right\}\sqrt{\frac{\tau_\alpha}{2\pi}}\exp\left\{-\frac{\tau_\alpha}{2}(\alpha-\mu_\alpha)^2\right\} \\ &\quad\times\prod_{l=1}^{p}\left[\sqrt{\frac{\tau_z}{2\pi}}\exp\left\{-\frac{\tau_z}{2}(z_l-\mu_z)^2\right\}\sqrt{\frac{\tau_\gamma}{2\pi}}\exp\left\{-\frac{\tau_\gamma}{2}(\gamma_{1l}-\mu_\gamma)^2\right\}\right],\end{aligned}$$

and noting that all latent variables are assumed to have zero mean, i.e. $\mu_\mu = \mu_\alpha = \mu_z = \mu_\gamma = 0$, allows us to simplify this to,

$$\pi(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{\sqrt{\tau_\mu\tau_\alpha}}{2\pi}\left(\frac{\sqrt{\tau_z\tau_\gamma}}{2\pi}\right)^p\exp\left\{-\frac{\tau_\mu\mu^2}{2}-\frac{\tau_\alpha\alpha^2}{2}-\sum_{l=1}^{p}\left[\frac{\tau_z z_l^2}{2}+\frac{\tau_\gamma\gamma_{1l}^2}{2}\right]\right\}. \tag{5.11}$$

Finally, the distribution of the parameters, $\pi(\boldsymbol{\theta})$, is calculated to be,

$$\begin{aligned}\pi(\boldsymbol{\theta}) &= \pi(\tau_z)\,\pi(\tau_\gamma)\,\pi(\tau_\varepsilon) \\ &= \frac{b_z^{a_z}}{\Gamma(a_z)}\tau_z^{a_z-1}e^{-b_z\tau_z}\times\frac{b_\gamma^{a_\gamma}}{\Gamma(a_\gamma)}\tau_\gamma^{a_\gamma-1}e^{-b_\gamma\tau_\gamma}\times\frac{b_\varepsilon^{a_\varepsilon}}{\Gamma(a_\varepsilon)}\tau_\varepsilon^{a_\varepsilon-1}e^{-b_\varepsilon\tau_\varepsilon}.\end{aligned} \tag{5.12}$$

Using Equation (5.8), the joint conditional distribution can be written as the product of Equations (5.10), (5.11) and (5.12):

$$\begin{aligned}\pi(\boldsymbol{x},\boldsymbol{\theta} \mid \boldsymbol{y}) &= \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l} \\ &\quad\times\exp\left\{-\frac{\tau_\varepsilon}{2}\sum_{l=1}^{p}\sum_{m=1}^{r_l}\left[(f_{0lm}-\mu-\alpha-Z_l)^2+(f_{1lm}-\mu-Z_l-\gamma_{1l})^2\right]\right\} \\ &\quad\times\frac{\sqrt{\tau_\mu\tau_\alpha}}{2\pi}\left(\frac{\sqrt{\tau_z\tau_\gamma}}{2\pi}\right)^p\exp\left\{-\frac{\tau_\mu\mu^2}{2}-\frac{\tau_\alpha\alpha^2}{2}-\sum_{l=1}^{p}\left[\frac{\tau_z z_l^2}{2}+\frac{\tau_\gamma\gamma_{1l}^2}{2}\right]\right\} \\ &\quad\times\frac{b_z^{a_z}}{\Gamma(a_z)}\tau_z^{a_z-1}e^{-b_z\tau_z}\times\frac{b_\gamma^{a_\gamma}}{\Gamma(a_\gamma)}\tau_\gamma^{a_\gamma-1}e^{-b_\gamma\tau_\gamma}\times^{a_\varepsilon-1}\frac{b_\varepsilon^{a_\varepsilon}}{\Gamma(a_\varepsilon)}\tau_\varepsilon e^{-b_\varepsilon\tau_\varepsilon}.\end{aligned} \tag{5.13}$$

5.4.2 Model featuring indicators

The target distribution can now be factorised as,

$$\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I} \mid \boldsymbol{y}) \propto \pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I})\ \pi(\boldsymbol{I} \mid \boldsymbol{x}, \boldsymbol{\theta})\ \pi(\boldsymbol{x} \mid \boldsymbol{\theta})\ \pi(\boldsymbol{\theta}),$$

where $\boldsymbol{I}$ represents the distribution of the indicator variables.

The Kuo & Mallick method assumes independence between the indicator and its effect—for the QFA model, this is the interaction strength, γ. Observation of Figure 5.8 reveals that the indicators are also conditionally independent from other latent variables and parameters, allowing us to simplify $\pi(\boldsymbol{I} \mid \boldsymbol{x}, \boldsymbol{\theta})$ into $\pi(\boldsymbol{I})$:

$$\pi(\boldsymbol{x}, \boldsymbol{\theta}, \mid \boldsymbol{y}) \propto \pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I})\ \pi(\boldsymbol{I})\ \pi(\boldsymbol{x} \mid \boldsymbol{\theta})\ \pi(\boldsymbol{\theta}). \tag{5.14}$$

Once again, we aim to find the distribution for each of these terms separately, beginning with the density, $\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I})$

$$\begin{aligned}
&\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I}) \\
&\quad = \prod_{c=0}^{1} \prod_{l=1}^{p} \prod_{m=1}^{r_l} \pi(f_{clm} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I}) \\
&\quad = \prod_{l=1}^{p} \prod_{m=1}^{r_l} \pi(f_{0lm} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I}) \pi(f_{1lm} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I}) \\
&\quad = \prod_{l=1}^{p} \prod_{m=1}^{r_l} \sqrt{\frac{\tau_\varepsilon}{2\pi}} \exp\left\{-\frac{\tau_\varepsilon}{2}(f_{0lm} - \hat{f}_{0l})^2\right\} \sqrt{\frac{\tau_\varepsilon}{2\pi}} \exp\left\{-\frac{\tau_\varepsilon}{2}(f_{1lm} - \hat{f}_{1l})^2\right\} \\
&\quad = \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l} \exp\left\{-\frac{\tau_\varepsilon}{2} \sum_{l=1}^{p} \sum_{m=1}^{r_l} \left[(f_{0lm} - \hat{f}_{0l})^2 + (f_{1lm} - \hat{f}_{1l})^2\right]\right\}.
\end{aligned} \tag{5.15}$$

Note that this is the identical to the distribution in Equation (5.9), but here the fitted values $\hat{f}_{cl}$ are different for query ($c = 1$) strains. With the indicators and the model defined in Section 5.3.3, they are now defined as follows:

$$\hat{f}_{0l} = \mu + \alpha + Z_l; \qquad \hat{f}_{1l} = \mu + Z_l + \delta_l \gamma_{1l}.$$

This provides the distribution,

$$\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I}) = \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l} \times \exp\left\{-\frac{\tau_\varepsilon}{2}\sum_{l=1}^{p}\sum_{m=1}^{r_l}\left[(f_{0lm} - \mu - \alpha - Z_l)^2 + (f_{1lm} - \mu - Z_l - \delta_l\gamma_{1l})^2\right]\right\}. \tag{5.16}$$

Similarly, using the sum-to-zero contrast for γ_l as specified in Section 5.3.4, the fitted values $\hat{f}_{cl}$ will instead be,

$$\hat{f}_{0l} = \mu + \alpha + Z_l - \frac{1}{2}\delta_l\gamma_l; \qquad \hat{f}_{1l} = \mu + Z_l + \frac{1}{2}\delta_l\gamma_l,$$

providing the distribution,

$$\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I}) = \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l} \times \exp\left\{-\frac{\tau_\varepsilon}{2}\sum_{l=1}^{p}\sum_{m=1}^{r_l}\left[\left(f_{0lm} - \mu - \alpha - Z_l + \frac{1}{2}\delta_l\gamma_l\right)^2 + \left(f_{1lm} - \mu - Z_l - \frac{1}{2}\delta_l\gamma_l\right)^2\right]\right\}. \tag{5.17}$$

The binary indicators used in the Kuo & Mallick case here are $\delta_l \sim \text{Bern}(p_\delta)$, where $\delta_l = 1$ denotes the l-th variable would be included in the model and $\delta_l = 0$ otherwise. Hence,

$$Pr(\delta_l) = \begin{cases} p_\delta & \text{if } \delta_l = 1 \\ 1 - p_\delta & \text{if } \delta_l = 0 \end{cases}; \qquad \pi(\delta_l) = p_\delta^{\delta_l}(1 - p_\delta)^{1-\delta_l}. \tag{5.18}$$

By combining Equations (5.16) and (5.18) with the unchanged densities of Equations (5.11) and (5.12), the complete joint distribution in Equation (5.14) can

be constructed for the model in Section 5.3.3:

$$\begin{aligned}\pi(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{I}\mid\boldsymbol{y}) &= \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l}\prod_{l=1}^{p}\left[p_\delta^{\delta_l}(1-p_\delta)^{1-\delta_l}\right]\\ &\times\exp\left\{-\frac{\tau_\varepsilon}{2}\sum_{l=1}^{p}\sum_{m=1}^{r_l}\left[(f_{0lm}-\mu-\alpha-Z_l)^2+(f_{1lm}-\mu-Z_l-\delta_l\gamma_{1l})^2\right]\right\}\\ &\times\frac{\sqrt{\tau_\mu\tau_\alpha}}{2\pi}\left(\frac{\sqrt{\tau_z\tau_\gamma}}{2\pi}\right)^{p}\exp\left\{-\frac{\tau_\mu\mu^2}{2}-\frac{\tau_\alpha\alpha^2}{2}-\sum_{l=1}^{p}\left[\frac{\tau_z z_l^2}{2}+\frac{\tau_\gamma\gamma_{1l}^2}{2}\right]\right\}\\ &\times\frac{b_z^{a_z}}{\Gamma(a_z)}\tau_z^{a_z-1}e^{-b_z\tau_z}\times\frac{b_\gamma^{a_\gamma}}{\Gamma(a_\gamma)}\tau_\gamma^{a_\gamma-1}e^{-b_\gamma\tau_\gamma}\times^{a_\varepsilon-1}\frac{b_\varepsilon^{a_\varepsilon}}{\Gamma(a_\varepsilon)}\tau_\varepsilon e^{-b_\varepsilon\tau_\varepsilon}.\end{aligned}\tag{5.19}$$

To obtain the equivalent expression for the model in Section 5.3.4, Equation (5.16) is replaced with (5.17) to provide,

$$\begin{aligned}\pi(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{I}\mid\boldsymbol{y}) &= \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l}\prod_{l=1}^{p}\left[p_\delta^{\delta_l}(1-p_\delta)^{1-\delta_l}\right]\\ &\times\exp\left\{-\frac{\tau_\varepsilon}{2}\sum_{l=1}^{p}\sum_{m=1}^{r_l}\left[(f_{0lm}-\mu-\alpha-Z_l+\frac{1}{2}\delta_l\gamma_l)^2+(f_{1lm}-\mu-Z_l-\frac{1}{2}\delta_l\gamma_l)^2\right]\right\}\\ &\times\frac{\sqrt{\tau_\mu\tau_\alpha}}{2\pi}\left(\frac{\sqrt{\tau_z\tau_\gamma}}{2\pi}\right)^{p}\exp\left\{-\frac{\tau_\mu\mu^2}{2}-\frac{\tau_\alpha\alpha^2}{2}-\sum_{l=1}^{p}\left[\frac{\tau_z z_l^2}{2}+\frac{\tau_\gamma\gamma_{1l}^2}{2}\right]\right\}\\ &\times\frac{b_z^{a_z}}{\Gamma(a_z)}\tau_z^{a_z-1}e^{-b_z\tau_z}\times\frac{b_\gamma^{a_\gamma}}{\Gamma(a_\gamma)}\tau_\gamma^{a_\gamma-1}e^{-b_\gamma\tau_\gamma}\times^{a_\varepsilon-1}\frac{b_\varepsilon^{a_\varepsilon}}{\Gamma(a_\varepsilon)}\tau_\varepsilon e^{-b_\varepsilon\tau_\varepsilon}.\end{aligned}\tag{5.20}$$

5.5 Full conditional distributions for linear Gaussian model parameters

There are three parameters of the model in Section 5.4.1 which can each be calculated as follows:

5.5.1 Full conditional for τ_z

The parameter τ_z specifies the common precision parameter for the ORF fitness random effect for each of the p *orf*Δ. Each of these nodes, Z_l, $l = 1,\ldots,p$, are specified as $Z_l \sim \mathrm{N}(0,\tau_z)$, where $\tau_z \sim \mathrm{Ga}(a_z,b_z)$. The full conditional distribution

then becomes,

$$\begin{aligned}
&\pi(\tau_z \mid \tau_\gamma, \tau_\varepsilon, \mu, \alpha_c, Z_l, \gamma_{cl}, \varepsilon_{clm}) \\
&= \pi(\tau_z \mid Z_l) \propto \pi(\tau_z) \prod_{l=1}^{p} \pi(Z_l \mid \tau_z) \\
&\propto \frac{b_z^{a_z} \tau_z^{a_z - 1} e^{-b_z \tau_z}}{\Gamma(a_z)} \prod_{l=1}^{p} \tau_z^{1/2} \exp\left\{ -\frac{\tau_z}{2} (Z_l - 0)^2 \right\} \\
&\propto \tau_z^{a_z + p/2 - 1} \exp\left\{ -b_z \tau_z - \sum_{l=1}^{p} \frac{\tau_z}{2} Z_l^2 \right\} \\
&\propto \mathrm{Ga}\left(\tau_z; a_z + \frac{p}{2}, b_z + \frac{1}{2} \sum_{l=1}^{p} Z_l^2 \right).
\end{aligned}$$

5.5.2 Full conditional for τ_ε

The parameter τ_ε specifies the common precision parameter for the error in every observation. Each of these nodes, ε_{clm}, $c = \{0, 1\}$, $l = 1, \ldots, p$, and $m = 1, \ldots, r_l$, are specified as $\varepsilon \sim \mathrm{N}(0, \tau_\varepsilon)$, where $\tau_\varepsilon \sim \mathrm{Ga}(a_\varepsilon, b_\varepsilon)$. The full conditional distribution then becomes:

$$\begin{aligned}
&\pi(\tau_\varepsilon \mid \tau_\gamma, \tau_z, \mu, \alpha_c, Z_l, \gamma_{cl}, \varepsilon_{clm}) \\
&= \pi(\tau_\varepsilon \mid \varepsilon_{clm}) \propto \pi(\tau_\varepsilon) \prod_{c=0}^{1} \prod_{l=1}^{p} \prod_{m=1}^{r_l} \pi(\varepsilon_{clm} \mid \tau_\varepsilon) \\
&\propto \frac{b_\varepsilon^{a_\varepsilon} \tau_\varepsilon^{a_\varepsilon - 1} e^{-b_\varepsilon \tau_\varepsilon}}{\Gamma(a_\varepsilon)} \prod_{c=0}^{1} \prod_{l=1}^{p} \prod_{m=1}^{r_l} \tau_\varepsilon^{1/2} \exp\left\{ -\frac{\tau_\varepsilon}{2} (\varepsilon_{clm} - 0)^2 \right\} \\
&\propto \tau_\varepsilon^{a_\varepsilon + \sum_{l=1}^{p} (r_l) - 1} \exp\left\{ -b_\varepsilon \tau_\varepsilon - \sum_{c=1}^{L} \sum_{l=1}^{p} \sum_{m=1}^{r_l} \frac{\tau_\varepsilon}{2} \varepsilon_{clm}^2 \right\} \\
&\propto \mathrm{Ga}\left(\tau_\varepsilon; a_\varepsilon + \sum_{l=1}^{p} r_l, b_\varepsilon + \frac{1}{2} \sum_{c=1}^{L} \sum_{l=1}^{p} \sum_{m=1}^{r_l} \varepsilon_{clm}^2 \right)
\end{aligned}$$

where $\varepsilon_{clm} = f_{clm} - \hat{f}_{cl}$, therefore,

$$\propto \mathrm{Ga}\left(\tau_\varepsilon; a_\varepsilon + \sum_{l=1}^{p} r_l, b_\varepsilon + \frac{1}{2} \sum_{c=1}^{L} \sum_{l=1}^{p} \sum_{m=1}^{r_l} \left(f_{clm} - \hat{f}_{clm} \right)^2 \right)$$

5.5.3 Full conditional for τ_γ

The parameter τ_γ specifies the common precision parameter for the genetic interaction strengths for each of the p *orf*Δ. Each of the nodes for the query strains, γ_{1l}, $l = 1, \ldots, p$, are specified as $\gamma_{1l} \sim \mathrm{N}(0, \tau_\gamma)$, where $\tau_\gamma \sim \mathrm{Ga}(a_\gamma, b_\gamma)$. The full conditional distribution then becomes:

$$\begin{aligned}
\pi(\tau_\gamma \mid \tau_z, \tau_\varepsilon, \mu, \alpha_c, Z_l, \gamma_{cl}, \varepsilon_{clm}) &= \pi(\tau_\gamma \mid \gamma_{1l}) \\
&\propto \pi(\tau_\gamma) \prod_{l=1}^{p} \pi(\gamma_{1l} \mid \tau_\gamma) \\
&\propto \frac{b_\gamma^{a_\gamma} \tau_z^{a_\gamma - 1} e^{-b_\gamma \tau_\gamma}}{\Gamma(a_\gamma)} \prod_{l=1}^{p} \tau_\gamma^{1/2} \exp\left\{ -\frac{\tau_\gamma}{2} (\gamma_{1l} - 0)^2 \right\} \\
&\propto \tau_\gamma^{a_\gamma + p/2 - 1} \exp\left\{ -b_\gamma \tau_\gamma - \sum_{l=1}^{p} \frac{\tau_\gamma}{2} \gamma_{1l}^2 \right\} \\
&\propto \mathrm{Ga}\left(\tau_\gamma; a_\gamma + \frac{p}{2}, b_\gamma + \frac{1}{2} \sum_{l=1}^{p} \gamma_{1l}^2 \right)
\end{aligned}$$

Note that this is analogous to the calculation for τ_z given in Section 5.5.1.

5.5.4 Full conditional for δ_l

This applies to the model described in Section 5.4.2. The parameter δ_l is a binary indicator to determine which of the genetic interaction strength terms for each of the p *orf*Δ should be included in the model. The prior for this is $\delta_l \sim \mathrm{Bern}(p_\delta)$.

For the calculations in this section, we will relax the assumption that the r_l, $l = 1, \ldots, p$, are the same for both control and query strains. We denote r_{0l} and r_{1l} to be the number of replicates for the l-th *orf*Δ in the control and query strains respectively. In practice, we find that $r_{0l} = r_{1l}$ due to the fixed *orf*Δ pattern on the agar plates.

For fixed $\gamma_{0l} = 0$

When applied the model described in Section 5.4.2, the full conditional is,

$$\begin{aligned}
&\pi(\delta_l \mid \tau_\gamma, \tau_z, \tau_\varepsilon, \mu, \alpha_c, Z_l, \gamma_{cl}, \varepsilon_{clm}, \boldsymbol{y}) \\
&= \pi(\delta_l \mid f_{1lm}) \propto \pi(\delta_l) \prod_{m=1}^{r_{1l}} \pi(f_{1lm} \mid \delta_l) \\
&\propto p^{\delta_l}(1-p)^{1-\delta_l} \prod_{m=1}^{r_{1l}} \frac{\tau_\varepsilon}{\sqrt{2\pi}} \exp\left\{-\frac{\tau_\varepsilon}{2}(f_{1lm} - \hat{f}_{1l})^2\right\} \\
&\propto p^{\delta_l}(1-p)^{1-\delta_l} \left(\frac{\tau_\varepsilon}{2\pi}\right)^{r_{1l}} \exp\left\{-\frac{\tau_\varepsilon}{2} \sum_{m=1}^{r_{1l}} (f_{1lm} - \mu - Z_l - \delta_l\gamma_{1l})^2\right\}. \qquad (5.21)
\end{aligned}$$

On noting that

$$\begin{aligned}
(f_{1lm} - \hat{f}_{1l})^2 &= (f_{1lm} - \mu - Z_l - \delta_l\gamma_{1l})^2 \\
&= -2f_{1lm}\delta_l\gamma_{1l} + 2\mu\delta_l\gamma_{1l} + 2z_l\delta_l\gamma_{1l} + \delta_l^2\gamma_{1l}^2 + C,
\end{aligned}$$

where C are other terms constant with respect to δ_l, additional simplification can be performed as $\delta_l^2 = \delta_l$ since δ_l only takes the values of 0 or 1:

$$\left(f_{1lm} - \hat{f}_{1l}\right)^2 \propto 2\delta_l\gamma_{1l}\left(\mu + z_l + \frac{1}{2}\gamma_{1l} - f_{1lm}\right),$$

and now observing that $\mu + z_l + \gamma_{1l} = \hat{f}_{1l}$, means

$$\left(f_{1lm} - \hat{f}_{1l}\right)^2 = 2\delta_l\gamma_{1l}\left(\hat{f}_{1l} - \frac{1}{2}\gamma_{1l} - f_{1lm}\right).$$

Substituting this into Equation (5.21) gives,

$$\begin{aligned}
&\propto p^{\delta_l}(1-p)^{1-\delta_l} \exp\left\{-\frac{\tau_\varepsilon}{2} \sum_{m=1}^{r_{1l}} 2\delta_l\gamma_{1l}\left(\hat{f}_{1l} - \frac{1}{2}\gamma_{1l} - f_{1lm}\right)\right\} \\
&\propto p^{\delta_l}(1-p)^{1-\delta_l} \exp\left\{r_{1l}\tau_\varepsilon\delta_l\gamma_{1l}\left(\frac{1}{r_{1l}}\sum_{m=1}^{r_{1l}} f_{1lm} + \frac{1}{2}\gamma_{1l} - \hat{f}_{1l}\right)\right\} \qquad (5.22)
\end{aligned}$$

For sum-to-zero γ_l contrasts

When applied the model described in Section 5.3.4, the full conditional is a development of Equation (5.21):

$$\pi(\delta_l \mid \tau_\gamma, \tau_z, \tau_\varepsilon, \mu, \alpha_c, Z_l, \gamma_{cl}, \varepsilon_{clm}, \boldsymbol{y}) \propto p^{\delta_l}(1-p)^{1-\delta_l}\left(\frac{\tau_\varepsilon}{2\pi}\right)^{r_{0l}+r_{1l}} \times \exp\left\{-\frac{\tau_\varepsilon}{2}\left[\sum_{m=1}^{r_{0l}}\left(f_{0lm}-\mu-\alpha-Z_l+\frac{\delta_l\gamma_l}{2}\right)^2+\sum_{m=1}^{r_{1l}}\left(f_{1lm}-\mu-Z_l-\frac{\delta_l\gamma_l}{2}\right)^2\right]\right\} \tag{5.23}$$

Using a method analogous to when the fixed $\gamma_{0l}=0$ parameterisation was used, it can be seen that,

$$\begin{aligned}\left(f_{0lm}-\mu-\alpha-Z_l+\frac{1}{2}\delta_l\gamma_l\right)^2 &= \delta_l\gamma_l\left(f_{0lm}-\mu-\alpha-Z_l+\frac{1}{4}\gamma_l\right)+C_1\\ &= \delta_l\gamma_l\left(f_{0lm}-\frac{1}{4}\gamma_l-\hat{f}_{0l}\right)+C_1,\\ \left(f_{1lm}-\mu-Z_l-\frac{1}{2}\delta_l\gamma_{1l}\right)^2 &= \delta_l\gamma_l\left(\mu+z_l+\frac{1}{4}\gamma_l-f_{1lm}\right)+C_2\\ &= \delta_l\gamma_l\left(\hat{f}_{1l}-\frac{1}{4}\gamma_l-f_{1lm}\right)+C_2,\end{aligned}$$

where C_1 & C_2 are other terms constant with respect to δ_l. Substituting this into equation (5.23) gives,

$$\pi(\delta_l \mid \tau_\gamma, \tau_z, \tau_\varepsilon, \mu, \alpha_c, Z_l, \gamma_{cl}, \varepsilon_{clm}, \boldsymbol{y}) \propto p^{\delta_l}(1-p)^{1-\delta_l}\times \exp\left\{\frac{\tau_\varepsilon\delta_l\gamma_l}{2}\left(\sum_{m=1}^{r_{1l}}f_{1lm}-\sum_{m=1}^{r_{0l}}f_{0lm}+r_{0l}\hat{f}_{0l}-r_{1l}\hat{f}_{1l}+\frac{r_{0l}r_{1l}}{2}\gamma_l\right)\right\}. \tag{5.24}$$

Normalising and simulating the next indicator values

We normalize the probability using,

$$\Pr(\delta_l=1 \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}) = \frac{\pi(\delta_l=1 \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})}{\pi(\delta_l=0 \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})+\pi(\delta_l=1 \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})}.$$

This results in the following full conditional distribution for the indicator variables:

$$\pi(\boldsymbol{I} \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \frac{1-p\exp\{\phi(\delta_l=0)\}}{(1-p)\exp\{\phi(\delta_l=0)\}+p\exp\{\phi(\delta_l=1)\}} & \text{if } \delta_l = 0 \\ \frac{p\exp\{\phi(\delta_l=1)\}}{(1-p)\exp\{\phi(\delta_l=0)\}+p\exp\{\phi(\delta_l=1)\}} & \text{if } \delta_l = 1 \end{cases}. \tag{5.25}$$

where, for the full conditional distribution in Equation (5.22) with $\gamma_{0l} = 0$ fixed,

$$\phi(\delta_l) = r_{1l}\tau_\varepsilon\delta_l\gamma_{1l}\left(\frac{1}{r_{1l}}\sum_{m=1}^{r_{1l}} f_{1lm} + \frac{1}{2}\gamma_{1l} - \hat{f}_{1l}\right),$$

and for the full conditional distribution in Equation (5.24) using sum-to-zero contrasts,

$$\phi(\delta_l) = \frac{\tau_\varepsilon\delta_l\gamma_l}{2}\left(\sum_{m=1}^{r_{1l}} f_{1lm} - \sum_{m=1}^{r_{0l}} f_{0lm} + r_{0l}\hat{f}_{0l} - r_{1l}\hat{f}_{1l} + \frac{r_{0l}r_{1l}}{2}\gamma_l\right).$$

Note that when $\delta_l = 0$, $\phi(0) = \exp\{0\} = 1$ in both of these parameterisations.

By generating a value u, a realisation from a $\mathrm{U}(0,1)$ distribution, we can compare this value to the probability in Equation (5.25). Hence, we can set $\delta_l = 1$ if,

$$u < \frac{p\exp\{\phi(1)\}}{(1-p)\exp\{\phi(0)\} + pe^\phi\exp\{\phi(1)\}} = \frac{pe^\phi}{(1-p)+pe^\phi}, \tag{5.26}$$

and $\delta_l = 0$, otherwise. Note that we have written e^ϕ to represent $\exp\{\phi(1)\}$ for notational simplicity.

Care should be taken when calculating e^ϕ, as large values of ϕ can cause numerical overflow. In software implementations that catch overflow and represent the value as positive ∞, the normalising step in Equation (5.25) will most likely evaluate to `NaN` or 0 rather than 1, resulting in Equation (5.26) always evaluating to false and $\delta_l = 0$. This also occurs as underflow if ϕ is too small, but this would cause Equation (5.26) to correctly default to 'false' anyway.

Use of the log-sum-exp method can provide a stable way to handle this issue of numerical under- and over-flow (Murphy, 2012), by observing the following:

$$\Pr(\delta = 1 \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}) = \frac{pe^\phi}{q + pe^\phi}.$$

Now letting $m = \max\{0, \phi\}$,

$$\begin{aligned} &= \frac{e^{-m}\left[pe^{\phi}\right]}{e^{-m}\left[q + pe^{\phi}\right]} \\ &= \frac{pe^{\phi-m}}{qe^{-m} + pe^{\phi-m}}. \end{aligned}$$

Since $m = \max\{0, \phi\}$, the largest possible value for e^{-m} and $e^{\phi-m}$ is 1, while the other term will be less than or equal to this. This minimises the risk of overflow or underflow occurring in the normalising step when e^{ϕ} is calculated, providing greater numerical stability to the operation.

5.5.5 Full conditional for τ_γ with Kuo & Mallick indicators

This distribution replaces that in Section 5.5.3, to incorporate the binary indicators featured in the Kuo & Mallick variable selection method in Section 5.4.2. The parameter τ_γ specifies the common precision parameter for the genetic interaction strengths for each of the p *orf*Δ. Each of the nodes for the query strains, γ_{1l}, $l = 1, \ldots, p$, are specified as $\gamma_{1l} \sim \mathrm{N}(0, \tau_\gamma)$, where $\tau_\gamma \sim \mathrm{Ga}(a_\gamma, b_\gamma)$. Here the distribution is now affected by what variables are currently selected for inclusion in the model. Variables which are currently included will be in the set Φ, of size p_Φ. The full conditional distribution then becomes,

$$\begin{aligned} &\pi(\tau_\gamma \mid \tau_z, \tau_\varepsilon, \mu, \alpha_c, Z_l, \gamma_{cl}, \delta_l, \varepsilon_{clm}) \\ &\qquad = \pi(\tau_\gamma \mid \delta_l\gamma_{1l}) \propto \pi(\tau_\gamma) \prod_{l=1}^{p} \pi(\delta_l\gamma_{1l} \mid \tau_\gamma) \\ &\qquad \propto \frac{b_\gamma^{a_\gamma} \tau_z^{a_\gamma - 1} e^{-b_\gamma \tau_\gamma}}{\Gamma(a_\gamma)} \prod_{l \in \Phi} \tau_\gamma^{1/2} \exp\left\{-\frac{\tau_\gamma}{2}(\delta_l\gamma_{1l} - 0)^2\right\} \\ &\qquad \propto \tau_\gamma^{a_\gamma + p_\Phi/2 - 1} \exp\left\{-b_\gamma\tau_\gamma - \sum_{l \in \Phi} \frac{\tau_\gamma}{2}\delta_l\gamma_{1l}^2\right\} \\ &\qquad \propto \mathrm{Ga}\left(\tau_\gamma; a_\gamma + \frac{p_\Phi}{2}, b_\gamma + \frac{1}{2}\sum_{l \in \Phi}\gamma_{1l}^2\right). \end{aligned}$$

5.6 Comparison of model efficiency for linear Gaussian models

5.6.1 About the QFA data

The data we use comes from the QFA experiment detailed by Addinall et al. (2011), where a logistic growth model has been fitted to the data obtained from the image analysis. We use MDRMDP[3] as a measure of culture fitness for *cdc13-1* and *URA3*Δ strains grown at 27 °C. 159 *orf*Δ were stripped from the analysis by Addinall et al. (2011) for varying experimental and biological issues which would cause spurious results for reasons known to be unrelated to epistasis. This leaves 35576 observations for 4135 possible *orf*Δ in each strain. For a QFA dataset containing 4135 different *orf*Δ, the natural ordering of the latent field may be $\mu, \alpha, Z_1, \ldots, Z_{4135}, \gamma_1, \ldots, \gamma_{4135}$, resulting in a latent dimension of 8272.

A subset of the data can also be used by selecting cultures grown on plate 15; a plate with known neutral and telomere-related genes, designed to indicate if the strain has grown as expected (Addinall et al., 2011). This contains at least 6 replicates for each of 50 possible *orf*Δ, allowing inference methods to be tested on smaller dimension datasets.

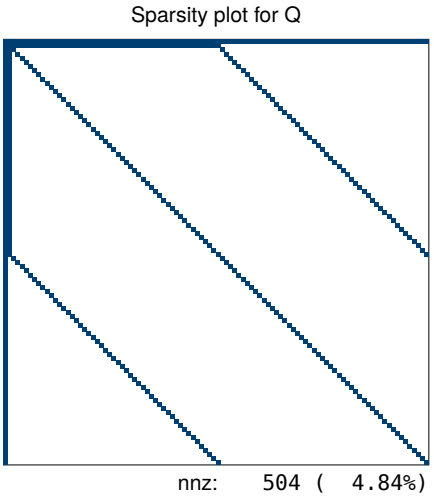
To verify the accuracy of the schemes under test, long thinned runs are performed and the densities are compared to ensure no discrepancies between results from `JAGS` and `GDAGsim`.

5.6.2 Comparison of results using permutation matrices

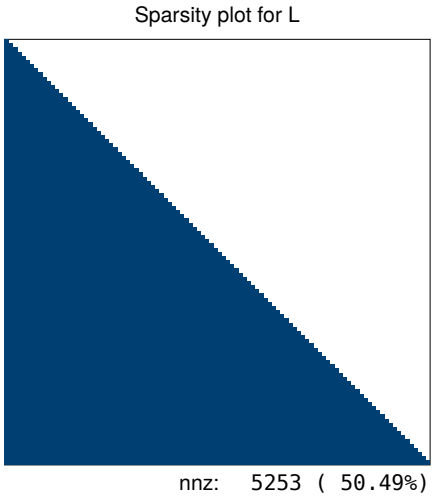
As explained in Section 2.3.2, the positioning of the non-zero values in the precision matrix, **Q**, can cause fill-in to occur in the Cholesky factor, **L**. Using the Plate 15 dataset and the model with Kuo & Mallick indicators described in Section 5.4.1, the benefit of the permutation matrix can be demonstrated visually in Figure 5.9.

When the Cholesky decomposition is done using the natural ordering, the Cholesky factor, **L**, (Figure 5.9b) suffers complete fill-in here, resulting in **L** being dense despite the original matrix (Figure 5.9a) being sparse. Having computed the AMD ordering and permuting the precision matrix (Figure 5.9c), the resulting

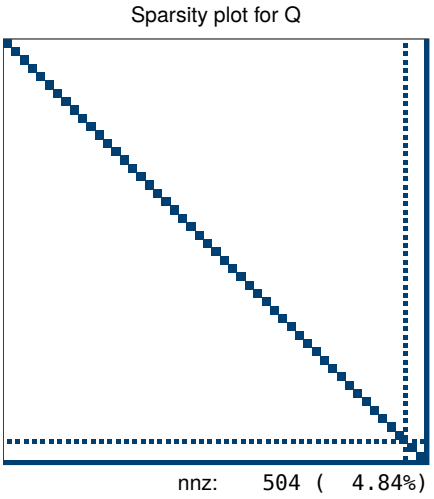
[3]MDRMDP = Maximum Doubling Rate × Maximum Doubling Potential for the yeast colony, as described in Section 5.2.1.



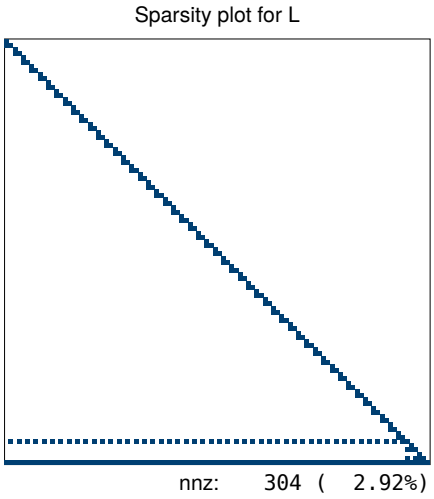
(a) The naturally ordered matrix **Q**. Number of non-zero entries in lower-triangle is 303 (5.768%)



(b) The Cholesky factor **L** of the naturally-ordered **Q** in Figure 5.9a. Number of non-zero entries in lower-triangle is 5253 (100%)



(c) The AMD ordered version of **Q** in Figure 5.9a. Number of non-zero entries in lower-triangle remains 303 (5.768%)



(d) The Cholesky factor **L** of the AMD-ordered **Q** in Figure 5.9c. Number of non-zero entries in lower-triangle is 304 (5.787%)

Figure 5.9: The location of non-zero entries in a 102×102 matrix. In the full matrix, **Q**, there are 504 non-zero elements (4.84%). For a fairer comparison, each caption includes the quantity and percentage of non-zero values in the *lower-triangle* only, since the Cholesky factors only occupy these elements. The natural-ordering for the variables from the model in Section 5.4.1 here is $(\mu, \alpha, z_1, \ldots, z_{50}, \gamma_1, \ldots, \gamma_{50})$.

Cholesky factor maintains the sparsity in Figure 5.9d.

As mentioned in Section 2.3.2, an optimum permutation can sometimes be identified manually. In this case, the optimal order when specifying the model in `GDAGsim` would be $(\gamma_1, z_1, \ldots, \gamma_{50}, z_{50}, \alpha, \mu)$. However, if the user was unable to identify this ordering, the AMD permutation would minimise the impact caused by Cholesky fill-in.

To understand the performance benefit that can be gained from using a permutation, a similar model that includes Kuo & Mallick, as defined in Section 5.3.3, on the full-size dataset containing all plates. The latent structure has a dimension of 8590 here. The time taken to perform the schemes with and without AMD permutations are listed in Table 5.1. It is worth noting that use of the permutation matrix is numerically equivalent and does not affect the mixing or accuracy of the scheme—there are only differences with the computation time and the memory usage in each implementation.

Permutation scheme	# of iter.	Time (s)	Avg. time per iter. (s)
Fixed permutation	100	3100	31.000
Dynamic AMD permutation	10000	704	0.070
Fixed AMD permutation	10000	713	0.071

Table 5.1: Time taken to obtain the stated number of unthinned iterations, depending on the permutation scheme in use.

It is immediately obvious from Table 5.1 that use of the permutation matrix allows more iterations to be computed in less time, in comparison to when no permutation matrix is used. The memory usage is also considerably lower when the permutation is applied.

The fixed permutation calculates the optimum permutation at the first iteration and stores it for future iterations. This removes the need to run an algorithm to find the optimum permutation at every iteration, which is unnecessary as the layout of the precision matrix won't change over each iteration. Somewhat counterintuitively, the scheme that reuses the original permutation is very slightly slower than the scheme that recomputes the approximately optimal permutation at each iteration. For this reason, we will use the dynamic permutation for the remainder of the results in this section.

We may find that on larger or models, or models with a more complicated dependancy structure, that the approximate permutation algorithm becomes more

costly to compute on each iteration, leading to a noticable benefit from fixing and reusing the original permutation. Further developments on the use of a fixed permutation will be discussed in Section 7.2.1.

5.6.3 Improvements from dynamic resizing of GDAG models

To find any improvement from implementing a dynamic resizing of the GDAG model, as explained in Section 4.10.6, the full URA3/CDC13-1 dataset is modelled with Kuo & Mallick indicators as defined in Section 5.4.2. The variable γ_{1l} will be removed from the GDAG model when the indicator is $\delta_l = 0$ for that iteration, and included if $\delta_l = 1$. The fixed-size and dynamically-resized models both use a re-computed AMD permutation before each Cholesky decomposition, as part of a variable selection Data Augmentation scheme (see Section 4.10.2). The resizing of the GDAG model does not affect the quality of the MCMC run, so only run times need to be considered.

Scheme	Time (s)
Fixed size	721
Dynamic resizing	624

Table 5.2: CPU user time taken to obtain 10000 unthinned iterations, depending on whether the size of the GDAG model is at the full, fixed size at each iteration, or resized to only include γ_{1l} variables where $\delta_l = 1$.

From the results in Table 5.2, there is a slight reduction in computation time over the course of a long run. Dynamically resizing the model requires additional overheads to correctly map which variables are being included, which negates some of the advantage obtained from working with smaller matrices. Despite the overheads, allowing the model to resize generally provides an appreciable performance improvement on large models or long runs that include a form of variable selection, where a relatively high proportion of variables are likely to be excluded at each iteration.

5.6.4 Improvements from model reparameterisation

Table 5.3 contains comparisons of the computation time and minimum effective sample size (ESS) obtained from a selection of schemes run in `JAGS` and using `GDAGsim` as part of a blocking method. The `JAGS` scheme performs a Gibbs sampler

on the variables, while the GDAGsim software is used for the Data Augmentation method described in Section 4.10.2. The smallest ESS value of all latent variables and parameters (excluding indicator, δ) simulated in the scheme is taken.

Parameterisation	Implementation	Time (s)	Min. ESS	Var.	ESS/s.
Fixed $\gamma_{1l} = 0$	JAGS	282	284	μ	1.01
	GDAG (fixed size)	721	483	z_{PGM2}	0.67
			2760	α	3.83
	GDAG (dyn. size)	674	483	z_{PGM2}	0.72
			2760	α	4.09
Sum-to-zero contrast	JAGS	429	300	μ	0.70
	GDAG (fixed size)	782	2191	γ_{YDR476C}	2.80
			6780	α	8.67
	GDAG (dyn. size)	751	2191	γ_{YDR476C}	2.92
			6780	α	9.03

Table 5.3: Time taken to obtain 10000 unthinned iterations in JAGS and the Data Augmentation using GDAGsim using both fixed and dynamic resizing.

The additional overheads associated with the blocking techniques mean that a GDAG scheme typically takes around 2.5 times longer to complete the same number of iterations than a JAGS scheme here. The ESS obtained by the GDAG implementation is generally much higher than that from the JAGS implementation for all variables. The lowest ESS from the Gibbs method in JAGS is 283.5 from the parameter μ, an expectedly low figure given that all observations depend on μ causing it to correlate heavily with other latent variables and mix poorly. In contrast, the ESS for the same parameter in a block updating scheme using GDAGsim was measured as 10000, indicating no autocorrelation in that chain. The poorer mixing for μ in the JAGS scheme is illustrated in Figure 5.10, where the blocking method from GDAGsim is not affected by the correlation with other latent variables.

The minimum ESS reading from the schemes using GDAGsim is hindered by the changeable state of the variable selection indicators. The random effect for *pgm2*Δ, z_{PGM2}, suffers from a bi-modal state whenever the corresponding indicator δ_{PGM2} spends an approximately $50 : 50$ split of time in its possible include/exclude states. The indicator will switch between states frequently if the interaction that the indicator is acting on, in this case $\gamma_{1,\text{PGM2}}$, is 'borderline' in whether it is a significant interaction or not.

Due to the parameterisation in use, including and excluding the $\gamma_{1,\text{PGM2}}$ for

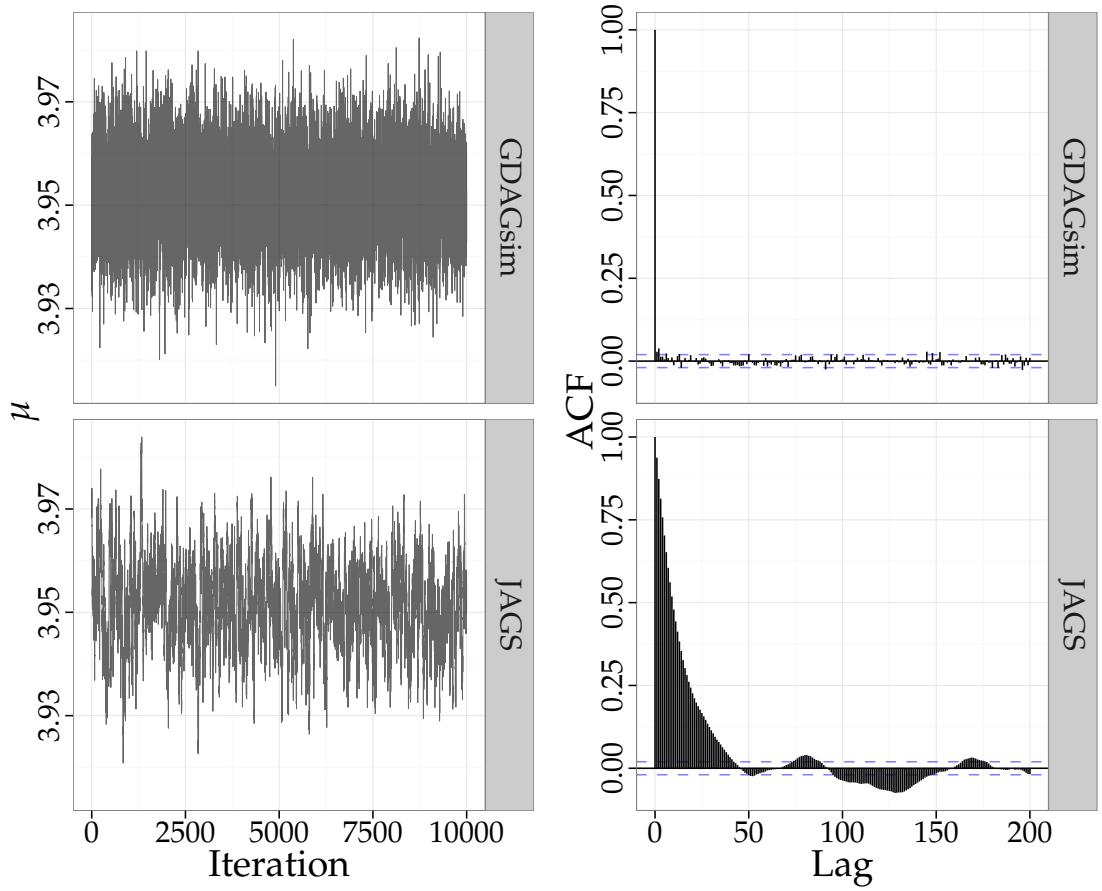


Figure 5.10: Trace plots (on left) of 10000 unthinned iterations for μ when performed under `GDAGsim` (top) and `JAGS` (bottom) schemes. Corresponding ACF plots with 95% intervals for each scheme are shown to the right.

roughly equal numbers of iterations has an effect on the z_{PGM2} parameter—the random effect z will swap between two modes to act as a compromise between frequently including and excluding the value of γ, as demonstrated by Figure 5.11. The bi-modal state in an MCMC chain will be penalised heavily in an ESS calculation and there will appear to be correlation between samples that don't stay in a single consistent state. By dissecting the chain for variables that suffer this problem and separately evaluating the mixing of each chain where the indicator is in the include and the exclude state, it can be seen that the mixing of samples for z is very good in its own state.

Since the mixing of the parameter z is very good when split by the state of the variable selection indicator, it could be argued that its poor performance in numerical measurements such as autocorrelation and ESS can be overlooked. However,

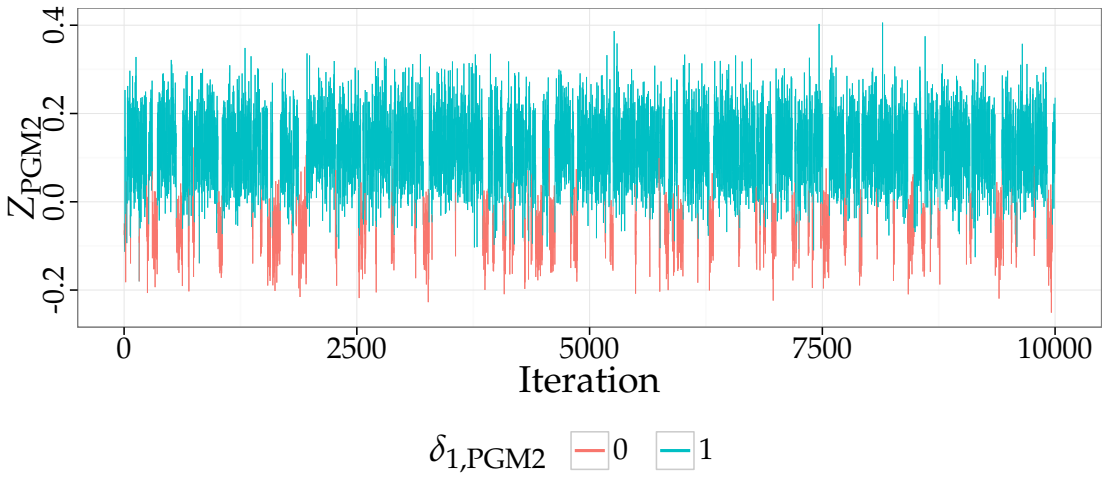


Figure 5.11: Trace plots of 10000 unthinned iterations for Z_{PGM2} when performed using `GDAGsim` methods. The colour of the line changes according to the current state of the indicator $\delta_{1,PGM2}$.

since the indicator acts on the genetic interaction strength in these models γ, and not on the ORF fitness, z, we would argue that this poor performance should not be overlooked.

To combat the problem of poor ESS and autocorrelation measurements in z, a reparameterisation is employed as described in Section 5.3.4 and the effects are investigated.

Computation time for the reparameterised model has increased over the equivalent implementation, due to a few extra computation overheads as more fitted values must be obtained to calculate the full conditional distributions that model parameters and indicators will be computed from.

While there has been little change to the smallest ESS value in the `JAGS` implementation, the results from the blocking methods in the `GDAGsim` implementation have improved considerably. The sum-to-zero contrast of the genetic interaction strength, γ_l, means the corresponding ORF fitness, z, is no longer behaving as a compromised value that has to switch between two different modes. Numerical assessments of these variables are now much better.

The minimum ESS value obtained is 2191 for the interaction strength of the *ydr476c*Δ, γ_{YDR476C}. This is also lowest as a result of the indicator being allowed to switch states frequently, but the relatively poor ESS is to be expected as a consequence of this—When the indicator $\delta_{\text{YDR476C}} = 0$, γ_{YDR476C} is simulated from the

prior distribution, but when $\delta_{\text{YDR476C}} = 1$, γ_{YDR476C} is simulated from a density that likely has a different mean to the prior mean. This means γ is expected to have a bi-modal distribution.

Even when including the poorest ESS from the γ_{YDR476C} variable, the `GDAGsim` implementation using a dynamic model resize has a greater ESS per second of computation time than both the implementation in `JAGS` and previous parameterisations of GDAG implementations. This suggests improved efficiency over standard Gibbs sampling methods, allowing large numbers of iterations with desirably low autocorrelations to be produced in less time.

If all γ variables are excluded from the analysis of ESS, on the grounds that they will appear artificially bad due to their indicator influenced bi-modal state, then the reparameterised blocking methods may be nearly 9 times more efficient, in terms of ESS per second on this large model, than standard Gibbs sampling methods. The minimum ESS reading from the schemes using `GDAGsim` is hindered by the changeable state of the variable selection indicators. The random effect z_{PGM2}, suffers from a bi-modal state whenever the corresponding indicator δ_{PGM2} spends an approximately $50 : 50$ split of time in its possible include/exclude states. The indicator will switch between states frequently if the interaction that the indicator is acting on, in this case $\gamma_{1,\text{PGM2}}$, is 'borderline' in whether it is a significant interaction or not.

5.6.5 Effects of Gibbs variable selection

We implement a Gibbs variable selection scheme for comparison against a simpler Kuo & Mallick indicator scheme that does not feature any pseudo-priors to simulate excluded variables. In both cases, these schemes use the sum-to-zero reparameterisation from Section 5.3.4 and feature no dynamic resizing of the model as mentioned in Section 4.10.6.

Timings and ESS measurements from these schemes can be viewed in Table 5.4. Where simpler indicator schemes caused a lower ESS for the γ_{YDR476C} chain due to its bi-modal behaviour, Gibbs variable selection avoids this by always ensuring that samples are generated close to the posterior. The lowest ESS reading from a GVS chain is three times higher than the equivalent model using Kuo & Mallick indicators. The GVS scheme takes longer to perform since a tuning period must be performed to find suitable hyperpriors and samples must then be taken from

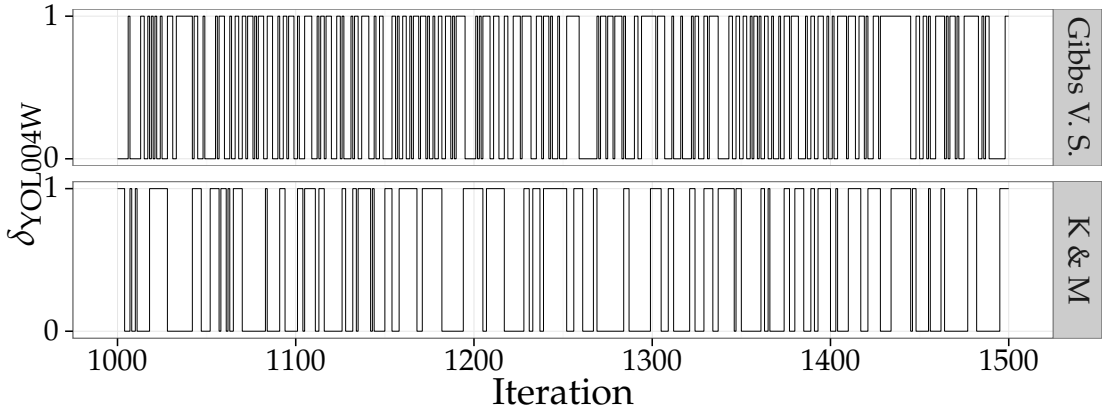


Figure 5.12: Trace plots of iterations 1000–1500 from an unthinned run for δ_{YOL004W} when performed under both the Gibbs variable selection scheme (top) and Kuo & Mallick indicators (bottom). For both methods, $\bar{\delta}_{\text{YOL004W}} \approx 0.52$.

the correct hyperpriors depending on which variables are currently featured in the model, as opposed to generating samples from a common prior distribution for the Kuo & Mallick scheme. Under these circumstances, there is still a greater minimum ESS/s from the GVS scheme.

Implementation	Time (s)	Min. ESS	Var.	ESS/s.
Kuo & Mallick indicators	782	2192	γ_{YDR476C}	2.80
		6780	α	8.67
Gibbs Variable Selection (incl. tuning and burning)	793 (825)	6107	γ_{YPL207W}	7.70 (7.40)
		6561	α	8.27 (7.95)

Table 5.4: CPU User time taken to obtain 10000 unthinned iterations before and after the implementation of GVS. Values in brackets factor in the time taken for an additional 400 iteration tuning and burning period.

Figure 5.12 demonstrates that the Gibbs variable selection scheme causes indicators to switch more frequently between the include/exclude states, without affecting the overall proportion of time spent in each state.

5.7 Latent Gaussian models

We will modify the linear Gaussian model from Section 5.3.4 to allow for a Gaussian distribution to be applied directly to fitness values, which is achieved by using an exponential link function on the linear predictor, creating a latent Gaussian model. The linear Gaussian model is equivalent to specifying a log-normal

$$
\begin{aligned}
F_{clm} &\sim \mathrm{N}(\hat{F}_{cl}, \tau_\varepsilon) & \hat{F}_{cl} &= \exp(\hat{f}_{cl}) \\
\hat{f}_{cl} &= \mu + \alpha_c + Z_l + \delta_l \times \gamma_{cl} & & \\
\mu &\sim \mathrm{N}(0, 0.001^{-1}) & Z_l &\sim \mathrm{N}(0, \tau_z^{-1}) \\
\gamma_l &\sim \mathrm{N}(0, \tau_\gamma^{-1}) & & \\
\alpha_c &\begin{cases} \sim \mathrm{N}(0, 0.001^{-1}) & \text{if } c = 0 \\ = 0 & \text{if } c = 1 \end{cases} & \gamma_{cl} &= \begin{cases} -\frac{1}{2}\gamma_l & \text{if } c = 0 \\ +\frac{1}{2}\gamma_l & \text{if } c = 1 \end{cases} \\
\delta_l &\sim \mathrm{Bern}(p_\delta) & \tau_\varepsilon, \tau_z, \tau_\gamma &\sim \mathrm{Ga}(1, 0.00005)
\end{aligned}
$$

Figure 5.13: Latent Gaussian model specification for QFA model.

distribution on the observation layer of fitness (e.g. MDRMDP) values, whereas the latent Gaussian model places a symmetric Gaussian distribution on the same values. Alternative distributions can also be applied to the observation layer if desired.

To implement this feature, we need to ensure that the likelihood for each observation is linked to a single latent variable, as shown in Equation (2.13), instead of a linear combination of latent variables which was allowed in the linear Gaussian models and demonstrated in Equation (2.2). This is achieved using an idea mentioned by Martins et al. (2013) by creating an additional latent node in the latent field, whose value is a linear combination of other nodes and has a precision of $\exp(15)$—not too high that it causes a numerical instability when a Cholesky decomposition is performed on the precision matrix, but small enough to ensure these nodes provide a negligible amount of error in the model. We will refer to these nodes as pseudo-deterministic nodes.

The latent Gaussian model that follows these criteria can be found in Figure 5.13, with a DAG representation shown in Figure 5.14. When the exponential link function is applied, each fitted fitness value, $\hat{F}_{cl}$, is equivalent to $e^{\mu} e^{\alpha_c} e^{Z_l} e^{\delta_l \gamma_l}$, in line with the multiplicative model of epistasis described in Section 5.2.1.

5.8 Joint distributions for latent Gaussian models

We can no longer sample from the full conditional distributions, as we could do with the latent models in Section 5.3, so we are limited in our choice of inference

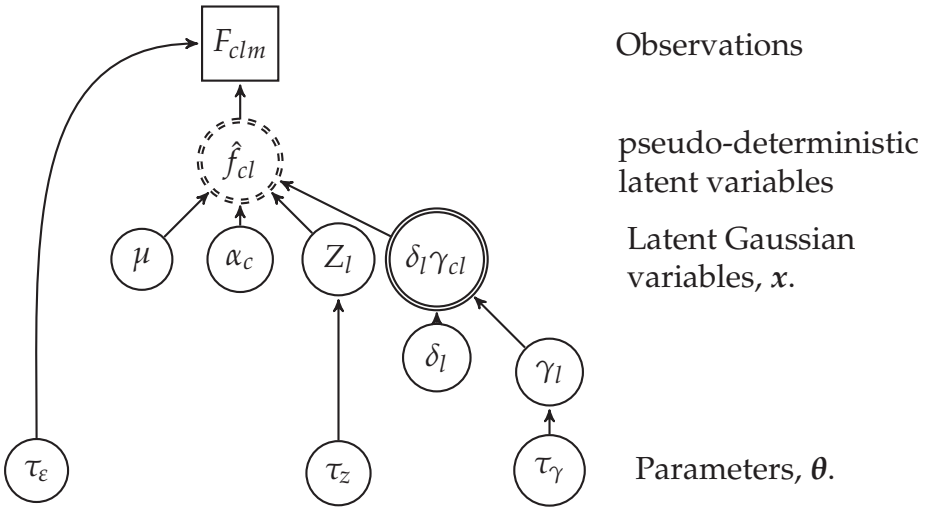


Figure 5.14: DAG for the latent Gaussian model including binary indicators to select γ_{1l} variables. The node with a double circle border represents a deterministic node, while the node with a dashed-double circle border represents a pseudo-deterministic node needs to be created in the prior structure.

schemes. Data augmentation cannot be used as it requires the ability to sample from the full conditional distributions. The Marginal updating scheme is of little use because we are interested in which *orf*Δ cause epistasis—this requires sampling genetic interaction strengths which the indicator values are based on, but neither of these are recorded in the Marginal scheme. We will attempt to use the Data Augmentation with Augmented Block (Section 4.10.3) to perform the inference. This means we will need to be able to sample from $\boldsymbol{I} \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}$ and all terms in Equation (4.8).

The density of the parameters, $\pi(\boldsymbol{\theta} \mid \boldsymbol{I})$, and the prior distribution of the latent variables, $\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{I})$, remain unchanged from Equations (5.12) and (5.11), respectively. The likelihood of the observations requires modification from Equation (5.15) to account for the new link function:

$$\pi(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{I}) = \left(\frac{\tau_\varepsilon}{2\pi}\right)^{\sum_{l=1}^{p} r_l} \exp\left\{-\frac{\tau_\varepsilon}{2}\sum_{l=1}^{p}\sum_{m=1}^{r_l}\left[(F_{0lm} - \hat{F}_{0l})^2 + (F_{1lm} - \hat{F}_{1l})^2\right]\right\},$$

where,

$$\hat{F}_{0l} = \exp\left(\mu + \alpha + Z_l - \frac{\delta_l\gamma_l}{2}\right); \qquad \hat{F}_{1l} = \exp\left(\mu + Z_l + \frac{\delta_l\gamma_l}{2}\right).$$

The indicator variables can still be generated from their full conditional distribution, $\boldsymbol{I} \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}$, where the density in Equation (5.23) is now modified to account for the new link function,

$$\pi(\delta_l \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}) \propto p^{\delta_l}(1-p)^{1-\delta_l} \times \exp\left\{-\frac{\tau_\varepsilon}{2}\left[\sum_{m=1}^{r_{0l}}\left(F_{0lm}-\hat{F}_{0l}\right)^2+\sum_{m=1}^{r_{1l}}\left(F_{1lm}-\hat{F}_{1l}\right)^2\right]\right\}. \quad (5.27)$$

The link function employed in this latent Gaussian model means there tends to be no useful simplification of $\phi(\delta_l)$, as was performed for the linear Gaussian model. As a result, $\phi(0)$ will not always take the value of zero, so must also be computed for use in Equation (5.26).

When performing the log-sum-exp method, as done for the linear Gaussian model, we now find the maximum, $m = \max\{\phi(0), \phi(1)\}$. The probability that each indicator should then be set to true is

$$\begin{aligned}\Pr(\delta = 1 \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}) &= \frac{pe^{\phi(1)}}{qe^{\phi(0)} + pe^{\phi(1)}} \\ &= \frac{pe^{\phi(1)-m}}{qe^{\phi(0)-m} + pe^{\phi(1)-m}}.\end{aligned}$$

5.9 Gaussian approximation terms for latent Gaussian model

At each iteration of the Data Augmentation scheme with Augmented block, a Gaussian approximation must be performed. The distribution of the optimised Gaussian approximation of the latent variables is used to generate a proposal for a new set of latent variables, before the density is used in the acceptance probability calculation. To perform this Gaussian approximation, as described in Section 3.2.2, we need to calculate the values of $\boldsymbol{b}$ and $\boldsymbol{c}$.

Suppose we have independent replicates, $j = 1, \ldots, r_i$, for observations, y_{ij}, all linked to latent node, x_i. Then we find the log-likelihood contribution term, $g_i(x_i)$

from Equation (3.9):

$$\begin{aligned} g_i(x_i) &= \prod_{j=1}^{r_i} \pi(y_{ij} \mid x_i, \boldsymbol{\theta}) \\ &= \sum_{j=1}^{r_i} \log[\pi(y_{ij} \mid x_i, \boldsymbol{\theta})] \\ &= \frac{r_i}{2}\log(2\pi) + \frac{r_i}{2}\log(\tau_\varepsilon) - \sum_{j=1}^{r_i} \tau_\varepsilon \left(2e^{x_i} - y_{ij}\right)^2, \end{aligned}$$

and taking the derivatives of these gives,

$$\begin{aligned} g_i'(x_i) &= \tau_\varepsilon e^{x_i} \sum_{j=1}^{r_i} \left(y_{ij} - e^{x_i}\right) = \tau_\varepsilon e^{x_i} \left[\sum_{j=1}^{r_i} (y_{ij}) - r_i e^{x_i}\right], \\ g_i''(x_i) &= \tau_\varepsilon e^{x_i} \left[\sum_{j=1}^{r_i} (y_{ij}) - 2r_i e^{x_i}\right]. \end{aligned}$$

The calculation of $\boldsymbol{c}$ follows as

$$c_i = -g_i''(x_i) = \tau_\varepsilon e^{x_i} \left(2e^{x_i} - \sum_{j=1}^{r_i} y_{ij}\right),$$

while the value of $\boldsymbol{b}$ is,

$$\begin{aligned} b_i &= g_i'(x_i) + \mu_i c_i \\ b_i &= \tau_\varepsilon e^{x_i} \left[\sum_{j=1}^{r_i} (y_{ij}) - e^{x_i}\right] + \tau_\varepsilon \mu_i e^{x_i} \left(2e^{x_i} - \sum_{j=1}^{r_i} y_{ij}\right). \end{aligned}$$

All values of b_i and c_i are set to zero if the latent variable does not have any directly linked observations. Once the values of $\boldsymbol{b}$ and $\boldsymbol{c}$ are known, the guess at the optimum mode of the latent variables can be found, as detailed in Section 3.2.2.

5.10 Results for latent Gaussian models

Using the smaller 'plate 15' dataset, we perform the same analysis to compare the Data Augmentation with Augmented block scheme from Section 4.10.3 against

Gibbs sampling methods from `JAGS` on the model in Section 5.7. For the plate 15 dataset containing 50 different *orf*Δ, the ordering of the latent field will be $\mu, \alpha, Z_1, \ldots, Z_{50}, \gamma_1, \ldots, \gamma_{50}, \hat{f}_{0,1}, \ldots, \hat{f}_{0,50}, \hat{f}_{1,1}, \ldots, \hat{f}_{1,50}$, resulting in a latent dimension of 204.

Implementation	Time (s)	Min. ESS	Var.	ESS/s.
`JAGS`	76	18	z_{NMD2}	0.24
		20	μ	0.26
`GDAGsim`	78	290	z_{PRM4}	3.72
		393	α	5.04

Table 5.5: CPU User time taken to obtain 10000 unthinned iterations for plate 15 as a latent Gaussian model.

Table 5.5 shows timings and minimum ESS from this comparison. Mixing for both these schemes is generally worse than was seen in the linear Gaussian models. There is no indication that poor mixing for the ORF fitness values is caused by a bi-modal state induced by the changing indicator state—these schemes both use the sum-to-zero contrasts for γ_l—indicating the worst mixing variables are indeed the z variables in both schemes. These schemes are likely to struggle with the smaller dataset as the model works best with a full genome-wide dataset.

In the case of the `JAGS` scheme the ESS is very low, so to verify this sample is not simply "noise" an additional run was performed using thinning of 10 iterations, which yielded an ESS for μ of 196.6, indicating that the quality of the output chain improves as expected when thinning is employed. On small scale models, the computation time per iteration is comparable, so the better mixing obtained from blocking methods provides a much more efficient scheme in terms of ESS/s.

When performed on a full QFA dataset, the Augmented block in Data Augmentation blocking method that uses `GDAGsim` fails as the ratio of accepted proposals drop considerably. In contrast, `JAGS` models continue to perform adequate inference after thinning.

Simulated data of varying dimension was created to test these blocking methods that fail. Due to the non-linear observation model, the quality of the latent Gaussian proposal deteriorates as the dimension of the system increases. As a consequence, these schemes have a tendency to converge to the "true" values of the simulated data before sticking to these values.

Since the linear Gaussian model in Section 5.3.4 is a special case of a latent Gaussian model, we can implement it using the the same Augmented block in

Data Augmentation MCMC method from Section 4.10.3. In this situation, we find it successfully performs inference on the full-size dataset, suggesting that proposals for the latent variables match well to the posterior, keeping the proportion of accepted proposals as the desired level. ESS/s figures are not as competitive with the equivalent Data Augmentation method in Section 4.6, primarily because the MH step means the chain will not move on all iterations.

A possible way to solve the issues encountered in the latent Gaussian model is to use multiple smaller blocks of latent variables which have a higher probability of being accepted, and deciding if each of these smaller-dimension proposals should be accepted in turn. However, reducing the size of the block can have a negative effect on the scheme's mixing, so this would form a compromise between the improved mixing of a large block and the greater acceptance rates of smaller blocks.

5.11 Model assessment methods

5.11.1 Assessing model accuracy

Working with large genome-wide datasets, where the functions of many genes are not fully understood already, makes it challenging to know whether these models are producing sensible results. There are some strategies that can be employed to decide if results look reasonable.

Using the 'plate 15' dataset allowed for the model to be created and tested quickly thanks to its small dimension, and since most of the genes selected for use on plate 15 were known to be neutral or to interact, it can be used to roughly gauge if the model is producing reasonable list of interacting genes. Once the model was working on plate 15, we created higher-dimension simulated datasets according to the model, allowing us to know which genes had a "true" genetic interaction. We then verified that the model was able to correctly identify which genes were interacting. Our tests on simulated data proved to be very accurate at identifying the "true" interactions, but it is always worth remembering that real data can, and probably will, behave differently to simulated data.

Residual plots were also investigated for these models to check whether residuals followed the Gaussian distribution that was expected. The results from these diagnostic plots were mostly satisfactory, albeit with some truncation in plots

of residuals against fitted fitness values since large negative values won't occur when the predicted fitness values are already close to zero—this combination of negative residual and small predicted fitness would imply a negative fitness value.

Cross-validation methods involve dividing the data into two non-overlapping subsets: a large training dataset and a smaller validation dataset. This is often used to check the model is able to accurately predict the outcomes for observations it hasn't previously seen, and assessed by finding the correlation between the prediction and actual observation. In the context of QFA, we consider the strategy of leaving out data when fitting these models is not a sensible strategy because these models rely on the full genome-wide sequence in order to accurately find interactors.

5.11.2 Comparison against results from Heydari et al. (2016)

The structure of our latent Gaussian model is less complicated than that seen in the Interaction Hierarchical Model (IHM) that Heydari et al. (2016) used to identify interacting *orf*Δ in the original QFA analysis. We expect the IHM should be more accurate at identifying *orf*Δ thanks to its increased complexity, while our latent Gaussian model structure will be more efficient at producing good quality inference.

To check that the results from both competing models are identifying similar results, a correlation plot can be produced to display the posterior means for interaction strengths obtained from each scheme—this can be seen in Figure 5.15. A positive linear correlation would indicate that the different models identify similar genetic interactions, and that the strongest interactions in our model appeared as the strongest interactions in the IHM. While a positive correlation is desirable, the points do not necessarily need to fall on the 1:1 line, as the models have different parameterisations for the interaction strength.

There are some points of concern which can be found in Figure 5.15. Points at the origin of the plot are acceptable, since this indicates both models are identifying a gene is neutral, but other any points lying along either axis at zero indicate that one model has identified an *orf*Δ as an interaction where the other model decided that the same *orf*Δ was neutral. Of the 502 interactions found by the latent Gaussian model and the 576 interactions by the IHM, 411 of these interactions

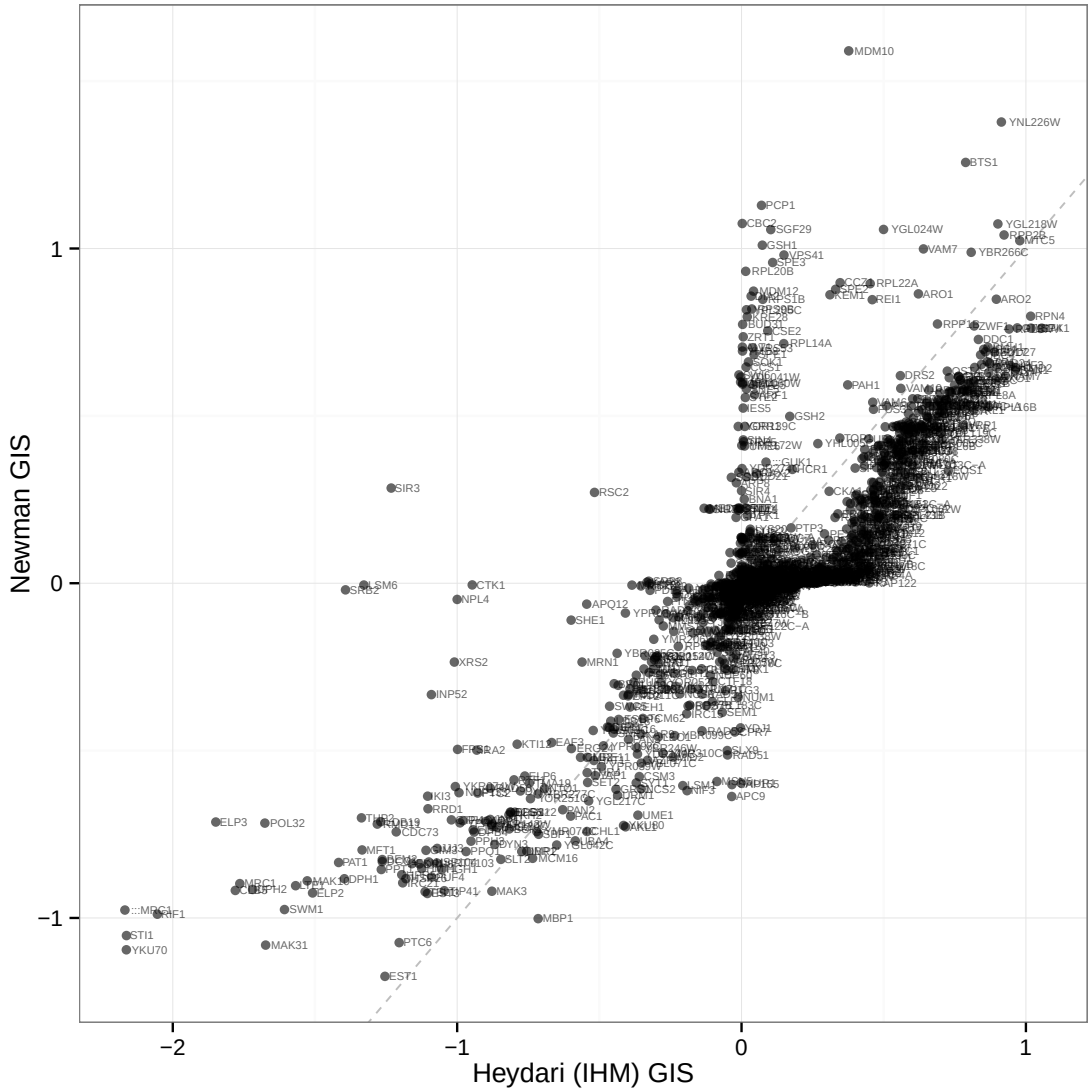


Figure 5.15: Correlation plot of the mean posterior genetic interaction strength from the Heydari et al. (2016) IHM and the latent Gaussian model specified in Section 5.7 (labelled as 'Newman GIS'). The dashed line represents the 1:1 line.

are common between both sets of results. Two of these overlapping hits appear in a concerning manner in Figure 5.15, since they appear in the top-left quadrant of the plot, indicating the two models have reached conflicting statements about whether *sir3*Δ and *rsc2*Δ cause suppressing or enhancing interactions. Overall, the plot shows a promising overlap in 'hits' and a positive and generally linear correlation in interaction strengths, suggesting the simplified latent Gaussian model performs acceptable inference in comparison to the IHM.

Chapter 6

Mini QFA

6.1 Introduction

Mini QFA is a recent and on-going development upon the previous QFA experiments seen in Chapter 5. It aims to find genetic epistasis between defective telomeres and the simultaneous deletion of two other non-essential genes of interest. As with QFA, this experiment is performed on *Saccharomyces cerevisiae*.

Groups of genes are known to work together to perform various operations, each according to their own interaction pathway. One relevant example that exists in *Saccharomyces cerevisiae* is the MRX complex, consisting of the genes *MRE11*, *RAD50* and *XRS2*. D'Amours and Jackson (2002) explains this complex is known to play a part in telomere length maintenance and DNA double-strand-break (DSB) repair pathways. Each gene in the complex plays its own part in these processes; if the complex is damaged by the deletion of one or more of these genes, the remaining genes will be unable to perform their task in maintaining telomere length or fixing a DSB.

Developing methods which can identify groups of genes that interact together with telomeres is of interest, not just for the case of *Saccharomyces cerevisiae*, but also for identifying homologues of these genes present in human cells. While *MRE11* and *RAD50* were found to exist in many species, a homologue of *XRS2* remained elusive. When Carney et al. (1998) discovered that *NBS1* deficiency prevented the growth of *MRE11*/*RAD50*, it was discovered that *NBS1* is the homologue of *XRS2* in vertebrates, thereby revealing the group on genes that affect telomere length maintenance and DSB repair in humans.

The previous QFA experiment was concerned with finding genetic epistasis between the defective telomere and a single deletion from across the whole genome, suggesting which of the 4294 non-essential genes (4135 after some were stripped from the analysis) interact with a defective telomere. The new experiment aims to delete all pairwise combinations of the non-essential genes from strains that feature one of two possible telomeres. Attempting to perform a genome-wide analysis of pairwise deletions across a set of 4135 genes with one of the telomeres would result in $4135^2 = 17,098,225$ possible combinations to check, even before any replicates have been taken; since each strain needs to be prepared, grown and photographed at regular time-points, this would not be a feasible experiment to perform at the laboratory under reasonable time and budget constraints, even with access to robotic equipment.

Therefore, this new experiment is performed by deleting two genes from a smaller selection of only 154 genes, leading to the name *Mini QFA*. The limit of 154 genes was pragmatically selected as it is the number of genes that can be grown on half an agar plate in 384-spot configuration; 384-spot format uses a 24×16 grid layout, but edge-colonies must be discarded from the experiment because they have a growth advantage due to the lack of competition for nutrients. The growth advantage of edge colonies is visible in the example 384-spot plate given in Figure 6.1, which shows the growth of yeast cultures after approximately 117 hours.

Genes were selected as part of the 154 *orf*Δ to be analysed if they fit any of these criteria: they affect telomere length, they were found to interact with telomere defects in the full QFA analysis, or they have homologues in human cells. Additionally, a very small number of genes featured in the study which are known neutral mutations, such as *his3*Δ, will be included in the selected 154 genes.

One concern exists about the lack of a genome-wide dataset because the selected 154 genes are an inherently biased selection, mostly chosen as these genes have exhibited an interaction in previous experiments. The inclusion of known neutral mutations could allow a fixed reference point, which the many interacting mutations can contrast against, with potential added benefits of helping prevent some identifiability issues.

A background screen is made with a gene deleted from the sequence, and divided up among the spots on each plate. Each query gene is then knocked out from each of the spots using a different screen. This results in $154^2 = 23716$

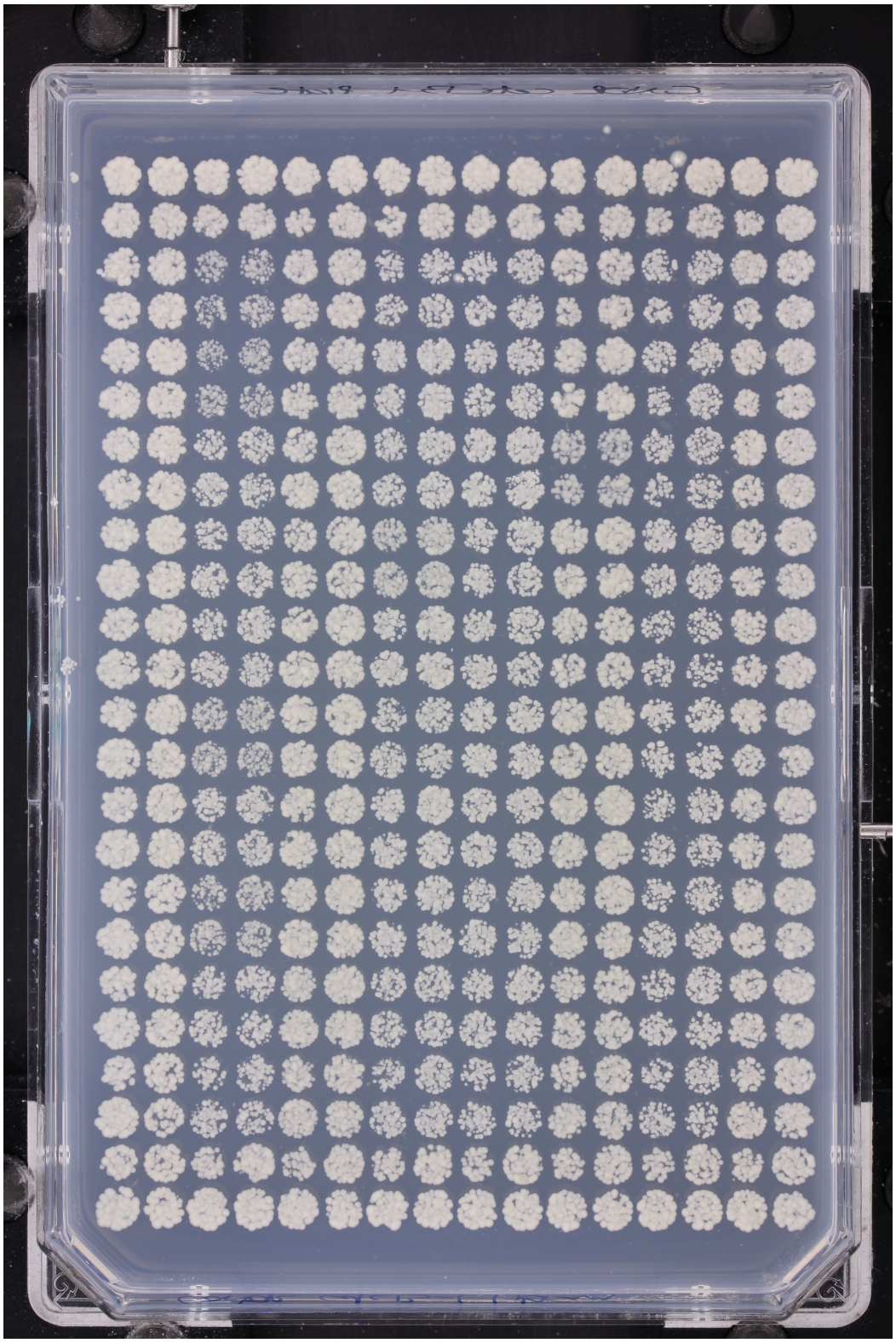


Figure 6.1: An example 384-spot format plate provided in the Colonyzer package (Lawless et al., 2010). The yeast cultures on this plate have grown for approximately 117 hours.

different pairwise combinations of deletions to be analysed in this "all-by-all" experiment.

We define the notation where a defective telomere (such as CDC13-1) is crossed with the deletion of two other genes (for the sake of this example, the background screen has YFG1 deleted, and the query screen deletes XYZ2[1]), then we denote this triple mutant strain as *cdc13-1 yfg1*Δ *xyz2*Δ.

We will find that for some of the observations, the same gene is deleted twice; once in the background screen and separately in the query screen. Using our notation, this situation of having the same gene deleted twice will be described as *cdc13-1 yfg1*Δ *yfg1*Δ, and we will refer to these as *double-l deletions* since the *l*-th ORF has been deleted twice. In such cases, we expect that all double-*l* deletion strains should *always* be dead[2] for reasons discussed in Section 5.1.3.

This experiment is performed at two different temperatures for each of two different telomeres, giving four different combinations: CDC13+ at 27°C and 33°C, and CDC13-1 at 27°C and 33°C. These are all treated as separate experiments and, as a result, they are analysed independently.

6.2 Modelling for Mini QFA

6.2.1 Models of epistasis

We aim to model for pairwise deletions that cause an interaction, which would be indicated by a deviation from expected behaviour. We use Fisher's multiplicative model of epistasis (Cordell, 2002), which was described previously in Section 5.2.2, to define what behaviour is expected. We will illustrate properties of this model by assigning hypothetical fitness values to some imaginary genes.

Suppose *cdc13-1 yfg1*Δ has a fitness of 2, *cdc13-1 xyz1*Δ has a fitness of 3, and *yfg1*Δ & *xyz1*Δ actually have independent functions so should have no interactions. We would expect to find that *cdc13-1 yfg1*Δ *xyz1*Δ has a fitness of $2 \times 3 = 6$.

Now suppose that *cdc13-1 yfg1*Δ has a fitness of 2, *cdc13-1 yfg3*Δ has a fitness of 3, and *yfg1*Δ & *yfg3*Δ do interact (for example, YFG1 may be involved in essential preparation for chromosome repair which YFG3 completes; the chromosome

[1]The genes YFG1 and XYZ2 are not real and they merely serve as placeholders; *YFG* is used here to stand for "***y**our **f**avourite **g**ene*".

[2]We may find extremely poor growth as opposed to zero growth from a truly dead strain.

cannot be repaired if either of these are missing). The multiplicative model would expect to find *cdc13-1 yfg1Δ yfg3Δ* to have a fitness of 6, but we might actually find the fitness is only 2. This deviation from the expected value is indicative that the pairwise deletion exhibits epistasis.

6.2.2 Standard Mini QFA model

Under the multiplicative model of genetic epistasis, we define the model as follows:

$$
\begin{aligned}
F_{cll'm} &\sim \mathrm{N}\left(\hat{F}_{cll'}, \tau_\varepsilon\right) & \hat{F}_{ll'} &= \exp\left(\hat{f}_{cll'}\right) \\
\hat{f}_{ll'} &= \mu + Z_l + Z_{l'} + \delta_{ll'}\gamma_{ll'} & & \\
\mu &\sim \mathrm{N}(0, 0.001^{-1}) & \gamma_{ll'} &\sim \mathrm{N}(0, \tau_\gamma^{-1}) \\
Z_l &\begin{cases} = 0 & \text{if } l = 1 \\ \sim \mathrm{N}(0, \tau_z^{-1}) & \text{otherwise} \end{cases} & \delta_{ll'} &\begin{cases} = 1 & \text{if } l = l' \\ \sim \mathrm{Bern}(p_\delta) & \text{otherwise} \end{cases} \\
\varepsilon_{ll'm} &\sim \mathrm{N}(0, \tau_\varepsilon^2) & \tau_\varepsilon, \tau_z, \tau_\gamma &\sim Ga(1, 0.00005)
\end{aligned}
\tag{6.1}
$$

where:

- F is a measure of fitness.
- l represents that ORF l is deleted from the background screen; l' denotes that ORF l is the query deletion. The order of these ORF deletions does not matter, excluding ORF $l = 1$ which we set to be the neutral deletion, HIS3.
- Z_l is the ORF fitness expected when ORF l is deleted from the strain. $Z_{l'}$ represents the same for the query gene that is deleted. Since $Z_1 = Z_{\text{HIS3}} = 0$, all other ORF fitnesses represent contrasts from a neutral deletion.
- μ represents an underlying overall mean fitness for the chromosome with its telomere. Since $Z_{\text{HIS3}} = 0$, the underlying mean is the fitness where a neutral HIS3 deletion has occurred.
- $\gamma_{ll'}$ is a genetic interaction strength between the two deleted genes.
- $\delta_{ll'}$ is a Kuo & Mallick indicator which determines whether there is a significant genetic epistasis between the deleted genes and the telomere. When

a double-l deletion occurs, the indicator is fixed to be 'on' as these strains should always be dead.

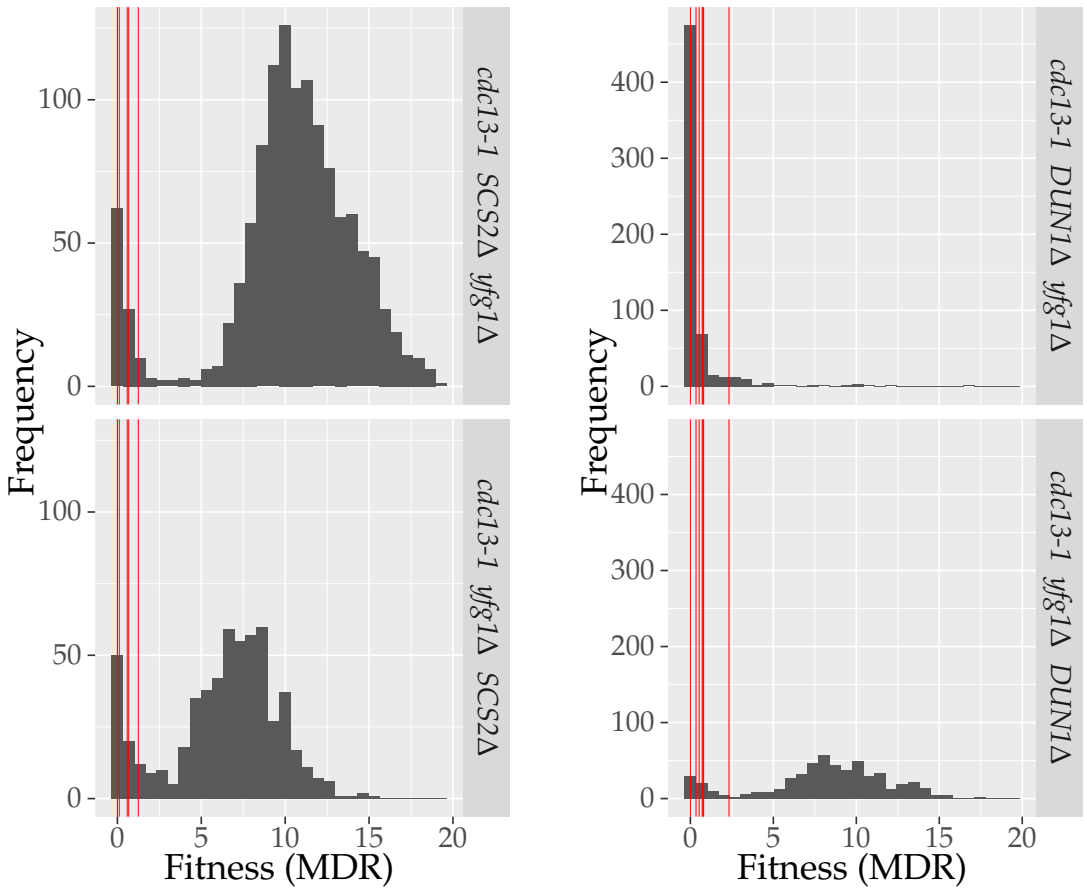
6.2.3 Model variations

The fitness measurements, $\boldsymbol{F}$, that are used in the model and described previously in Section 6.2.2 are inferred using an updated version of the `Colonyzer` software (Lawless et al., 2010) for image analysis and the fitting of logistic-growth curves. The parameters from fitting these growth curves allow for multiple possible measures of fitness, such as MDR, MDRMDP and nAUC which were described in Section 5.2.1, along with the doubling rate parameter, r, from the logistic growth model. There is current debate as to which is the best measurement to use, so models have been run on all four different fitness measures to investigate which may be most informative.

As many of the ORF fitnesses, $\boldsymbol{Z}$, selected for this study will often be quite far from zero—an artifact from selecting 154 genes that mostly interact with the telomere—we expect this distribution to have heavier tails than a Gaussian distribution. For this reason, we also explore the possibility of modelling ORF fitness using a t-distribution on 3 degrees of freedom, as this is more robust when presented with non-Normal data.

We have also explored whether ORF fitness distributions behave similarly if the gene of interest is deleted in the background screen or the query. For biological reasons, it is expected that the fitness of *cdc13-1 yfg1*Δ *xyz2*Δ should behave identically or similar to *cdc13-1 xyz2*Δ *yfg1*Δ, but investigations on early datasets showed examples where the fitness did not behave like this. Examples of this occurring are shown in Figure 6.2, where in Figure 6.2a there is a slight discrepancy between the fitness depending on where SCS2 was deleted. A more obvious case of this discrepancy is visible in Figure 6.2b. All *cdc13-1 dun1*Δ *yfg1*Δ strains were grown on the same plate and all had extremely poor growth, leading to the decision that DUN1 results should be stripped from the data and the plate grown and evaluated again.

To account for this in the model based on the current data we have been provided, we implement *asymmetric* ORF fitness values by treating the ORF fitness for the background screen deletion separately to the ORF fitness for the query screen deletion. This would be represented in the model as an adjustment to



(a) *scs2*Δ against all other ORF deletions

(b) *dun1*Δ against all other ORF deletions

Figure 6.2: Asymmetry in the ORF fitnesses depending on whether the gene was deleted in the background strain or the query. Red vertical lines denote the fitness of double-*l* deletions, i.e. the *cdc13-1 scs2*Δ *scs2*Δ fitness values in Figure (a).

Equation (6.1):

$$\hat{f}_{ll'} = \mu + Z^b_l + Z^q_{l'} + \delta_{ll'}\gamma_{ll'}$$
$$Z^b_l, Z^q_{l'} \begin{cases} = 0 & \text{if } l = 1 \\ \sim \mathrm{N}(0, \tau_z^{-1}) & \text{otherwise} \end{cases}$$

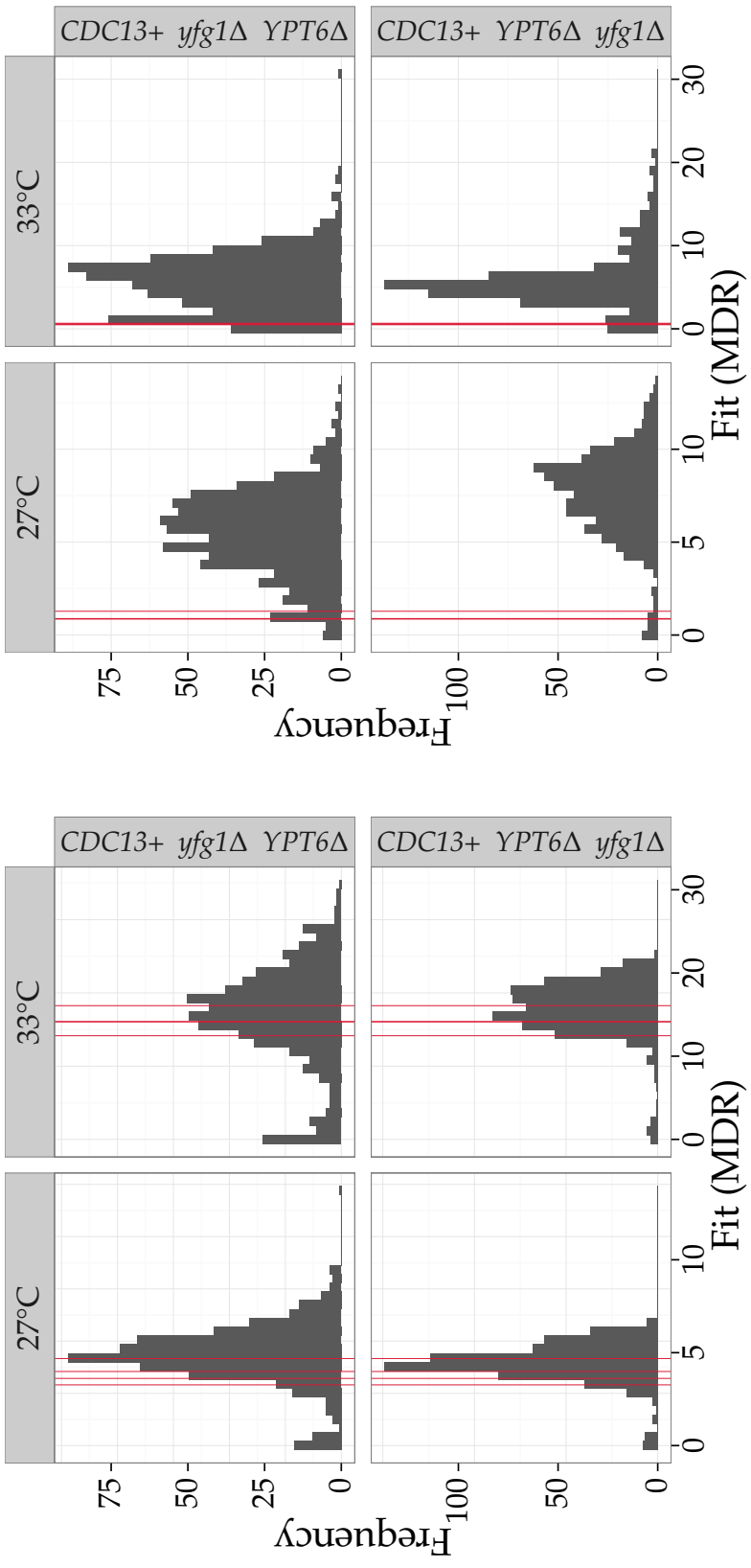
where Z^b and Z^q are ORF fitness for the background and query deletions, respectively.

Early datasets also showed some cases where fitness of the double-l deletion strains had higher than expected fitness values. In some cases, this was an indication that the screen had not worked correctly and would therefore need to be stripped from the experiment. In other cases, it revealed that the double-l deletion culture was actually dead, but neighbouring cultures that were very fit would overlap into the empty space left by the double-l deletion, causing the image analysis to believe the dead strain was growing. Recent refinements in Colonyzer's image analysis by Lawless et al. (2010) has proven more successful at correcting for this, as shown in Figure 6.3, where the old methods in Figure 6.3a picked up artificially high fitnesses for double-l deletions, but newer results in Figure 6.3b show this issue has now been corrected..

This highlights some of the uncertainty there can be in the quality of the data, and the problems presented in deciding whether these quirks in the data should be handled by the Bayesian model, the image analysis and logistic growth-curve fitting, or at the laboratory stage. The histograms of fitness values, such as those in Figures 6.2 and 6.3, are useful for identifying which genes should be investigated for potential experimental error; this can be where the distributions are different depending on whether the same gene is deleted from the background or query, or whether there are several double-l deletion strains with unusually high fitness values.

6.3 Implementation and analysis

We can implement the model and its variants described in Sections 6.2 and 6.2.3 using JAGS (Plummer, 2003), as this provides the simplest way to explore the multiple model variations to be considered. As there are 4 independent experiments



(a) *CDC13+ yfg1Δ YPT6Δ* and *CDC13+ YPT6Δ yfg1Δ* under an early method of analysis.

(b) *CDC13+ yfg1Δ YPT6Δ* and *CDC13+ YPT6Δ yfg1Δ* under a newer version of analysis.

Figure 6.3: Fitness values for strains containing *ypt6Δ* under an earlier and more recent analysis. The fitness values of double-*l* deletions, *CDC13+ ypt6Δ ypt6Δ*, are marked by red vertical lines.

to analyse for each telomere and temperature, we can run each analysis in parallel to make use of multi-core processors or computing clusters.

Scripts are used to create batches of jobs, making it more manageable to initiate all the independent experiments for each of the different model variations. It is generally best for the user to manually check that the chains have converged to their posterior distributions and are mixing well, although methods suggested by Raftery and Lewis (1992) could be used to identify the amount of iterations to be discarded as the burn in and the amount of thinning to be applied to the remaining results. This provides scope for the chain from each experiment to be checked automatically as part of the Mini QFA analysis workflow.

Once the chains are manually checked, further scripts are used to calculate genetic interaction strengths for each of the pairwise *orf*Δ combinations for each experiment. With over $23,000$ possible combinations of deleted genes, an intuitive way of viewing the results from the model's analysis is required. We have created a HTML web page which finds and displays output from the Mini QFA analysis scripts, and places these in the relevant parts of a template. This template allows for results from each experiment to be viewed, for each of the telomere defects used at the different temperatures, along with variations of the models explored.

A searchable and sortable data table of key numerical summaries can be created using `DT` (Jardine, 2014), listing information such as the genetic interaction strength, predicted and actual fitness measurements, and whether each pairwise deletion is a phenotypic suppressor or enhancer.

To supplement this, a genetic interaction plot in PDF format is included displaying how actual observed fitness values deviate from their predicted fitness under the assumption of no interaction occurring (i.e. $\gamma_{ll'} = 0$), highlighting terms which deviate from this by a significant amount. Each interaction is labelled with the concatenation of the names from the deleted genes, allowing the PDF to be searched for deletions of interest. There is additional scope for using `plotly` (Sievert et al., 2016) to provide an interactive version of these genetic interaction plots, which may prove easier to explore due to the large quantity of points and labels featured in the plot.

An example of a genetic interaction plot can be found in Figure 6.4. For the horizontal axis of these plots, we compute the 'predicted fitness' value under the assumption of Fisher's multiplicative model of epistasis based on what is expected when each gene is deleted; this is the fitted value assuming no interaction occurs

(i.e. $\gamma_{ll'}$). For the vertical axis, we have the fitted value of the fitness measure for each pairwise deletion; this is equivalent to adding the genetic interaction strength, $\gamma_{ll'}$, back into the predicted 'fitness assuming no interactions' measure. Most pairs of deletions do not interact, which is expected, and these lie on or sufficiently close along a 1:1 gradient. Deviations above this line suggest that the telomere defect is suppressed by the deletion of both ORF, performing better than predicted. Conversely, deviations below the line suggest a telomere defect is enhanced if the actual fitness was worse than predicted.

Purple points denote where a double-l deletion has occurred. As we expect these to always be dead, we expect to find their predicted 'fitness including interactions' values to be minimal, meaning all these points should appear along the bottom. This can be used as a form of quality control for indicating if the results show any issues with any of the strains.

6.4 Adjustments from collaborator feedback

Providing the tables and plots of genetic interactions allows our collaborators to identify problematic screens and provide feedback on what model variations are showing the most promising ability to identify interactions correctly, based on findings from smaller scale analyses that have previously been performed.

6.4.1 Identifying problematic data

Our collaborators believed they had the greatest understanding of what patterns to expect in the cdc13-1 at 33°C experiment, so they requested for side-by-side plots of results from the various models for each of the possible fitness measures in the cdc13-1 at 33°C experiment. From these plots, they found the models had identified a number of unexpectedly strong hits involving the POT1 gene where the doubling rate parameter, r, was used as the fitness measure—an original plot which exhibited this behaviour is featured in Figure C.3 of Appendix C.2. On further investigation, it emerged the metadata for POT1 had all dates offset by one day, leading to inflated growth rate parameters being provided to the interaction model. After re-performing the analysis with corrected POT1 data, the interaction model also highlighted similar issues with the DPH5 screen, which was subsequently removed from the dataset.

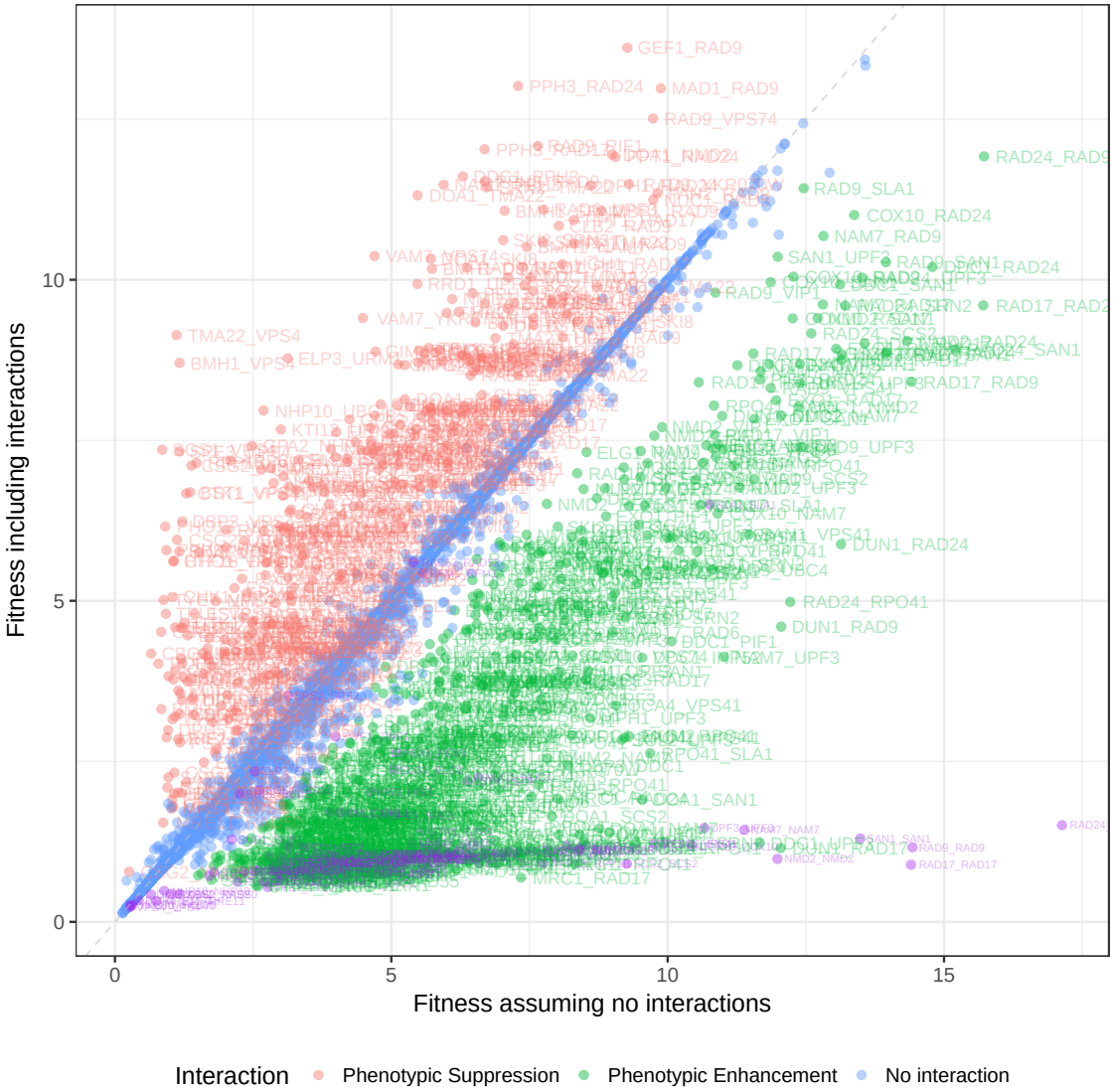


Figure 6.4: Example GIS plot for Mini QFA data for cdc13-1 at 27°C using the MDR measure of fitness. Each interacting gene combinations is labelled by its concatenated name. Purple points represent the double-*l* deletions which are always set to be interacting. The dashed line follows a 1:1 line of fitness.

Under these circumstances, the interaction model proved useful in highlighting problematic subsets of the data to our collaborators. When supplied with unusually high fitness measurements, the interaction model remained robust enough to avoid predicting correspondingly high fitness values, opting instead to highlight these as strong interactions. Unfortunately, the model is not be robust enough to reliably infer all the remaining interactions accurately as the very strong false-interactions from the bad data cause many of the weaker interactions to be considered neutral.

To produce a more robust model in the face of potentially incorrect data, fitness measurements were modelled using a t_3-distribution, in the hope that the heavier tails of the distribution would be more robust to measurement errors. When this model was fit to data containing the original incorrect POT1 and DPH5 values, these genes were not highlighted as strong interactions; it is robust enough to continue identifying other interactions, but does not highlight the problematic screens that needed correcting. This can be seen in Figure C.4 of Appendix C.2.

6.4.2 Selecting from model variations

A number of model variations are mentioned in Section 6.2.3 and we wish to identify a single model that is best for identifying genetic interactions. Producing side-by-side plots of the results from each of the model variations proved to be a useful in identifying which models work best.

As problematic data was corrected or removed from the datasets provided to the interaction models, we found that the results from the different models converged to produce similar results, as seen in Figure C.5 of Appendix C.2. Modelling the fitness measurements using Gaussian or t_3-distributions often highlighted the greatest differences when problematic data was present. As these models have now converged to give similar results, this might suggest that most, if not all, problematic data has been successfully removed. We may opt for the most convenient model to use; this may arguably be the model which applies Gaussian distributions to both the fitness measurements and the ORF fitnesses, since this is quickest to perform due to its latent Gaussian structure.

Models with asymmetric ORF fitnesses were also considered. As problematic data was identified and corrected, we found that most of the symmetry has returned to the ORF fitnesses. This was evident in plots where a pairwise dele-

tion (e.g. cdc13-1 *yfg1*Δ *xyz2*Δ) would appear near its complement deletion (e.g. cdc13-1 *xyz2*Δ *yfg1*Δ). In light of this improvement, a model containing symmetric ORF fitnesses could be used, which would fall inline with our collaborator's expectation of the underlying genetic processes.

Chapter 7

Conclusions and future work

7.1 Conclusions

The aim of this thesis was to explore the use of methods which can aid in more efficient Bayesian inference on latent Gaussian models, including those containing an aspect of Bayesian variable selection. DAG representations of latent Gaussian models highlight the often-sparse dependence structure that appears between variables. By exploiting the conditional independence seen between many of the latent variables, we are able to represent the model using a multivariate Gaussian distribution with a sparse precision matrix. This opens up the possibility of using blocking methods to more efficiently sample from a correlated posterior distribution (Roberts and Sahu, 1997; Amit and Grenander, 1991; Seewald, 1992) since the sparse matrix operations required to do this are feasible, even in large dimension problems.

In the special case of linear Gaussian models, where observations are modelled as Gaussian variables with mean equal to a linear combination of the latent variables, we are able to simplify the calculation of the posterior distribution through use of a canonical parameterisation of the multivariate Gaussian distribution, which Wilkinson and Yeung (2004) demonstrate by constructing a linear Gaussian model from its DAG representation. `GDAGsim` software (Wilkinson, 2002) assists in the construction of linear Gaussian models based on its DAG representation; we created a new version of this in Java, `GDADGsimJ`, built using Parallel Colt (Wendykier and Nagy, 2010) which, crucially, handles AMD permutations of the precision matrix before Cholesky factors are taken and allows very large matrix

operations to be performed in parallel. This therefore extends the original `GDAGsim` implementation in two important ways: fill-reduction and parallelisation. These both facilitate applications to larger and more challenging problems. The new Java version of `GDAGsimJ` is to be publicly released on GitHub once fully tested and documented.

For latent Gaussian models which feature more general likelihoods, `GDAGsimJ` can be used for construction of the prior distribution of the model, but an alternative way of conditioning on the observations is required. Gaussian approximations, also referred to as a GMRF approximation by Rue and Held (2005), provide a method for conditioning on the latent Gaussian model, supplying an approximate multivariate Gaussian density with optimised mean and precision parameters. We explored using a combination of `GDAGsimJ` with Gaussian approximations as part of a blocking scheme for an example in Chapter 4. For this scheme, the model was sufficiently small that standard Gibbs methods from JAGS did not struggle too much with the correlation between variables. An approximate marginal scheme was able to generate independent samples from the posterior with greatest efficiently in terms of ESS/s, but only performs approximate inference and doesn't automatically produce any inference for latent variables without employing a collapsed Gibbs Sampler (Liu, 1994). On larger dimension models, the single- and two-block samplers could not cope with the dimension of the latent field it was attempting to sample from.

Integrated nested Laplace approximations are a deterministic algorithm proposed by Rue et al. (2009), which allows previously intractable integrals to be computed using Laplace approximations. We recreated this method for use on the same toy model and found it to produce close approximations to the exact posterior density targeted by an MCMC scheme, although we could not match the fast processing times achieved in the highly developed and refined `INLA` software. Use of INLA is restricted to low-parameter-dimension problems, as the integration of the parameter space suffers as dimension increases. For this reason, it does not work well for Bayesian variable selection problems as the number of possible indicator states to integrate over can be of a prohibitively high dimension, but some of the methods used in INLA are shared with some of the MCMC blocking methods used on latent Gaussian models.

In Chapter 5, we examined the effect of blocking methods for Bayesian variable selection models by applying them to a genome-wide genetics study to identify

genes that interact with defective telomeres, developing on previous studies by Addinall et al. (2011) and Heydari et al. (2016). The work was focused on producing a more efficient sampling method at the potential cost of a slight decrease in the model's ability to identify interactions. On a linear Gaussian model, we investigated how efficiency improved where we employed a number of tricks to try and reduce computational time or improve mixing. We found: permutations on precision matrices can be essential to reduce Cholesky fill-in; dynamic resizing of the model gives reasonable speed-ups where variables are often likely to be excluded; choice of parameterisation can influence the mixing of a scheme. In comparison with results obtained by JAGS, the blocking methods in one case were over seven times more efficient in terms of the minimum ESS/s of any variable in the chain.

Applying methods for the latent Gaussian model on the same genome-wide dataset was problematic, as the high-dimension of the latent field creates a very low acceptance probability for accepting new values of the latent variables. On a smaller model, using only data from Plate 15, the blocking scheme manages to perform inference successfully being over 14 times more efficient by ESS/s than what is obtained from JAGS. However, this positive performance rapidly deteriorates as the dimension increases.

Finally, in Chapter 6 we discussed details from an ongoing experiment, detailing the considerations made to build an new model for detecting genetic epistasis based on the fitness data provided by image analysis and growth-curve fitting data.

7.2 Future work

7.2.1 Permutations for dynamically resizing models

In Section 2.3.2, we discussed how the AMD algorithm permutes a matrix to minimise Cholesky fill-in. Then we see the effects of these permutations in Section 5.6.2, and while fixing the permutation here does not cause a reduction in computation time, it is possible that larger or more complicated models will benefit from a fixed permutation. Then Section 5.6.3 demonstrated that dynamically resizing the model at each iteration does reduce the CPU user time of the scheme. Since the contents of the precision matrix was changing at each iteration, we al-

lowed the permutation to be recomputed from scratch each time.

What has not yet been explored is whether a permutation can be fixed at the start, while allowing the model to resize. Recall that in Section 2.3.2, we described three key properties that the permutation aims for: densest final row/column, low bandwidth and empty blocks. Suppose you begin with a sparse matrix, $\mathbf{A}$, representing a full model which is permuted by the AMD algorithm into $\mathbf{C} = \mathbf{PAP}^T$, and that each row/column i in $\mathbf{A}$ will have a bijective mapping to row/column j in $\mathbf{C}$, according to the permutation matrix $\mathbf{P}$. We conjecture that once an optimum permutation for the full model is obtained, any row/column i can be removed can be removed from $\mathbf{A}$, which will leave $\mathbf{C}$ approximately close to its optimal layout.

This is in fact because removing any row/column j from $\mathbf{C}$ will not damage the three desirable properties for an *approximately* optimal permutation: we can't cause a dense row/column to appear further from the bottom-right of a matrix by deleting any row/column; we can't increase the maximum bandwidth by removing any row/column; while we can reduce the size of an empty block, we can't contaminate an empty block with a non-zero value by removing any row/column. Since any j can be removed from $\mathbf{C}$, it follows that any i can be removed can be removed from $\mathbf{A}$. By further extension, multiple rows and columns can be deleted at once.

7.2.2 Multi-block methods for latent Gaussian models in QFA

Where performing a latent Gaussian model on a full-size dataset proved unsuccessful due to the large dimension of the latent field being sampled, workarounds can be investigated. One possible solution to the problem is to divide the latent field into multiple smaller blocks, and proposing and accepting each block in turn. Creating smaller blocks should increase the acceptance probability of each, allowing the chain to move more frequently. Increasing the blocks will unfortunately decrease the positive effect blocking has on the mixing, as correlation can exist between the blocks, but this is favourable to a scheme that rarely moves at all. A suitable way of dividing the QFA dataset into smaller blocks may be to perform one block for each of the 15 plates the experiment is performed on. A paper is in preparation awaiting results from the this multi-block approach before it is to be submitted for publication.

7.2.3 Developments on Mini QFA

Progress has been made in exploring which model should be used out of the model variations that were considered, but since analysis of the Mini QFA experiment is an ongoing project, this has not yet been finalised. Feedback and refinement is still being performed in collaboration with the experimentalists, who found the use of different fitness measures useful at tracking-down problematic datasets and spotting interactions. For this reason, they have proposed that three additional fitness measures be considered in the analysis: the carrying capacity parameter from the logistic growth model, K; the product of rate and capacity parameters, rK; a parametric area under the growth curve measurement, AUC. Once the model has been finalised, utilising methods for blocking could be explored to investigate if the inference can be performed in a more efficient manner.

Use of clustering methods can also be investigated, with an aim to identify groups of variables that feature in pairwise deletions. Genetic interaction strengths can also be uploaded to GeneMANIA (Warde-Farley et al., 2010) to assist in predicting clusters of genes that interact and guess at their genetic functions.

The scripts used to automatically create and run the JAGS models and process results afterwards were designed to be easy to implement as a package for R (R Core Team, 2016), which is likely to be released on R-Forge once further refinements have been made. Potential additions to the work include automatically checking the model has converged and is sufficiently thinned to remove autocorrelation, although it could be argued that this stage should contain an element of human intervention to double-check the results.

Current Mini QFA analysis provides point estimates for posterior measurements. For example, the posterior genetic interaction terms, $\delta_{ll'}\gamma_{ll'}$, are given as the posterior mean. It would be beneficial to include a measure of uncertainty around this estimate, especially since the lengthy and complicated process of experimental data collection creates a number of opportunities for error to enter into the model. Probability intervals could easily be included in the searchable results tables. Unfortunately, the large number of overlapping points on the genetic interaction plots would make it impractical to also show probability intervals around each point, unless interactive plots are generated where the user would be able to zoom in on an area of interest in the plot.

Current fitness measures supplied to the interaction models specified in Sec-

tion 6.2 are point estimates from the `Colonyzer` software. A more recent version of `Colonyzer` implementing Bayesian methods is being trialled which would be able to provide posterior predictive distributions for each of the fitness measures that are later used in interaction model specified in Section 6.2. The interaction model would need to be restructured considerably in order to take advantage of the posterior predictive distributions, but this would allow uncertainty from the growth-curve fitting stages to be propagated through the model to the final identification of interacting pairwise gene deletions. Under such a framework, it may be advantageous to create one large Bayesian model which handles both the inference of growth-curve parameters and genetic interaction identification. This would be analogous to the Joint Hierarchical Model created by Heydari et al. (2016) for the QFA experiment. A potential drawback to creating this model would be the long time currently needed to infer the growth-curve parameters using MCMC, which takes over one month to create a satisfactory sample.

Key findings on the detection of genetic epistasis in the Mini QFA paper are expected to be published following further model refinement.

Part III

Appendix

Appendix A

Algorithms

Algorithm 5: Specification of a general GDAG model in `GDAGsim`.

Data: Values for precisions $\tau_.$, number of roots n_1, number of nodes n_2, model dimension $n_1 + n_2 = n$.

Create GDAG instance of dimension n;

`/* Add roots of latent variables using prior values */`

for $i := 1$ **to** *Number of roots* n_1 **do**

  addRoot(index $= i$, mean $= \mu_i$, precision $= \tau_i$);

end

`/* Condition other latent variables on the roots */`

for $i := n_1 + 1$ **to** *Model dimension* n **do**

  Node := Specified dependencies for Node[i] conditioned on roots;

  addNode(Node, index $= i$, offset $= b_i$, precision $= \tau_i$);

end

if *log-density values are needed* **then**

  Process prior structure;

end

`/* Add observations to the model */`

for $i := 1$ **to** *Number of observations* **do**

  Node := Specified dependencies of observation's node;

  addObservation(Node, precision $= \tau_{\text{observation } i}$);

end

Process GDAG model;

Result: Access to log-likelihood, simulations of latent variables, precision matrix **Q**, Lower Cholesky factor **L**.

Algorithm 6: Calculating a multivariate Gaussian density.

Function *mvnLogDensity* ($\boldsymbol{x}$, $\boldsymbol{\mu}$, $\mathbf{L}$)

Data: Quantile vector $\boldsymbol{x}$ to evaluate the density at, Cholesky factor $\mathbf{L}$ from precision matrix $\mathbf{Q}$, mean vector $\boldsymbol{\mu}$, dimension d of $\mathbf{L}$ and $\boldsymbol{\mu}$.

Result: The log-density of $\mathrm{N}_d(\boldsymbol{x}; \boldsymbol{\mu}, \mathbf{Q}^{-1})$.

$d :=$ dimension($\boldsymbol{\mu}$);

logd $:= -\frac{d}{2}\log(2 \times \pi)$;

/* Now find the $(\boldsymbol{x} - \boldsymbol{\mu})^T \mathbf{Q} (\boldsymbol{x} - \boldsymbol{\mu})$ term */

for $i := 1$ **to** d **do**

  $z[i] := x[i] - \mu[i]$;

end

$v :=$ Mult($\mathbf{L}^T, z$); /* Matrix multiply $\mathbf{L}^T z = v$ */

tot $:= 0$;

for $i := 1$ **to** d **do**

  tot $:=$ tot $+ (v[i] \times v[i])$;

end

logd $:=$ logd $- 0.5 \times$ tot;

/* Calculate the square of the determinant for $\mathbf{Q}$ */

tot $:= 0$; /* Reset the total counter */

for $i := 1$ **to** d **do**

  tot $:=$ tot $+ \log(\mathbf{L}[i, i])$;

end

logd $:=$ logd $+$ tot;

return *logd*;

end

Algorithm 7: Generating a sample from a multivariate Gaussian distribution.

Function *multivariateGaussianSample* ($\boldsymbol{\mu}$, $\mathbf{L}$)

Data: Cholesky factor $\mathbf{L}$ from precision matrix $\mathbf{Q}$, mean vector $\boldsymbol{\mu}$, dimension d of $\mathbf{L}$ and $\boldsymbol{\mu}$.

Result: A sample of values from $\mathrm{N}_d(\boldsymbol{\mu}, \mathbf{Q}^{-1})$.

$d :=$ dimension($\boldsymbol{\mu}$);

for $i := 1$ **to** d **do**

  $z[i] :=$ random sample from $\mathrm{N}(0, 1)$ distribution;

end

$w :=$ backsolve($\mathbf{L}^T, z$); /* Solves $\mathbf{L}^T w = z$ */

for $i := 1$ **to** d **do**

  $x[i] := w[i] + \mu[i]$;

end

return $\boldsymbol{x}$;

end

Appendix B

Additional derivations

B.1 Demonstration that Equation (3.5) has no dependence on x

Section 3.2 contains the statement that the $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$ term on the left-hand side of Equation (3.5) carries no dependence on the latent values $\boldsymbol{x}$, and therefore the right-hand side of the equation also does not depend on $\boldsymbol{x}$, despite this term appearing in the equation. By factorising Equation (3.5) further, it can be shown there is no dependence on $\boldsymbol{x}$:

$$\begin{aligned}
\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) &= \frac{\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})\, \pi(\boldsymbol{y})} \\
\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) &= \frac{\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})\, \pi(\boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})\, \pi(\boldsymbol{y})} \\
\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) &= \frac{\pi(\boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{y})} \\
\pi(\boldsymbol{\theta}, \boldsymbol{y}) &= \pi(\boldsymbol{\theta} \mid \boldsymbol{y})\, \pi(\boldsymbol{y}).
\end{aligned}$$

Therefore, the expression in Equation (3.5) simplifies to a standard statement of conditional probability containing no dependence on $\boldsymbol{x}$.

B.2 Demonstration of Equations (3.12) and (3.13)

The AR(1) process for $\boldsymbol{x}$ in Equation (3.11) can be summarised as,

$$x_i = \phi x_{i-1} + \epsilon_i; \quad |\phi| < 1; \quad \epsilon_i \sim \mathrm{N}(0, \sigma^2); \quad i = 2, \ldots, n.$$

The variance of this process for $\boldsymbol{x}$ can be derived as,

$$\begin{aligned}
\mathrm{Var}(x_i) &= \mathrm{Var}(\phi x_{i-1} + \epsilon_i) \\
\mathrm{Var}(x_i) &= \phi^2 \mathrm{Var}(x_{i-1}) + \mathrm{Var}(\epsilon_i) \\
(1 - \phi^2)\mathrm{Var}(x_i) &= \sigma^2 \qquad [\text{Since } \mathrm{Var}(x_i) = \mathrm{Var}(x_{i-1})] \\
\mathrm{Var}(x_i) &= \frac{\sigma^2}{1 - \phi^2}.
\end{aligned}$$

The covariance between two consecutive values in $\boldsymbol{x}$ is calculated as,

$$\begin{aligned}
\mathrm{Cov}(x_{i-1}, x_i) = \mathrm{Cov}(x_i, x_{i-1}) &= \mathrm{Cov}(\phi x_{i-1} + \epsilon_i, x_{i-1}) \\
&= \phi \mathrm{Cov}(x_{i-1}, x_{i-1}) \\
&= \phi \mathrm{Var}(x_{i-1}) \\
&= \phi \frac{\sigma^2}{1 - \phi^2}.
\end{aligned}$$

This can be extended to find the covariance between two latent values in $\boldsymbol{x}$ which are k values apart, for $k = 0, \ldots, i-1$, since,

$$\begin{aligned}
\mathrm{Cov}(x_{i-k}, x_i) = \mathrm{Cov}(x_i, x_{i-k}) &= \mathrm{Cov}(\phi x_{i-1} + \epsilon_i, x_{i-k}) \\
&= \phi \mathrm{Cov}(x_{i-1}, x_{i-k}) \\
&= \phi \mathrm{Cov}(\phi x_{i-2} + \epsilon_i, x_{i-k}) \\
&= \phi^2 \mathrm{Cov}(x_{i-2}, x_{i-k}) \\
&\quad \vdots \\
&= \phi^k \mathrm{Cov}(x_{i-k}, x_{i-k}) \\
&= \phi^k \mathrm{Var}(x_{i-k}) \\
&= \phi^k \frac{\sigma^2}{1 - \phi^2}.
\end{aligned}$$

Appendix C

Supplementary plots

C.1 Simple example for MCMC

To produce the plots shown in Figure 4.3, long MCMC runs were performed on the simple model shown in Section 4.12.5 of 30000 iterations, each thinned by 50 iterations. ACF and trace plots of these runs are provided in Figures C.1 and C.2.

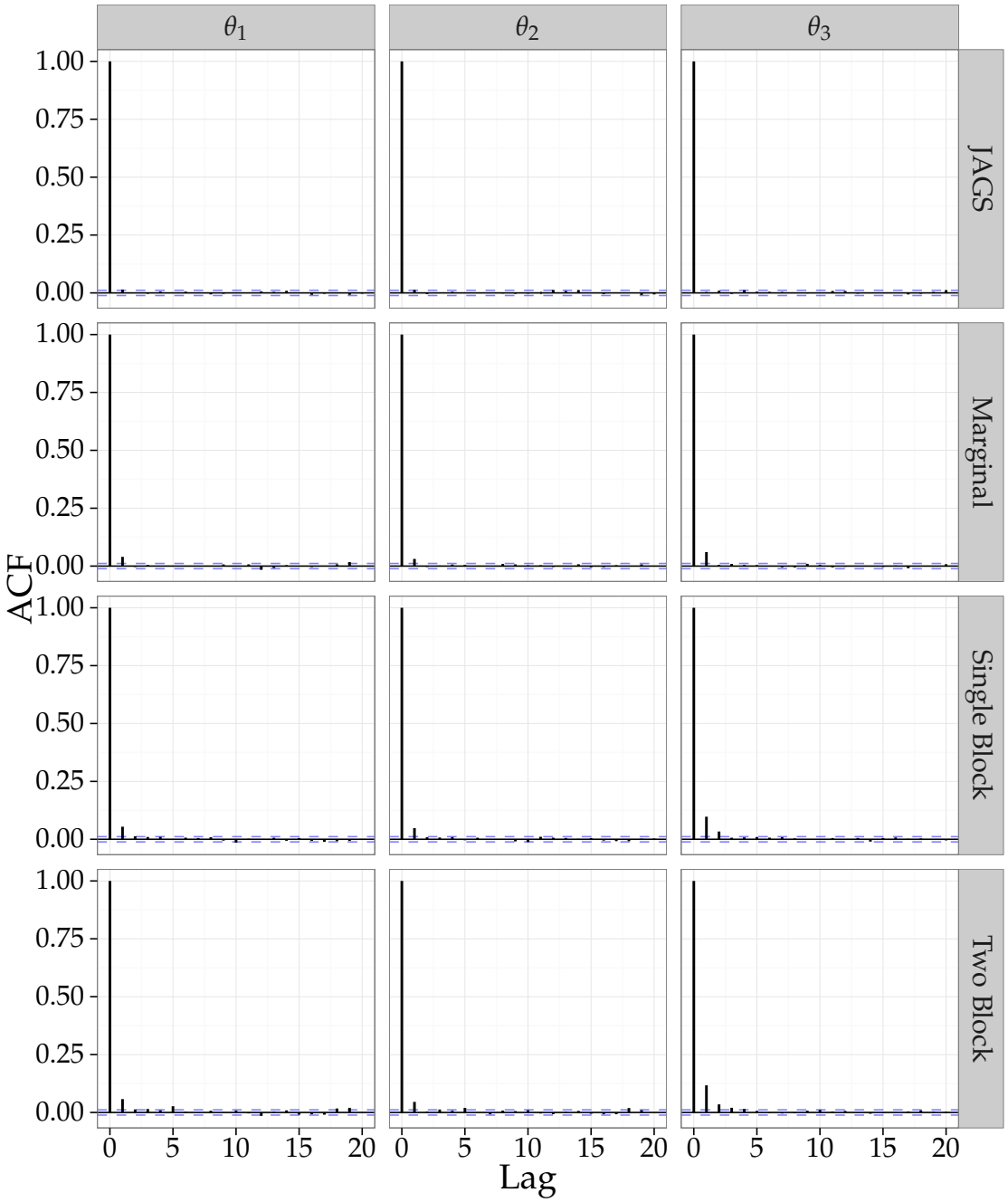


Figure C.1: Autocorrelation plots with corresponding 95% intervals for the 30000 iterations (thinned by 50) shown in Figure 4.1, for the parameters (from left column to right column) θ_1, θ_2 and θ_3, for each of the MCMC schemes used (from top row to bottom row): JAGS, marginal, single-block and two-block methods.

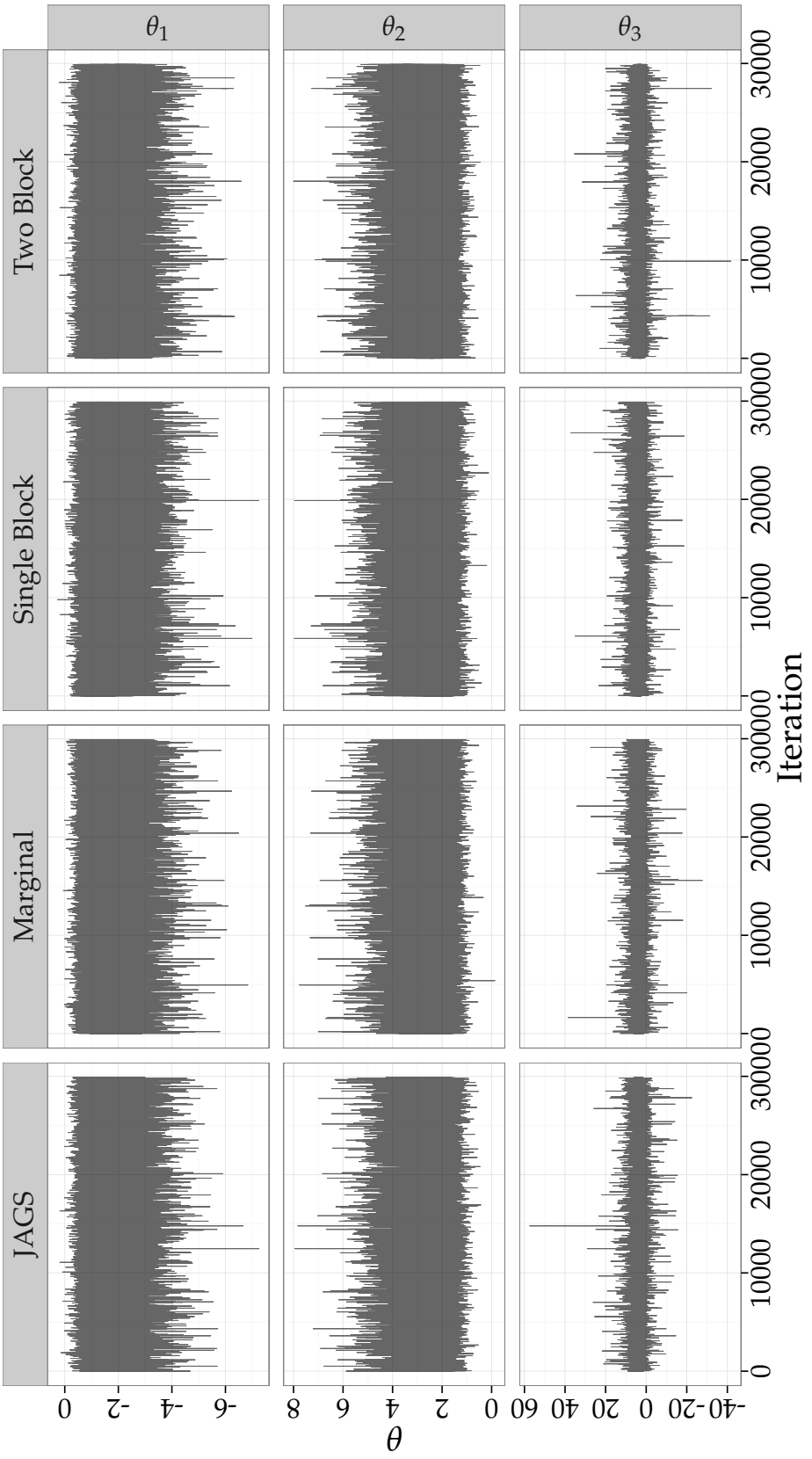


Figure C.2: Trace plots over 30000 iterations (thinned by 50) for the parameters (from top row to bottom row) θ_1, θ_2 and θ_3, for each of the MCMC schemes used (from left column to right column): JAGS, marginal, single-block and two-block methods.

C.2 Highlighting problematic data in Mini QFA

Section 6.4.1 discusses how the experimentalists had the greatest understanding of the CDC13-1 experiment at 33°C from the small scale experiments they performed, so the genetic interaction plots for these experiments provided interesting comparisons to the behaviour which they expected. The unexpectedly strong interactions involving POT1 and DPH5 that appeared in Figure C.3 warranted further investigation and highlighted problematic subsets in the dataset.

Figure C.4 is the analysis on the same problematic data featured in Figure C.3, but where the fitness values are modelled with a t_3-distribution instead of a Gaussian distribution.

Figure C.5 shows results from the same model used to produce Figure C.3, but this uses a more recent dataset where problematic data was corrected or stripped from the analysis.

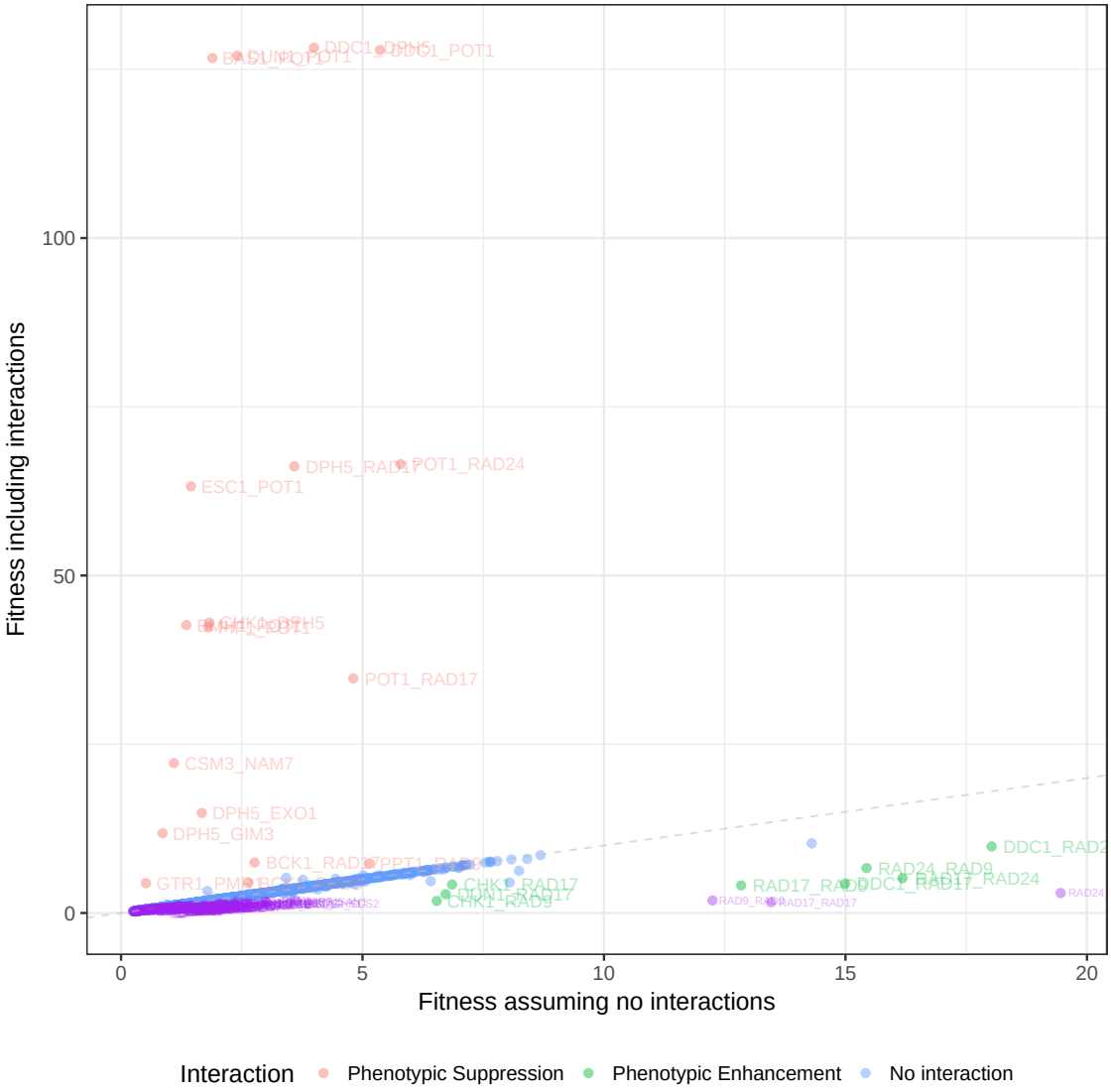


Figure C.3: Genetic interaction strength plot for CDC13-1 at 33°C with a Gaussian distribution used to model the fitness (r) measurements. Problematic POL1 and DPH5 data is included in the analysed dataset. Note that very few interactions are found due to the extreme outliers.

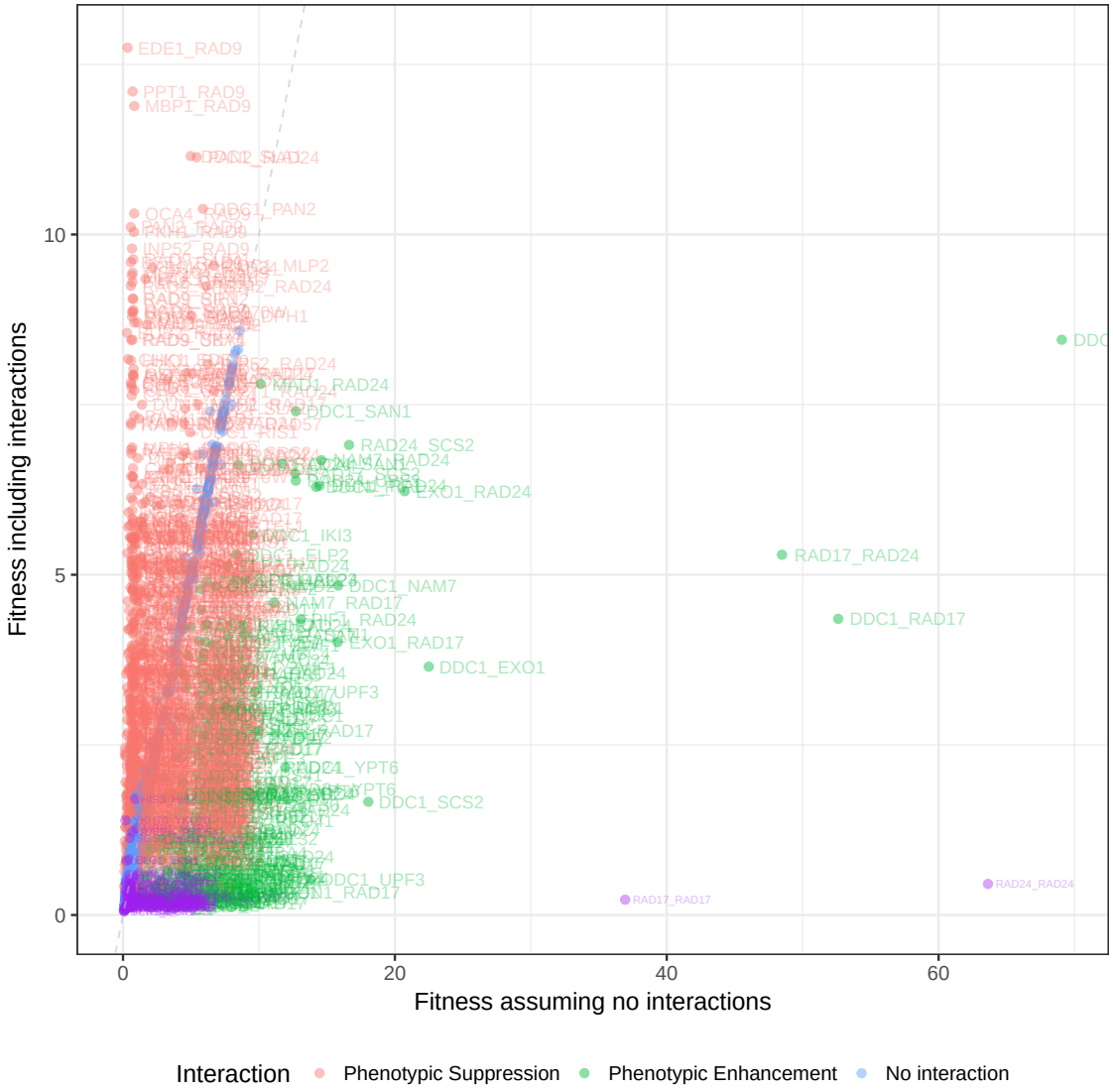


Figure C.4: Genetic interaction strength plot for CDC13-1 at 33°C with a t_3-distribution used to model the fitness (r) measurements. Problematic POL1 and DPH5 data is included in the analysed dataset.

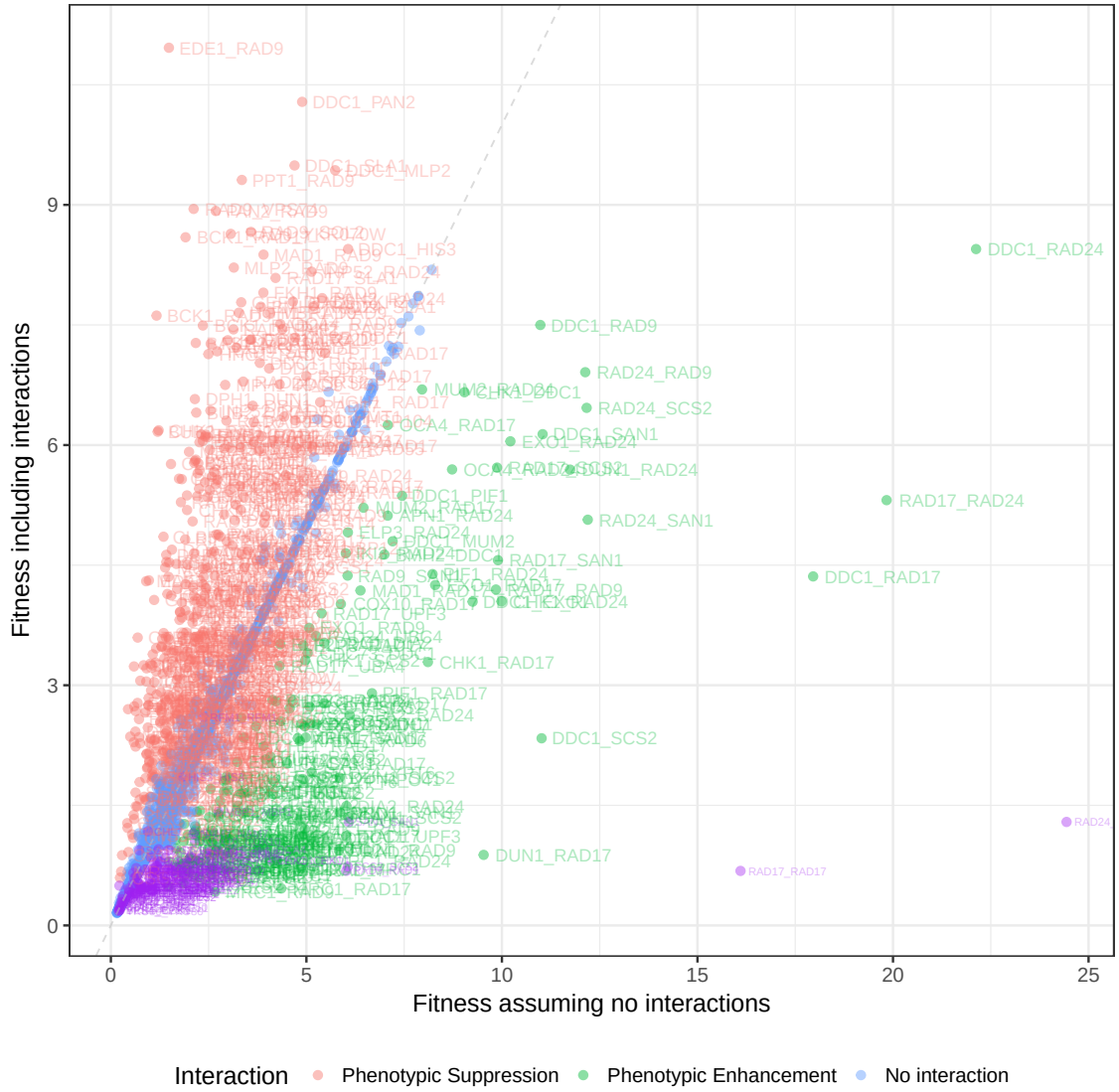


Figure C.5: Genetic interaction strength plot for CDC13-1 at 33°C with a Gaussian distribution used to model the fitness (r) measurements. Problematic datasets have been removed or corrected for this more recent analysis.

Bibliography

Addinall, S. G., Holstein, E.-M., Lawless, C., Yu, M., Chapman, K., Banks, A. P., Ngo, H.-P., Maringele, L., Taschuk, M., Young, A., Ciesiolka, A., Lister, A. L., Wipat, A., Wilkinson, D. J. and Lydall, D. (2011), 'Quantitative Fitness Analysis Shows That NMD Proteins and Many Other Protein Complexes Suppress or Enhance Distinct Telomere Cap Defects', *PLoS Genetics* **7**(4), e1001362.

Amestoy, P. R., Davis, T. A. and Duff, I. S. (1996), 'An Approximate Minimum Degree Ordering Algorithm', *SIAM Journal on Matrix Analysis and Applications* **17**(4), 886–905.

Amestoy, P. R., Davis, T. A. and Duff, I. S. (2004), 'Algorithm 837: AMD, an Approximate Minimum Degree Ordering Algorithm', *ACM Transactions on Mathematical Software* **30**(3), 381–388.

Amit, Y. and Grenander, U. (1991), 'Comparing sweep strategies for stochastic relaxation', *Journal of Multivariate Analysis* **37**(2), 197–222.

Bagchi, R., Gallery, R. E., Gripenberg, S., Gurr, S. J., Narayan, L., Addis, C. E., Freckleton, R. P. and Lewis, O. T. (2014), 'Pathogens and insect herbivores drive rainforest plant diversity and composition', *Nature* **506**(7486), 85–88. Letter.

Bates, D. and Mächler, M. (2014), *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.1-2.

Benoît, C. (1924), 'Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues, (Procédé du Commandant Cholesky)', *Bulletin Géodésique* **2**, 67–77.

Björck, Å. (2014), *Numerical Methods in Matrix Computations*, Texts in Applied Mathematics, Springer International Publishing.

Box, G. E. P. and Muller, M. E. (1958), 'A Note on the Generation of Random Normal Deviates', *The Annals of Mathematical Statistics* **29**(2), 610–611.

Brezinski, C. and Tournès, D. (2014), *André-Louis Cholesky: Mathematician, Topographer and Army Officer*, Springer International Publishing.

Brooks, S. P. (1998), 'Markov chain Monte Carlo method and its application', *Journal of the Royal Statistical Society. Series D (The Statistician)* **47**(1), 69–100.

Broyden, C. G. (1970), 'The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations', *IMA Journal of Applied Mathematics* **6**(1), 76–90.

Buttery, S. M., Kono, K., Stokasimov, E. and Pellman, D. (2012), 'Regulation of the formin Bnr1 by septins and a MARK/Par1-family septin-associated kinase', *Molecular Biology of the Cell* **23**(20), 4041–4053.

Carney, J. P., Maser, R. S., Olivares, H., Davis, E. M., Le Beau, M., Yates III, J. R., Hays, L., Morgan, W. F. and Petrini, J. H. J. (1998), 'The hMre11/hRad50 Protein Complex and Nijmegen Breakage Syndrome: Linkage of Double-Strand Break Repair to the Cellular DNA Damage Response', *Cell* **93**(3), 477–486.

Cartwright, R. A. and Graur, D. (2011), 'The multiple personalities of Watson and Crick strands', *Biology Direct* **6**, 7.

Casarin, R., Craiu, R. V. and Leisen, F. (2015), 'Embarrassingly parallel sequential markov-chain monte carlo for large sets of time series'.

Cawthon, R. M., Smith, K. R., O'Brien, E., Sivatchenko, A. and Kerber, R. A. (2003), 'Association between telomere length in blood and mortality in people aged 60 years or older', *The Lancet* **361**(9355), 393–395.

Cherry, J. M., Hong, E. L., Amundsen, C., Balakrishnan, R., Binkley, G., Chan, E. T., Christie, K. R., Costanzo, M. C., Dwight, S. S., Engel, S. R., Fisk, D. G., Hirschman, J. E., Hitz, B. C., Karra, K., Krieger, C. J., Miyasato, S. R., Nash, R. S., Park, J., Skrzypek, M. S., Simison, M., Weng, S. and Wong, E. D. (2012), 'Saccharomyces Genome Database: the genomics resource of budding yeast', *Nucleic Acids Research* **40**(Database issue), D700–D705.

Chib, S. and Ramamurthy, S. (2010), 'Tailored randomized block MCMC methods with application to DSGE models', *Journal of Econometrics* **155**(1), 19–38.

Coppersmith, D. and Winograd, S. (1990), 'Matrix Multiplication via Arithmetic Progressions', *Journal of Symbolic Computation* **9**(3), 251–280. Computational algebraic complexity editorial.

Cordell, H. J. (2002), 'Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans', *Human Molecular Genetics* **11**(20), 2463–2468.

D'Amours, D. and Jackson, S. P. (2002), 'The MRE11 complex: at the crossroads of DNA repair and checkpoint signalling', *Nature Reviews Molecular Cell Biology* **3**(5), 317–327.

Davis, T. A. (2006), *Direct methods for sparse linear systems*, SIAM.

Dellaportas, P., Forster, J. J. and Ntzoufras, I. (2002), 'On Bayesian model and variable selection using MCMC', *Statistics and Computing* **12**(1), 27–36.

Flegal, J. M., Hughes, J. and Vats, D. (2016), *mcmcse: Monte Carlo Standard Errors for MCMC*, Riverside, CA and Minneapolis, MN. R package version 1.2-1.

Fletcher, R. (1970), 'A new approach to variable metric algorithms', *The Computer Journal* **13**(3), 317–322.

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., Rossi, F. and Ulerich, R. (2009), *GNU Scientific Library Reference Manual*, third edn, Network Theory Ltd.

Gamerman, D. and Lopes, H. (2006), *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, Chapman & Hall/CRC Texts in Statistical Science, second edn, Taylor & Francis.

Gelfand, A. E. and Smith, A. F. M. (1990), 'Sampling-Based Approaches to Calculating Marginal Densities', *Journal of the American Statistical Association* **85**(410), 398–409.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A. and Rubin, D. (2013), *Bayesian Data Analysis*, Chapman & Hall/CRC Texts in Statistical Science, third edn, CRC Press.

Geman, S. and Geman, D. (1984), 'Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**(6), 721–741.

George, A., Heath, M. T. and Liu, J. (1986), 'Parallel Cholesky factorization on a shared-memory multiprocessor', *Linear Algebra and its Applications* **77**, 165–187.

George, A. and Liu, J. W. H. (1989), 'The evolution of the minimum degree ordering algorithm', *SIAM Review* **31**(1), 1–19.

Geyer, C. J. (1992), 'Practical Markov Chain Monte Carlo', *Statistical Science* **7**(4), 473–483.

Gilks, W. R., Richardson, S. and Spiegelhalter, D. (1995), *Markov Chain Monte Carlo in Practice*, Chapman & Hall/CRC Interdisciplinary Statistics, Taylor & Francis.

Goffeau, A., Barrell, B. G., Bussey, H., Davis, R. W., Dujon, B., Feldmann, H., Galibert, F., Hoheisel, J. D., Jacq, C., Johnston, M., Louis, E. J., Mewes, H. W., Murakami, Y., Philippsen, P., Tettelin, H. and Oliver, S. G. (1996), 'Life with 6000 Genes', *Science* **274**(5287), 546–567.

Goldfarb, D. (1970), 'A Family of Variable-Metric Methods Derived by Variational Means', *Mathematics of Computation* **24**(109), 23–26.

Gong, L. and Flegal, J. M. (2016), 'A Practical Sequential Stopping Rule for High-Dimensional Markov Chain Monte Carlo', *Journal of Computational and Graphical Statistics* **25**(3), 684–700.

Green, P. J. (1995), 'Reversible jump Markov chain Monte Carlo computation and Bayesian model determination', *Biometrika* **82**(4), 711–732.

Haas, S. E., Hooten, M. B., Rizzo, D. M. and Meentemeyer, R. K. (2011), 'Forest species diversity reduces disease risk in a generalist plant pathogen invasion', *Ecology Letters* **14**(11), 1108–1116.

Hajat, A., Diez-Roux, A., Adar, S. D., Auchincloss, A. H., Lovasi, G. S., O'Neill, Marie, S., Sheppard, L. and Kaufman, J. D. (2013), 'Air pollution and individual and neighborhood socioeconomic status: Evidence from the multi-ethnic study of atherosclerosis (mesa)', *Environmental Health Perspectives* **121**(11-12), 1325.

Hastings, W. K. (1970), 'Monte Carlo sampling methods using Markov chains and their applications', *Biometrika* **57**(1), 97–109.

Heydari, J., Lawless, C., Lydall, D. A. and Wilkinson, D. J. (2016), 'Bayesian hierarchical modelling for inferring genetic interactions in yeast', *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **65**(3), 367–393.

Jardine, A. (2014), *DataTables*. DT version 1.10.2.

Joshi, M., Karypis, G., Kumar, V., Gupta, A. and Gustavson, F. (1999), PSPASES: An Efficient and Scalable Parallel Sparse Direct Solver, *in* 'In proceedings of the ninth SIAM conference on parallel processing for scientific computing'.

Kass, R. E., Carlin, B. P., Gelman, A. and Neal, R. M. (1998), 'Markov Chain Monte Carlo in Practice: A Roundtable Discussion', *The American Statistician* **52**(2), 93–100.

Knorr-Held, L. and Rue, H. (2002), 'On Block Updating in Markov Random Field Models for Disease Mapping', *Scandinavian Journal of Statistics* **29**(4), 597–614.

Kuo, L. and Mallick, B. (1998), 'Variable Selection for Regression Models', *Sankhyā: The Indian Journal of Statistics, Series B (1960–2002)* **60**(1), 65–81.

Lauritzen, S. L. (1992), 'Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models', *Journal of the American Statistical Association* **87**(420), 1098–1108.

Lawless, C., Wilkinson, D. J., Young, A., Addinall, S. G. and Lydall, D. A. (2010), 'Colonyzer: automated quantification of micro-organism growth characteristics on solid agar', *BMC Bioinformatics* **11**(1), 1–12.

Link, W. A. and Eaton, M. J. (2012), 'On thinning of chains in MCMC', *Methods in Ecology and Evolution* **3**(1), 112–115.

Liu, J. S. (1994), 'The Collapsed Gibbs Sampler in Bayesian Computations with Applications to a Gene Regulation Problem', *Journal of the American Statistical Association* **89**(427), 958–966.

Lydall, D. (2009), 'Taming the tiger by the tail: modulation of DNA damage responses by telomeres', *The EMBO Journal* **28**(15), 2174–2187.

MacEachern, S. N. and Berliner, L. M. (1994), 'Subsampling the Gibbs Sampler', *The American Statistician* **48**(3), 188–190.

Mani, R., St.Onge, R. P., Hartman, J. L., Giaever, G. and Roth, F. P. (2008), 'Defining genetic interaction', *Proceedings of the National Academy of Sciences* **105**(9), 3461–3466.

Martins, T. G., Simpson, D., Lindgren, F. and Rue, H. (2013), 'Bayesian computing with INLA: New features', *Computational Statistics & Data Analysis* **67**, 68–83.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953), 'Equation of State Calculations by Fast Computing Machines', *The Journal of Chemical Physics* **21**(6), 1087–1092.

Murphy, K. P. (2012), *Machine learning: a probabilistic perspective*, MIT press.

Neiswanger, W., Wang, C. and Xing, E. (2013), 'Asymptotically exact, embarrassingly parallel mcmc'.

O'Hara, R. B. and Sillanpää, M. J. (2009), 'A Review of Bayesian Variable Selection Methods: What, How and Which', *Bayesian Analysis* **4**(1), 85–117.

Pagon, R. A., Adam, M. P., Ardinger, H. H. et al. (2017), 'GeneReviews Glossary'. [Online; accessed 27 January 2017].
URL: *https://www.ncbi.nlm.nih.gov/books/NBK5191/?report=classic*

Plummer, M. (2003), JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling, *in* K. Hornik, F. Leisch and A. Zeileis, eds, 'Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)', Vienna, Austria.

Plummer, M., Best, N., Cowles, K. and Vines, K. (2006), 'CODA: Convergence Diagnosis and Output Analysis for MCMC', *R News* **6**(1), 7–11.

Poggio, L., Gimona, A., Spezia, L. and Brewer, M. J. (2016), 'Bayesian spatial modelling of soil properties and their uncertainty: The example of soil organic matter in Scotland using R-INLA', *Geoderma* **277**, 69–82.

R Core Team (2016), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Raftery, A. E. and Lewis, S. M. (1992), '[Practical Markov Chain Monte Carlo]: Comment: One Long Run with Diagnostics: Implementation Strategies for Markov Chain Monte Carlo', *Statistical Science* **7**(4), 493–497.

Ripley, B. D. (1987), *Stochastic Simulation*, John Wiley & Sons, Inc., New York, NY, USA.

Roberts, G. O., Gelman, A. and Gilks, W. R. (1997), 'Weak convergence and optimal scaling of random walk Metropolis algorithms', *The Annals of Applied Probability* **7**(1), 110–120.

Roberts, G. O. and Sahu, S. K. (1997), 'Updating Schemes, Correlation Structure, Blocking and Parameterization for the Gibbs Sampler', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **59**(2), 291–317.

Rue, H. (2001), 'Fast sampling of Gaussian Markov random fields', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **63**(2), 325–338.

Rue, H. and Held, L. (2005), *Gaussian Markov Random Fields: Theory and Applications*, Vol. 104 of *Monographs on Statistics and Applied Probability*, Chapman & Hall, London.

Rue, H., Martino, S. and Chopin, N. (2009), 'Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **71**(2), 319–392.

Schrödle, B., Held, L., Riebler, A. and Danuser, J. (2011), 'Using integrated nested Laplace approximations for the evaluation of veterinary surveillance data from Switzerland: a case-study', *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **60**(2), 261–279.

Seewald, W. (1992), Discussion on Parameterization issues in Bayesian inference (by S. E. Hills and A. F. M. Smith), *in* J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, eds, 'Bayesian Statistics 4', Oxford University Press, Oxford, pp. 241–243.

Shanno, D. F. (1970), 'Conditioning of Quasi-Newton Methods for Function Minimization', *Mathematics of Computation* **24**(111), 647–656.

Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M. and Despouy, P. (2016), *plotly: Create Interactive Web Graphics via 'plotly.js'*. R package version 4.5.6.

Smith, A. F. M., Skene, A. M., Shaw, J. E. H. and Naylor, J. C. (1987), 'Progress with Numerical and Graphical Methods for Practical Bayesian Statistics', *Journal of the Royal Statistical Society. Series D (The Statistician)* **36**(2/3), 75–82.

Stewart, D. E. and Leyk, Z. (1994), *Meschach: Matrix Computations in C : Version 1.2*, Proceedings of the Centre for Mathematics and Its Applications, Australian National University, Centre for Mathematics and its Applications, Australian National University, School of Mathematical Sciences.

Tanner, M. A. and Wong, W. H. (1987), 'The Calculation of Posterior Distributions by Data Augmentation', *Journal of the American Statistical Association* **82**(398), 528–540.

Tierney, L. (1994), 'Markov Chains for Exploring Posterior Distributions', *The Annals of Statistics* **22**(4), 1701–1728.

Tong, A. H. Y. and Boone, C. (2006), Synthetic Genetic Array Analysis in Saccharomyces cerevisiae, *in* W. Xiao, ed., 'Yeast Protocol', second edn, Vol. 313 of *Methods in molecular biology*, Humana Press, Totowa, New Jersey, chapter 17, pp. 171–191.

Tong, A. H. Y., Evangelista, M., Parsons, A. B., Xu, H., Bader, G. D., Pagé, N., Robinson, M., Raghibizadeh, S., Hogue, C. W. V., Bussey, H., Andrews, B., Tyers, M. and Boone, C. (2001), 'Systematic Genetic Analysis with Ordered Arrays of Yeast Deletion Mutants', *Science* **294**(5550), 2364–2368.

Turing, A. M. (1948), 'Rounding-off errors in matrix processes', *The Quarterly Journal of Mechanics and Applied Mathematics* **1**(1), 287–308.

Vallen, E. A., Caviston, J. and Bi, E. (2000), 'Roles of Hof1p, Bni1p, Bnr1p, and Myo1p in Cytokinesis in Saccharomyces cerevisiae', *Molecular Biology of the Cell* **11**(2), 593–611.

van de Wiel, M. A., Leday, G. G., Pardo, L., Rue, H., van der Vaart, A. W. and van Wieringen, W. N. (2012), 'Bayesian analysis of RNA sequencing data by estimating multiple shrinkage priors', *Biostatistics* .

Warde-Farley, D., Donaldson, S. L., Comes, O., Zuberi, K., Badrawi, R., Chao, P., Franz, M., Grouios, C., Kazi, F., Lopes, C. T., Maitland, A., Mostafavi, S., Montojo, J., Shao, Q., Wright, G., Bader, G. D. and Morris, Q. (2010), 'The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function', *Nucleic Acids Research* **38**(Web Server issue), W214–W220.

Watson, J. D. and Crick, F. H. C. (1953), 'Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid', *Nature* **171**(4356), 737–738.

Wendykier, P. and Nagy, J. G. (2010), 'Parallel Colt: A High-Performance Java Library for Scientific Computing and Image Processing', *ACM Transactions on Mathematical Software* **37**(3), 31:1–31:22.

Whiley, M. and Wilson, S. P. (2004), 'Parallel algorithms for Markov chain Monte Carlo methods in latent spatial Gaussian models', *Statistics and Computing* **14**(3), 171–179.

Wilkinson, D. J. (2002), 'GDAGsim: Sparse matrix algorithms for Bayesian computation'.

Wilkinson, D. J. and Yeung, S. K. H. (2002), 'Conditional simulation from highly structured Gaussian systems, with application to blocking-MCMC for the Bayesian analysis of very large linear models', *Statistics and Computing* **12**(3), 287–300.

Wilkinson, D. J. and Yeung, S. K. H. (2004), 'A sparse matrix approach to Bayesian computation in large linear models', *Computational Statistics & Data Analysis* **44**(3), 493–516.

Yannakakis, M. (1981), 'Computing the Minimum Fill-In is NP-Complete', *SIAM Journal on Algebraic Discrete Methods* **2**(1), 77–79.

Yoon, J. W. and Wilson, S. (2011), The Efficient Gaussian Approximation for a class of Latent Gaussian model, Technical report, Trinity College Dublin.

Yuster, R. and Zwick, U. (2005), 'Fast Sparse Matrix Multiplication', *ACM Transactions on Algorithms* **1**(1), 2–13.