High-dimensional hierarchical models and massively parallel computing

by

William Michael Landau

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Statistics

Program of Study Committee:
Jarad B. Niemi, Major Professor
Dan Nettleton
Peng Liu
Vivekananda Roy
Patrick S. Schnable

Iowa State University

Ames, Iowa

2016

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

This research was supported by National Institute of General Medical Sciences (NIGMS) of the National Institutes of Health and the joint National Science Foundation / NIGMS Mathematical Biology Program under award number R01GM109458. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health or the National Science Foundation.

Drs. Jarad Niemi and Dan Nettleton of Iowa State University oversaw and guided this work at every stage, and their consistent help was absolutely essential.

ABSTRACT

This work expounds a computationally expedient strategy for the fully Bayesian treatment of high-dimensional hierarchical models. Most steps in a Markov chain Monte Carlo routine for such models are either conditionally independent draws or low-dimensional draws based on summary statistics of parameters at higher levels of the hierarchy. We construct both sets of steps using parallelized algorithms designed to take advantage of the immense parallel computing power of general-purpose graphics processing units while avoiding the severe memory transfer bottleneck. We apply our strategy to RNA-sequencing (RNA-seq) data analysis, a multiple-testing, low-sample-size scenario where hierarchical models provide a way to borrow information across genes. Our approach is solidly tractable, and it performs well under several metrics of estimation, posterior inference, and gene detection. Best-case-scenario empirical Bayes counterparts perform equally well, lending support to existing empirical Bayes approaches in RNA-seq. Finally, we attempt to improve the robustness of estimation and inference of our RNA-seq model using alternate hierarchical distributions.

CHAPTER 1. INTRODUCTION

This work leverages state-of-the-art high-performance computing to unlock otherwise intractable methodology, pushing against one of the longest-standing obstacles to the practice of Bayesian Statistics. We begin with the general task of estimating the full joint posterior distribution of the parameters of a hierarchical model using a Markov chain Monte Carlo (MCMC) algorithm with a Gibbs sampling structure. Serially-implemented MCMC is intractable when the model is high-dimensional or cumbersome with respect to the number of hierarchical levels, the number of groups at each level, or the number of observations in each group. Our parallelized MCMC, on the other hand, utilizes parallel computing on a massive scale. Large collections of Gibbs steps for conditionally independent parameters run simultaneously in embarrassingly parallel routines, and for the rest of the parameters, parallelized reductions quickly supply the parameters of the sampling distributions. The algorithm is designed to leverage the parallelism capabilities of a general-purpose graphics processing unit (GPU), and the tradeoff is a memory transfer bottleneck so severe that we cannot use MCMC parameter samples in conventional or direct ways. However, cumulatively-estimated posterior means and mean squares allow for efficient convergence diagnosis and posterior inference. The overall strategy is solidly tractable, as we demonstrate in Chapter 2 with a real next-generation genomic sequencing dataset and a simulation study.

With the statistical and computational frameworks established, we turn to statistical genomics, which stands to benefit from strategies like ours. RNA-sequencing experiments in particular measure the expression levels of tens of thousands of genes using relatively few observations per gene, and the goal is often to detect genes with scientifically important characteristics. In this multiple-testing, low-sample-size scenario, hierarchical models are helpful because they borrow information across genes. In fact, empirical Bayes procedures, which reduce the computation burden by skipping the sampling of the hyperparameters,

are already becoming popular (Hardcastle and Kelly, 2010; Wu et al., 2012; Ji et al., 2014; Niemi et al., 2015). Upgrading to a fully Bayesian approach could theoretically improve inference and gene detection by taking into account the uncertainties in all the parameters, and we examine this possibility in Chapter 3 with RNA-seq simulation studies motivated by a maize experiment by Paschold et al. (2012). For our simulation studies and our GPU-accelerated dual R package implementation, the fully Bayesian approach is equally matched with its best-case-scenario empirical Bayes counterparts in terms of estimation, inference, and gene detection ability, lending support to the use of empirical Bayes methods in statistical genomics when the requisite point estimates of hyperparameters are accurate and precise.

In addition to comparing fully Bayesian and empirical Bayes versions of our application, the simulations in Chapter 3 uncover two major shortcomings of the hierarchical RNA-seq model. First, estimated credible intervals fail to capture important parameters when the true values are extreme or when the model assumptions are violated. Second, estimates of gene-specific posterior probabilities are poorly calibrated when the model disagrees with the data-generating mechanism. In an attempt to rectify these issues, we use scale mixtures of normals to vary the hierarchical distributions of important gene-specific parameters. This technique allows previously normal hierarchical distributions to become Laplace or Student t distributions, chosen because their heavier tails can, in principle, relax the rigidness of a model's behavior. Indeed, the simulation studies in Chapter 4 show that the estimation of extreme-valued model coefficient parameters does improve slightly, but the robustness of parameter estimation and the calibration of posterior probabilities do not.

CHAPTER 2. PARALLELIZED MCMC FOR BAYESIANS

Markov chain Monte Carlo (MCMC) is the predominant tool used in Bayesian parameter estimation for hierarchical models. When the model expands due to an increasing number of hierarchical levels, number of groups at a particular level, or number of observations in each group, a fully Bayesian analysis via MCMC can easily become computationally intractable. We illustrate how the steps in an MCMC for hierarchical models are predominantly one of two types: conditionally independent draws or low-dimensional draws based on summary statistics of parameters at higher levels of the hierarchy. Parallel computing can increase efficiency by performing embarrassingly parallel computations for conditionally independent draws and calculating the summary statistics using parallel reductions. During the MCMC algorithm, we record running means and means of squared parameter values to allow convergence diagnosis and posterior inference while avoiding the costly memory transfer bottleneck. We demonstrate the effectiveness of the algorithm on a model motivated by next generation sequencing data, and we release our implementation in R packages `fbseq` and `fbseqCUDA`.

2.1 Introduction

A two-level hierarchical model has the form,

$$y_g|\mu_g \overset{\text{ind}}{\sim} p(y_g|\mu_g), \qquad \mu_g|\phi \overset{\text{ind}}{\sim} p(\mu_g|\phi) \tag{2.1}$$

where y_g may be a scalar or vector, $y = \{y_1, \ldots, y_G\}$ is the collection observed data, each μ_g may be a scalar or vector, $\mu = \{\mu_1 \cdots \mu_G\}$ is the collection of group-specific parameters, ϕ is the vector of hyperparameters, and $\overset{\text{ind}}{\sim}$ indicates conditional independence. Figure 2.1 displays a directed acyclic graph (DAG) representation of this model. Given a prior $\phi \sim p(\phi)$, our goal is to obtain the full joint posterior density of the parameters, $p(\mu, \phi|y)$. Typically, this posterior is analytically intractable, so approximation techniques are used. Most

commonly, a Markov chain Monte Carlo (MCMC) algorithm such as Metropolis-Hastings, Hamiltonian Monte Carlo, slice sampling, Gibbs sampling, or a combination of these or other techniques are used to obtain samples that converge to draws from this posterior. If G is large, implementations of MCMC algorithms that estimate $p(\mu, \phi|y)$ can be slow or even computationally intractable.

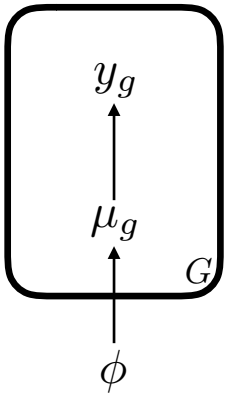

Figure 2.1: Directed acyclic graph (DAG) representation of a two-level hierarchical model. The box with G in the corner indicates nodes μ_g and y_g for $g = 1, \ldots, G$ where the node ϕ has a directed edge to each μ_g and each μ_g has a directed edge to the associated y_g.

The primary motivating context for our work is in the estimation of parameters in a high-dimensional hierarchical model for data from RNA-sequencing (RNA-seq) experiments. RNA-seq experiments measure the expression levels of tens of thousands of genes in a small collection of samples, and they are used to answer important scientific questions in a multitude of fields, such as biology, agriculture, and medicine (Mortazavi et al., 2008; Paschold et al., 2012; Ramskold et al., 2012). Each gene is observed a relatively small number of times, and the task is to detect important genes according to application-specific criteria. A hierarchical model allows data-based borrowing of information across genes and thereby ameliorates difficulties due to the small sample sizes for each gene. Unfortunately, estimation of parameters in these models via general purpose Bayesian software or custom-built serial algorithms is computationally intractable.

The development of parallelized algorithms for Bayesian analysis is an active area of re-

search enhanced by the wide availability of general purpose graphics processing units (GPUs). Suchard and Rambaut (2009) utilized GPU-acceleration for likelihood calculations in phylogenetic models. Lee et al. (2010) described strategies for parallelizing population-based MCMC and sequential Monte Carlo. Suchard et al. (2010) outlined a strategy for applying parallelized MCMC to fit mixture models in Bayesian fashion. Tibbits, MM and Haran, M and Liechty, JC (2011) proposed and implemented parallel multivariate slice sampling. Jacob et al. (2011) built a block independence Metropolis-Hastings algorithm to improve estimators while incurring no additional computational expense due to parallelization. Murray and Adams (2014) used hundreds of cores to accelerate an elliptical slice sampling algorithm that approximates the target density with a mixture of normal densities constructed by sharing information across parallel Markov chains. White and Porter (2014) performed MCMC using GPU-acceleration to calculate likelihoods while modeling terrorist activities. Gramacy et al. (2014) applied multiple high-performance parallel computing paradigms to accelerate Gaussian process regression. Beam et al. (2015) built a GPU-parallelized version of Hamiltonian Monte Carlo in the context of a multinomial regression model. Gruber et al. (2016) utilized GPU-parallelized importance sampling and variational Bayes for estimation and prediction in dynamic models.

We develop a fully Bayesian approach for analyses that use high-dimensional hierarchical models made feasible by the development of efficient parallelized algorithms. Section 2.2 develops a strategy for designing parallel MCMC algorithms for estimating full joint posterior distributions of hierarchical models. Section 2.3 suggests that graphics processing units (GPUs) offer the most appropriate parallel computing platform, and this section explains how to maximize the effectiveness of GPUs. Section 2.4 describes an application of our strategy in the analysis of RNA-seq data along with its implementation, a pair of publicly-available R packages. Finally, Section 2.5 explores the speed of the implementation for both a real dataset and a collection of simulated datasets.

2.2 Parallelized MCMC

In most cases, the joint full posterior density $p(\mu, \phi|y)$ for the model in Equation (2.1) (Figure 2.1) cannot be found analytically, so Markov chain Monte Carlo (MCMC) is often used to obtain samples that converge to draws from this posterior. A two-step Gibbs sampler involves alternately sampling μ from its full conditional, $p(\mu|\ldots)$, and ϕ from its full conditional, $p(\phi|\ldots)$, where '$\ldots$' indicates all other parameters and the data. If these full conditionals have no known form, then the Gibbs step is typically replaced with a Metropolis-Hastings, rejection sampling, or slice sampling step (Gelman et al., 2013; Neal, 2003).

For high-dimensional group-specific parameters μ_g and hyperparameters ϕ, it is often impractical to sample the entire vector μ or ϕ jointly. In these scenarios, the group-specific parameters and hyperparameters are decomposed into subvectors $\mu_g = (\mu_{g1}, \ldots, \mu_{gJ})$ and $\phi = (\phi_1, \ldots, \phi_K)$, respectively. The component-wise MCMC then proceeds by sampling from these lower-dimensional full conditionals using composition, random scan, or random sequence sampling (Johnson et al., 2013).

Parallelism increases the efficiency of these MCMC approaches in hierarchical models by simultaneous sampling when parameters are conditionally independent and using parallelized reductions when full conditionals depend on low-dimensional summaries of other parameters. For hierarchical models, each MCMC step uses conditional independence, reductions, or both, and this designation partitions steps into classes. When the number of groups G is large, conditional independence can lead to a G-fold speedup while parallelized reductions can give a speedup of $\approx G/\log_2(G)$.

2.2.1 Simultaneous steps for conditionally independent parameters

In the two-level hierarchical model of Equation (2.1), the group-specific parameters μ_g are conditionally independent since

$$p(\mu|\ldots) \propto \prod_{g=1}^{G} p(y_g|\mu_g)p(\mu_g|\phi) \propto \prod_{g=1}^{G} p(\mu_g|y_g, \phi).$$

The theory of DAGs also reveals this conditional independence, specifically in nodes that are *d-separated* given the conditioning nodes (Koller and Friedman, 2009, Ch. 3). In Figure 2.1, the nodes $\mu_1, \ldots, \mu_G$ are d-separated given ϕ and therefore conditionally independent. Thus the vectors μ_g can be sampled simultaneously and in parallel.

Often it is more convenient to sample subvectors of the vector μ_g. The jth subvector is conditionally independent across g since $p(\mu_{gj}|\ldots) \propto p(y_g|\mu_g)p(\mu_g|\phi)$. Hence, we sample the μ_g's, or μ_{gj}'s, in parallel, simultaneous Gibbs steps.

Parallel execution is accomplished by assigning each group parameter to its own independent unit of execution, or thread. With G simultaneous threads, parallelizing across these groups has a theoretical maximum G-fold speedup relative to a serial implementation. For RNA-seq data analysis with $G \approx 40000$, this is a sizable improvement.

In addition to conditional independence, the full conditional for a group-specific parameter μ_g or μ_{gj} depends only on the data y_g for that group (and the hyperparameters ϕ). Thus, when performing parallel operations, memory transfer is minimized since only a small amount of the total data will need to be accessed by each parallel thread.

Our approach to parallelizing Gibbs steps is a special case of "embarrassing parallel" computation, which is parallelism without any interaction (i.e. data transfer or synchronization) among simultaneous units of execution. Embarrassingly parallel computation is already utilized in existing applications of GPU computing in the acceleration of Bayesian computation. For instance, the strategy by Jacob et al. (2011) shows how embarrassingly parallel computation can accelerate independence Metropolis-Hastings. Much of Metropolis-Hastings is unavoidably sequential because, as with any Monte Carlo algorithm, the value at the current state depends on the value at the previous state. However, in independence Metropolis-Hastings, each proposal draw is generated independently of the previous one, so all proposals can be calculated beforehand in embarrassingly parallel fashion using simultaneous independent threads. Similarly, the Metropolis-Hastings acceptance probability (Figure 1, Jacob et al. (2011)) at each step contains a factor that depends only on the current

proposal, and these factors can similarly be computed in parallel.

Parallelization of importance sampling is similarly straightforward, as in Figure 2 of Lee et al. (2010). The strategy by Gruber et al. (2016), which uses a decoupling/recoupling strategy to fit dynamic linear models of multivariate time series, takes advantage of embarrassingly parallel computation both within and among importance samplers. At each time point, they parallelize across the non-temporal dimension to draw Monte Carlo samples separately from independent prior distributions in their model (Section 3-B), and then parallelize across both the non-temporal dimension and the Monte Carlo sample size to draw from approximate posterior distributions (Section 3-C,D).

2.2.2 Reductions to aid the efficiency of hyperparameter sampling

The hyperparameter full conditionals, $p(\phi|\ldots)$ or $p(\phi_k|\ldots)$, usually depend on *sufficient quantities* that act as sufficient statistics of μ. For example, if $\mu_g \overset{\text{ind}}{\sim} N(\phi_1, \phi_2^2)$, then the sum of μ_g and the sum of μ_g^2 over index g are minimal sufficient for $\phi = (\phi_1, \phi_2)$. More generally, if $p(\mu_g|\phi)$ is an exponential family (or generalized linear model), then there is a sufficient quantity that depends on the model matrix (design matrix) and μ (McCullagh and Nelder, 1989, Ch. 2).

Each sufficient quantity can be computed using a *reduction*, i.e. repeated application of a binary operator to pairs of μ_g's until a single scalar is returned. A serial application of a reduction over G quantities requires $G-1$ operations. In contrast, a parallelized reduction over $G/2$ threads has complexity $\log_2(G)$. For large G, the speedup is considerable, so parallelizing the reductions on μ speeds up the sampling of the hyperparameter full conditionals. For example, for RNA-seq data analysis with $G \approx 40000$, a parallelized reduction provides a theoretical speedup of $\frac{G-1}{\log_2(G)} \approx 2600$. Of course, the observed efficiency gain depends on the software implementation, and many parallel computing frameworks have built-in optimized reduction functionality. CUDA's Thrust library, for example, allows the user to perform a fast parallelized reduction with a single line of code (NVIDIA, 2015).

Steps requiring reductions can be identified in a DAG where nodes have directed edges outward. When the number of edges from a node is large, a parallelized reduction is beneficial. For the two-level hierarchical model of Figure 2.1, the node ϕ has G exiting edges, and when G is large, the corresponding sampler benefits from a parallelized reduction. In the case where μ_g is of low-dimension and y_g is relatively large, e.g. $y_{gj} \overset{\text{ind}}{\sim} N(\mu_g, \sigma^2)$ for $j = 1, \ldots, J$ where J is large, a parallelized reduction for each μ_g may also be beneficial.

Reductions are used in other GPU-accelerated Bayesian analyses and Markov chain Monte Carlo routines. As an example, consider the GPU-accelerated Gaussian process modeling method by Gramacy et al. (2014). A major goal is to generate a large set of predictions, where each prediction is computed using a different subset of the available data. Each of these optimal subsets is determined with a criterion equivalent to mean squared prediction error, and the computation of this criterion, which depends on quadratic forms involving the correlation matrix, is expensive. As part of the acceleration, Gramacy et al. (2014) use parallelized pairwise summation in the calculation of these quadratic forms. Rather than Thrust, their implementation uses the parallelized reduction method by SHARCNET (2012). For other examples of parallelized reductions in Bayesian methods and MCMC, see Suchard et al. (2010) and Suchard and Rambaut (2009).

2.2.3 More hierarchical levels

Dichotomizing Gibbs steps into those that benefit from conditional independence and those that benefit from parallelized reductions extends to additional levels of hierarchy. Consider the three-level hierarchical model

$$y_{kg} \mid \mu_{kg} \overset{\text{ind}}{\sim} p(y_{kg}|\mu_{kg}), \quad \mu_{kg} \mid \phi_k \overset{\text{ind}}{\sim} p(\mu_{kg}|\phi_k), \quad \text{and} \quad \phi_k \mid \psi \overset{\text{ind}}{\sim} p(\phi_k|\psi) \tag{2.2}$$

where $k = 1, \ldots, K$, $g = 1, \ldots, G_k$, and y_{kg}, μ_{kg}, ϕ_k, and ψ could all be vectors. Figure 2.2 displays a DAG representation of the model in Equation (2.2). A two-step Gibbs sampler

for this model alternately samples

$$\mu, \psi \sim p(\mu|y, \phi)p(\psi|\phi) \quad \text{and} \quad \phi \sim p(\phi|y, \mu, \psi)$$

which shows that μ and ψ are conditionally independent given ϕ. The components of μ are conditionally independent, as well as the components of ϕ, since

$$p(\mu|\ldots) \propto \prod_{k=1}^{K}\prod_{g=1}^{G_k} p(y_{kg}|\mu_{kg})p(\mu_{kg}|\phi_k) \quad \text{and} \quad p(\phi|\ldots) \propto \prod_{k=1}^{K}\prod_{g=1}^{G_k} p(\mu_{kg}|\phi_k)p(\phi_k|\psi).$$

Figure 2.2 also displays these conditional independencies: ψ and μ_{kg} ($k = 1, \ldots, K$, $g = 1, \ldots, G_k$) are d-separated given y and ϕ, and the ϕ_k's are d-separated given μ and ψ.

As before, the full conditional of ϕ_k depends on a sufficient quantity calculated from $\{\mu_{k1}, \ldots, \mu_{kG_k}\}$ and the full conditional of ψ depends on a sufficient quantity calculated from ϕ. Figure 2.2 displays this relationship as well since there are 1) many edges from ψ to the ϕ_k's, and 2) many edges from ϕ_k to the μ_{kg}'s.

If K and G_k for $k = 1, \ldots, K$ are large, then parallelizing these conditional independencies and calculations of sufficient quantities will dramatically improve computational efficiency. When additional levels are added to the hierarchy, each full conditional can be categorized into a conditional independence step, a parallelized reduction step, or both.

2.3 Acceleration with high-performance computing

Three general parallel computing architectures currently exist: multi-core/CPU machines, clusters, and accelerators. Modern computers have multiple CPUs, each with multiple cores, where each core can support one or more parallel threads or processes. This multi-core/CPU hardware allows fast communication among threads, as the threads have abundant shared memory. However, this paradigm only supports tens of simultaneous threads, not hundreds or thousands, and thus will not provide the desired efficiency gain. In contrast, clusters, or collections of networked computers, provide the possibility of unlimited parallelism, but communication of threads occurs across a relatively slow network. Between these

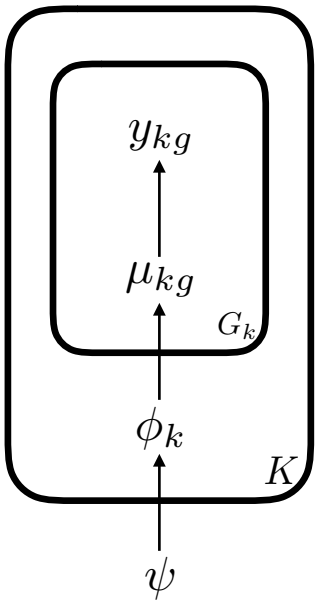

Figure 2.2: Directed acyclic graph (DAG) representation of the three-level hierarchical model in Section 2.2.3. The box with K indicates replication over k, and the box with G_k indicates replication over g.

extremes lie accelerators such as NVIDIA CUDA graphics processing units (GPUs) and Intel MIC coprocessors. GPUs in particular are capable of spawning hundreds of thousands of threads at a time, and these threads are partitioned into groups called blocks (Nickolls et al., 2008). Each block can contain hundreds of threads, and communication among the threads in a single block is extremely fast, driving the acceleration of reductions even when several blocks are needed.

However, if GPUs are used, the implementation strategy needs to be optimized for GPU computing. In particular, it is important to minimize the amount of data transferred between CPU memory and GPU memory (Beam et al., 2015). In our applications, this data transfer is by far the most time-consuming step, and misuse can easily defeat the purpose of GPU computing altogether. In particular, copying all MCMC parameter samples from GPU memory to CPU memory would be intractably slow. In addition, it is important to avoid exhausting all available GPU memory. There are opportunities to make GPU computing effective throughout the whole analysis.

2.3.1 Cumulative means and means of squares

Using cumulative means on the GPU, keep track of the mean and mean square of each parameter's MCMC samples, separately for each MCMC chain in the analysis. More specifically, suppose C independent MCMC chains with M iterations each are used to estimate some joint posterior distribution. In addition, let θ be an arbitrary parameter and $\theta_c^{(m)}$ be the the m'th MCMC sample of θ in chain c, where $m = 1, \ldots, M$ and $c = 1, \ldots, C$. Using a one-pass algorithm (Ling, 1974) over the course of the MCMC, record each $\overline{\theta}_c = \frac{1}{M}\sum_{m=1}^{M}\theta_c^{(m)}$ and $\overline{\theta^2}_c = \frac{1}{M}\sum_{m=1}^{M}\left(\theta_c^{(m)}\right)^2$. One option for the computation of $\overline{\theta}_c$ is to update the cumulative sum $\sum_{i=1}^{m}\theta_c^{(i)}$ on each MCMC iteration m and divide by M at the end, an approach that may suffer a loss of precision in some applications. A one-pass algorithm due to Welford (1962), on the other hand, which updates $x_{m-1} = \frac{1}{m-1}\sum_{i=1}^{m-1}\theta_c^{(i)}$ to $x_m = x_{m-1} + \frac{\theta_c^{(m)} - x_{m-1}}{m}$ on iteration m, is more numerically stable.

The quantities $\overline{\theta}_c$ and $\overline{\theta^2}_c$ $(c = 1, \ldots, C)$ have two major uses. The first is for assessing convergence via the Gelman-Rubin potential scale reduction factor (Gelman et al., 2013) for θ, which is given by

$$\widehat{R} = \sqrt{1 + \frac{1}{M}\left(\frac{B}{W} - 1\right)}$$

where

$$B = \frac{M}{C-1}\sum_{c=1}^{C}\left(\overline{\theta}_c - \overline{\theta}\right)^2, W = \frac{1}{C}\sum_{c=1}^{C}S_c^2,$$

$$\overline{\theta} = \frac{1}{C}\sum_{c=1}^{C}\overline{\theta}_c, \text{ and } S_c^2 = \frac{M}{M-1}\left[\overline{\theta^2}_c - \overline{\theta}_c^2\right] \approx \overline{\theta^2}_c - \overline{\theta}_c^2.$$

A Gelman factor $\widehat{R}$ far above 1 is evidence of lack of convergence in θ. It is a recommended and common practice to run at least 4 MCMC chains, starting at parameter values overdispersed relative to the full joint posterior distribution, and then check that the Gelman factors of parameters of interest are below 1.1 before moving forward with the analysis. The use of cumulative means allows the calculation of Gelman factors, and therefore convergence assessment, without the need to return all parameter samples.

The second main use of the cumulative mean and mean of squares is for point and interval estimates. By the Strong Law of Large Numbers, $\overline{\theta}$ and $\overline{\theta^2}$ converge almost surely to the expected values $E(\theta|y)$ and $E(\theta^2|y)$, respectively. Thus, $\overline{\theta}$ and $\overline{\theta^2} - \overline{\theta}^2$ are MCMC approximations to the posterior mean and variance, respectively. Since the posterior distribution itself converges to a normal distribution for large amounts of data (the primary use case for the computational methods developed here), a $100(1-\alpha)\%$ approximate equal-tail credible interval can be constructed via $\overline{\theta} \pm z_{\alpha/2}\sqrt{\overline{\theta^2} - \overline{\theta}^2}$ where $P(Z > z_\alpha) = \alpha$ and Z is a standard normal distribution.

With approximate credible intervals for each model parameter, it is typically not necessary to save all MCMC parameter samples. To reduce data transfer between the GPU and CPU, we recommend copying only a select few parameter samples back to CPU memory for future use, preferably the hyperparameters ϕ and some group-specific parameter samples μ_{gj} for a select, perhaps random, few values of g and j. For those parameters, an appropriate thinning interval should be used, although the cumulative means and means of squares should be calculated using all samples, and the GPU should retain only a single iteration at any given time. That way, memory-based computational bottlenecks are avoided, and enough parameter samples will be available post-hoc for checking the distributional assumptions of the approximate credible intervals.

2.3.2 Inference

Similar to the point and interval estimates in Section 2.3.1, inferential quantities that depend on μ should be calculated using cumulative means instead of the full collection of MCMC parameter samples. For example, a posterior probability that can be expressed as $P(f(\mu, \phi) \mid y)$ should be estimated by $\frac{1}{M}\sum_{m=1}^{M} \mathrm{I}\left(f\left(\mu^{(m)}, \phi^{(m)}\right)\right)$, where f is any function of the parameters that returns a true/false value, $\mathrm{I}(\cdot)$ is the indicator function, and the mean of indicator functions can be calculated using a one-pass algorithm as in Section 2.3.1.

Unfortunately, most parallel computing tools operate at a low level, so it is generally

impossible to allow the user to specify a generic function f. However, posterior probabilities involving contrasts are straightforward to implement. Such a probability is of the form, $P\left(u_1^T\eta > b_1 \text{ and } \ldots \text{ and } u_K^T\eta > b_K \mid y\right)$, where η is the vector obtained by concatenating the μ_g vectors and ϕ, each fixed vector u_k $(k = 1, \ldots, K)$ has the same length as η, and $b_1, \ldots, b_K$ are fixed scalars.[1] The MCMC estimate is
$\frac{1}{M}\sum_{m=1}^{M} \mathrm{I}\left(u_1^T\eta^{(m)} > b_1 \text{ and } \ldots \text{ and } u_K^T\eta^{(m)} > b_K\right)$, where $\eta^{(m)}$ is the MCMC sample of η at iteration m. For probabilities specific to each μ_g, if the μ_g's are all of the same length, the formulation is
$P\left(v_1^T\mu_g > b_1 \text{ and } \ldots \text{ and } v_K^T\mu_g > b_K \mid y\right)$ for $g = 1, \ldots, G$, where $v_1, \ldots, v_K$ are fixed vectors of the same length as μ_1. In this case, the estimate for index g is
$\frac{1}{M}\sum_{m=1}^{M} \mathrm{I}\left(v_1^T\mu_g^{(m)} > b_1 \text{ and } \ldots \text{ and } v_K^T\mu_g^{(m)} > b_K\right)$, and these estimates can be updated in parallel over index g. An example of this last construction is
$P\left(\mu_{g2} + \mu_{g4} > 0 \text{ and } \mu_{g3} + \mu_{g4} > 0 \mid y\right)$ for $g = 1, \ldots, G$, estimated by
$\frac{1}{M}\sum_{m=1}^{M} \mathrm{I}\left(\mu_{g2}^{(m)} + \mu_{g4}^{(m)} > 0 \text{ and } \mu_{g3}^{(m)} + \mu_{g4}^{(m)} > 0\right)$ for a given g. These posterior probabilities often arise in RNA-seq data analysis where the goal is often to detect genes with important patterns in their expression levels.

2.4 Application to RNA-sequencing data analysis

We apply the above strategy to RNA-sequencing (RNA-seq) data analysis. RNA-seq is a class of next-generation genomic experiments that measure the expression levels of genes in organisms across multiple groups or experimental conditions. The data from such an experiment is a matrix of counts, where the count in row g and column n is the relative expression level of gene g found in RNA-seq sample n. For a more detailed, technical

[1] Practitioners may desire $P\left(u_1^T\eta > b_1 \otimes_1 \ldots \otimes_K u_K^T\eta > b_K \mid y\right)$, where each $\otimes_k$ could be either "and" or "or". This general form can be obtained from probabilities using only "and" along with the general disjunction rule in basic probability theory. For example, $P\left(u_1^T\eta > b_1 \text{ or } u_1^T\eta > b_1\right) = P\left(u_1^T\eta > b_1\right) + P\left(u_2^T\eta > b_2\right) - P\left(u_1^T\eta > b_1 \text{ and } u_1^T\eta > b_1\right)$. The probabilities on the right are estimated using a one-pass algorithm during the MCMC, and then the estimate on the left is calculated afterwards. This restriction to "and" in the main program simplifies the implementation.

description of RNA-seq experiments and data preprocessing, see Datta and Nettleton (2014), Oshlack et al. (2010), and Wang et al. (2010).

The goal of the analysis is to model gene expression levels and detect important genes, a difficult task because there are typically $G \approx 40000$ genes and $N \approx 10$ RNA-seq samples. Hierarchical models are suitable because they borrow information across genes to improve detection. However, fitting them is computationally demanding because of the high number of genes and low number of observations per gene. Many approaches ease the computation with empirical Bayes methods, where the hyperparameters ϕ are set constant at values calculated from the data that approximate the respective target densities before the MCMC begins (Hardcastle (2012); Ji et al. (2014); Niemi et al. (2015)). However, empirical Bayes approaches ignore uncertainty in the hyperparameters, so a fully Bayesian solution may be preferred.

2.4.1 Model

Let y_{gn} be the RNA-seq count for sample n ($n = 1, \ldots, N$) and gene g ($g = 1, \ldots, G$). Let X be the $N \times L$ model matrix for gene-specific effects $\beta_g = (\beta_{g1}, \ldots, \beta_{gL})$. Let X_n be the n^{th} row of X. We assume $y_{gn}|\varepsilon_{gn}, \beta_g \overset{\text{ind}}{\sim} \text{Poisson}\left(\exp\left(h_n + \varepsilon_{gn} + X_n\beta_g\right)\right)$. The h_n's are constants estimated from the data, and they take into account sample-specific nuisance effects such as sequencing depth (Si and Liu (2013), Anders and Huber (2010), Robinson et al. (2010a)). The ε_{gn} parameters account for overdispersion, and we assign $\varepsilon_{gn}|\gamma_g \overset{\text{ind}}{\sim} \text{Normal}(0, \gamma_g)$. The γ_g parameters are analogous to the typical gene-specific negative-binomial dispersion parameters used in many other methods of RNA-seq data analysis (Landau and Liu, 2013). We assign $\gamma_g|\tau, \nu \overset{\text{ind}}{\sim} \text{Inverse-Gamma}\left(\nu/2, \nu\tau/2\right)$. τ is a prior measure of center of the γ_g terms (between the prior mean and the prior mode), and ν is the degree to which the γ_g's "shrink" towards τ. We assign $\tau \sim \text{Gamma}(a, \text{rate} = b)$ and $\nu \sim \text{Uniform}(0, d)$, where $a = 1$, $b = 1$, and $d = 1000$ are fixed constants such that these priors are diffuse (Gelman, 2006).

The β_g terms relate elements of the model parameterization to gene expression levels,

and we interpret $X_n\beta_g$ to be the log-scale mean expression level of gene g in RNA-seq sample n. For each fixed ℓ from 1 to L, we assign $\beta_{g\ell}|\theta_\ell, \sigma_\ell \overset{\text{ind}}{\sim} \text{Normal}(\theta_\ell, \sigma_\ell^2)$. Lastly, we assign $\theta_\ell \overset{\text{ind}}{\sim} \text{Normal}(0, c_\ell^2)$ and $\sigma_\ell \overset{\text{ind}}{\sim} \text{Uniform}(0, s_\ell)$, where $c_\ell = 10$ and $s_\ell = 100$ ($\ell = 1, \ldots, L$) are fixed constants so that these priors are diffuse (Gelman, 2006). This model is summarized and depicted as a DAG in Figure 2.3.

The conditional independence of the $\beta_{g\ell}$'s depends on the model matrix X. Parameters $\beta_{1\ell}, \ldots, \beta_{G\ell}$ are always conditionally independent given θ_ℓ and σ_ℓ, but β_{gi} and β_{gj} are not necessarily conditionally independent for $i \neq j$. To see this, it is easiest to refer to the directed acyclic graph (DAG) representation of the model in Figure 2.3. The dashed arrow from $\beta_{g\ell}$ to y_{gn} indicates that an edge is present if and only if $X_n\beta_g$ is a non-constant function of $\beta_{g\ell}$: that is, if and only if $X_{n\ell} \neq 0$. If there exists any integer n from 1 to N such that there is a directed edge from β_{gi} to y_{gn} and another directed edge from β_{gj} to y_{gn}, then β_{gi} and β_{gj} are not conditionally independent: here, y_{gn} is a collider on an undirected path between β_{gi} and β_{gj}, making β_{gi} and β_{gj} not d-separated in the DAG given the other nodes. If no such n exists, then β_{gi} and β_{gj} are d-separated given the other nodes and thus conditionally independent.

This RNA-seq model is a special case of the model in equation (2.1) and Figure 2.1. To make the transition, note that $(y_{g1}, \ldots, y_{gN})$ becomes y_g, $(\varepsilon_{g1}, \ldots, \varepsilon_{gN}, \gamma_g, \beta_{g1}, \ldots, \beta_{gL})$ becomes μ_g, and $(\tau, \nu, \theta_1, \ldots, \theta_L, \sigma_1, \ldots, \sigma_L)$ becomes ϕ.

2.4.2 MCMC

To fit the model to RNA-seq data, we use an overall Gibbs sampling structure and apply the univariate stepping-out slice sampler in Appendix A within each of several Gibbs steps. This versatile slice-sampling-within-Gibbs approach was suggested by Neal (2003) (Section 4: Single-variable slice sampling methods), then detailed by Cruz et al. (2015) and alluded to by Gelman et al. (2013) (Ch 12.3) and Banerjee et al. (2015). In each of the steps of Algorithm 1, a slice sampler is used to sample from all non-normal full conditionals. Each

$$
\begin{aligned}
y_{gn} &\overset{\text{ind}}{\sim} \text{Poisson}\left(\exp\left(h_n + \varepsilon_{gn} + X_n\beta_g\right)\right) \\
\varepsilon_{gn} &\overset{\text{ind}}{\sim} \text{Normal}(0, \gamma_g) \\
\gamma_g &\overset{\text{ind}}{\sim} \text{Inverse-Gamma}\left(\frac{\nu}{2}, \frac{\nu\tau}{2}\right) \\
\nu &\sim \text{Uniform}(0, d) \\
\tau &\sim \text{Gamma}(a, \text{rate} = b) \\
\beta_{g\ell} &\overset{\text{ind}}{\sim} \text{Normal}(\theta_\ell, \sigma_\ell^2) \\
\theta_\ell &\overset{\text{ind}}{\sim} \text{Normal}(0, c_\ell^2) \\
\sigma_\ell &\overset{\text{ind}}{\sim} \text{Uniform}(0, s_\ell)
\end{aligned}
$$

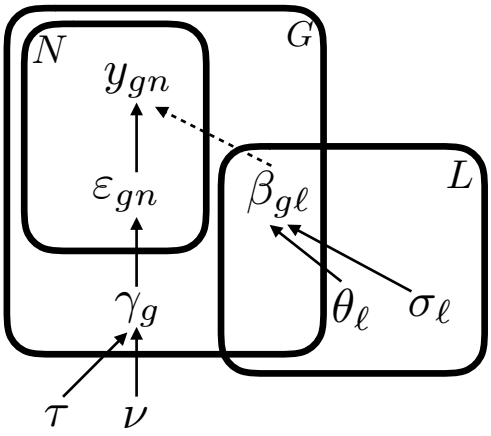

Figure 2.3: Directed acyclic graph (DAG) representation of the RNA-seq model in Section 2.4.1, along with a formulaic representation on the left. The box with G in the corner indicates that each parameter inside represents multiple nodes, each specific to a value of $g = 1, \ldots, G$. The analogous interpretation holds for the boxes with N and L, respectively. The dashed arrow from $\beta_{g\ell}$ to y_{gn} indicates that an edge is present if and only if $X_n\beta_g$ is a non-constant function of $\beta_{g\ell}$: that is, if and only if $X_{n\ell} \neq 0$, where X is the model matrix and X_n is its n'th row.

slice-sampled parameter (γ_1, $\gamma_2, \varepsilon_{50,5}$, etc.) has its own tuning variable w and auxiliary variable w_{aux}.

Slice sampling is used for the gamma and inverse-gamma full conditionals in addition to the full conditionals with unknown distributional form. This is because CURAND, the random number generation library for CUDA, has no gamma sampler. Although a gamma sampler could have been implemented (see Appendix B.3 of Gruber et al. (2016)), this slice-sampling approach is more versatile.

In Algorithm 1, we highlight the two types of steps: **in parallel** for the steps with conditionally independent parameters and **reduction** for the parameters whose full conditionals depend on sufficient quantities calculated from other parameters. In step 5, the $\beta_{g\ell}$'s are conditionally independent across g for a given ℓ, but not necessarily conditionally independent across ℓ, as the conditional independence of the $\beta_{g\ell}$'s depends on the model matrix. In steps 6 and 7, parameter sampling after the parallelized reductions could be parallelized, but the efficiency gain is small if L is small. In our application, L is 5.

Algorithm 1 MCMC for hierarchical RNA-seq model

1. **In parallel,** sample the ε_{gn}'s.
2. **In parallel,** sample the γ_g's.
3. **Reduction** to calculate $\sum_{g=1}^{G}\left[\log\gamma_g + \frac{\nu}{\gamma_g}\right]$. Then sample ν from its full conditional density, which is proportional to

$$\exp\left(-G\log\Gamma\left(\frac{\nu}{2}\right) + \frac{G\nu}{2}\log\left(\frac{\nu\tau}{2}\right) - \frac{\nu}{2}\sum_{g=1}^{G}\left[\log\gamma_g + \frac{\nu}{\gamma_g}\right]\right).$$

4. **Reduction** to calculate $\sum_{g=1}^{G}\frac{1}{\gamma_g}$. Then sample $\tau \sim \text{Gamma}\left(a + \frac{G\nu}{2},\ \text{rate} = b + \frac{\nu}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right)$.
5. For $\ell = 1, \ldots, L$, **in parallel,** sample $\beta_{1\ell}, \ldots, \beta_{G\ell}$.
6. **Reduction** to calculate means and variances of the relevant $\beta_{g\ell}$'s. Then sample $\theta_1, \ldots, \theta_L$.
7. **Reduction** to calculate the shape and scale parameters of the inverse-gamma distributions. Then sample $\sigma_1, \ldots, \sigma_L$.

2.4.3 Implementation

We release the implementation of this algorithm in R packages `fbseq` and `fbseqCUDA`, publicly available on GitHub in repositories named `fbseq` and `fbseqCUDA`, respectively. We use two packages for the same method in order to separate the GPU-dependent backend from the platform-independent user interface. `fbseq` is the pure-R user interface, which is for planning computation and analyzing results on any machine, such as a local office computer. `fbseqCUDA` is the CUDA-accelerated backend that runs Algorithm 1. The `fbseqCUDA` package uses custom CUDA kernels (functions encoding parallel execution on the GPU) to run sets of parallel Gibbs steps and CUDA's Thrust library for parallelized reductions. Users can install it on a computing cluster, a G2 instance on Amazon Web Services, or another (likely remote) CUDA-capable resource, and run the algorithm with a function in `fbseq` that calls the `fbseqCUDA` engine. For step-by-step user guides, please refer to the package vignettes. We also release `fbseqComputation`, an R package that replicates the results of this paper. The `fbseqComputation` package is publicly available through the GitHub repository of the same name. Install `fbseqComputation` according to the instructions in the package vignette, and run the `paper_computation()` function to reproduce the computation in Section 2.5.

2.5 Assessing computational tractability

As an example of RNA-seq data, we consider the dataset from Paschold et al. (2012). The underlying RNA-seq experiment focused on $N = 16$ biological replicates (pooled from the harvested primary roots of 3.5-day-old seedlings), each from one of 4 genetic varieties, and reported the expression levels of $G = 39656$ genes. The genetic varieties are B73 (an inbred population of Iowa corn), Mo17 (an inbred population of Missouri corn), B73×Mo17 (a hybrid population created by pollinating B73 with Mo17), and Mo17×B73 (a hybrid population created by pollinating Mo17 with B73).

The $N = 16$ by $L = 5$ model matrix is compactly represented as

$$X = \left(\begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \right)$$

where "$\otimes$" denotes the Kronecker product. With this model matrix, we can assign rough interpretations to the $\beta_{g\ell}$'s in terms of log counts. For gene g, β_{g1} is the mean of the parent varieties B73 and Mo17, β_{g2} is half the difference between the mean of the hybrids and Mo17, β_{g3} is analogous for B73, and β_{g4} is half the difference between the hybrid varieties. Finally, β_{g5} is a gene-specific block effect that separates the first two libraries from the last two libraries within each genetic variety due to the samples being on different flow cells.

One goal of the original experiment was to detect heterosis genes: in the case of high-parent heterosis, genes with significantly higher expression in the hybrids relative to both parents, and in the case of low-parent heterosis, genes with significantly lower expression in the hybrids relative to both parents. For example, to detect genes with high-parent heterosis with respect to B73×Mo17, we estimated $P\left(2\beta_{g2} + \beta_{g4} > 0 \text{ and } 2\beta_{g3} + \beta_{g4} > 0 | y\right)$ using the cumulative mean technique described in Section 2.3.

We fit the model in Section 2.4.1 to the Paschold dataset using our CUDA-accelerated R package implementation, `fbseq` and `fbseqCUDA`. We used a single node of a computing cluster with a single NVIDIA K20 GPU, two 2.0 GHz 8-Core Intel E5 2650 processors, and 64 GB of memory. We ran 4 independent Markov chains with starting values overdispersed relative to the full joint posterior distribution. We ran each chain with 10^5 iterations of burn-in and 10^5 true iterations. We used a thinning interval of 20 iterations so that 5000 sets of parameter samples were saved for each chain. As in Section 2.3, we only saved parameter samples for the hyperparameters $(\tau, \nu, \theta_1, \ldots, \theta_L, \sigma_1, \ldots, \sigma_L)$ and a small random subset of the gene-specific parameters. Running the 4 Markov chains in sequence, the total elapsed runtime was 3.89 hours.

To assess convergence, we combined the post-burn-in results of all 4 Markov chains. We used estimated posterior means and mean squares to calculate Gelman-Rubin potential scale reduction factors (see Section 2.3), which we used to monitor the $2L + 2$ hyperparameters, the $G \times L$ model coefficient parameters $\beta_{g\ell}$, and the G hierarchical variance parameters γ_g. All the corresponding Gelman factors fell below 1.1 except for $\beta_{26975,2}$ (at 1.167), $\beta_{33272,2}$ (at 1.148), γ_{33272} (at 1.112), and $\beta_{6870,2}$ (at 1.107). Although above 1.1, these last Gelman factors were still low and are not cause for serious concern. Next, for the total 2×10^4 saved parameter samples of each hyperparameter and of each of a small subset of gene-specific parameters, we computed effective sample size (Gelman et al., 2013). Most observed effective sizes were in the thousands and tens of thousands, the only exception being σ_2^2 at around 561 effective samples, well above the 10 to 100 effective samples recommended by Gelman et al. (2013). There was no convincing evidence of lack of convergence.

2.5.1 The scaling of performance with the size of the data

We used a simulation study to observe how the performance of our method scales with the number of genes and the number of RNA-seq samples. We used multiple new datasets, each constructed as follows. First, duplicate copies of the Paschold data were appended to produce a temporary dataset with the original 39656 genes and the desired number RNA-seq samples, N. Next, the desired number of genes, G, were sampled with replacement from the temporary dataset. We created 16 of these resampled datasets, each with a unique combination of $N = 16, 32, 48, 64$ and $G = 8192, 16384, 32768, 65536$ (i.e., $2^{13}, 2^{14}, 2^{15}$, and 2^{16}, respectively).

To each dataset, we applied the same method as in Section 2.5, with the same number of chains, iterations, and thinning interval. We also monitored convergence exactly as in Section 2.5. For 14 out of the 16 datasets, all Gelman factors of interest fell below our tolerance threshold of 1.1. For $G = 16384$ and $N = 16$, only the Gelman factors for $\beta_{9130,3}$ (at 1.119) and $\beta_{13704,2}$ (at 1.113) fell above 1.1. For $G = 65536$ and $N = 16$, there were

8 Gelman factors above 1.1. The highest of these was 1.325, and all corresponded to $\beta_{g\ell}$ and γ_g parameters. Across all 16 datasets, the minimum effective sample size (ESS) for any hyperparameter was roughly 185 (for σ_2^2). Again, evidence of lack of convergence is unconvincing.

Figure 2.4 shows the elapsed runtime in hours plotted against G and N. Runtime appears linearly proportional to both G and N within the range of values considered. These runtimes, also listed in the runtime column of Table 2.1, vary from 1.27 hours to 16.56 hours. Our method appears expedient given the size of a typical RNA-seq dataset at the time of this publication.

Table 2.1 also shows ESS for hyperparameters $(\nu,\ \tau,\ \theta_1, \ldots, \theta_L,\ \sigma_1^2, \ldots, \sigma_L^2)$. Overall, effective sample size appears acceptably high, and the time required to produce 1000 effective samples was relatively low for $N = 32$, 48, and 64. Of all the hyperparameters, σ_2^2 has the lowest ESS for $N = 16$. This parameter is the hierarchical variance of the β_{g2} parameters, which, for the current model parameterization and on the natural log scale, are the gene-specific half-differences between the mean of all the B73xMo17 and Mo17xB73 expression levels and the mean of the B73 expression levels. For $N = 32$, 48, and 64, τ is the minimum-ESS hyperparameter. Recall that τ is the prior center (between the prior mean and the prior mode) of the γ_g parameters, the counterparts of the gene-specific negative-binomial dispersion parameters often used in other models of RNA-seq data.

Naively, we should expect ESS to increase with both G and N, since hyperparameter estimation generally improves with increased information to borrow across genes. Prior speculation about the time required to produce a given number of effective samples, however, is trickier. With additional data, estimation improves, but computation is slower. Table 2.1 shows the interplay of these competing factors.

Many of the findings in Table 2.1 are unsurprising given our prior expectations. Median ESS nearly doubled from $N = 16$ to $N = 32$ for all values of G listed. Median ESS varied little among the larger values of N and G, presumably since ESS is already close to the total

aggregated 2×10^4 MCMC samples by that point. Next to median ESS in Table 2.1 is the average time required to produce a median ESS of 1000 across the hyperparameters. For the larger values of G, there is a noticeable increase in this timespan between $N = 48$ and $N = 64$, and for N fixed at 16, 32, 48 or 64, it increased roughly linearly with G. Minimum ESS, the minimum-ESS hyperparameter, and the time required to obtain 1000 effective samples of the minimum-ESS hyperparameter also showed some unsurprising trends. Minimum ESS increased from $N = 16$ to $N = 32$ for each value of G, and when τ was the minimum-ESS parameter, the time required to obtain 1000 effective samples increased roughly linearly with both N and G.

There are some surprises as well. Minimum ESS decreased with increasing N when τ was the minimum-ESS hyperparameter and also decreases from $G = 32768$ to $G = 65536$ when σ_2^2 was the minimum-ESS hyperparameter. Also for σ_2^2 at $N = 16$, the time required to obtain 1000 effective samples decreased as G increased from 8192, to 32768, but then spiked by the time $G = 65536$.

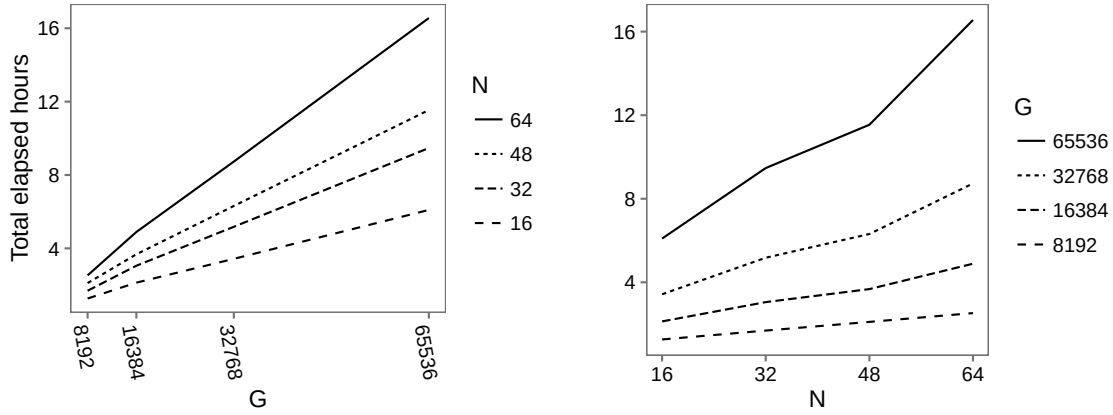

Figure 2.4: Elapsed runtime (hours) plotted against the number of genes (G) and the number of RNA-seq samples (N) for 2×10^5 total MCMC iterations for four chains run in sequence.

Table 2.1: Runtimes and effective sample sizes for the simulation study in Section 2.5.1. G is the number of genes, and N is the number of libraries. The runtime column shows the total elapsed runtime in hours. The ESS columns show numerical summaries (either median or minimum, as indicated in the top row) of effective sample size across all hyperparameters (ν, τ, $\theta_1, \ldots, \theta_L$, $\sigma_1^2, \ldots, \sigma_L^2$). The ratio columns show 1000 times runtime divided by ESS: that is, the average elapsed hours required to produce 1000 effective samples (median or minimum across hyperparameters).

			median		minimum		
G	N	runtime	ESS	ratio	ESS	ratio	parameter
8192	16	1.27	10170	0.12	185	6.86	σ_2^2
8192	32	1.69	19057	0.09	5473	0.31	τ
8192	48	2.11	18882	0.11	3810	0.55	τ
8192	64	2.53	19463	0.13	2303	1.10	τ
16384	16	2.13	10384	0.20	398	5.35	σ_2^2
16384	32	3.05	18845	0.16	5018	0.61	τ
16384	48	3.68	19525	0.19	3590	1.02	τ
16384	64	4.89	19510	0.25	2663	1.84	τ
32768	16	3.43	12567	0.27	990	3.46	σ_2^2
32768	32	5.18	18402	0.28	5501	0.94	τ
32768	48	6.31	19158	0.33	3279	1.92	τ
32768	64	8.73	19499	0.45	2450	3.56	τ
65536	16	6.09	11418	0.53	308	19.78	σ_2^2
65536	32	9.47	18945	0.50	5554	1.71	τ
65536	48	11.54	19657	0.59	3624	3.18	τ
65536	64	16.56	19409	0.85	2673	6.19	τ

2.6 Discussion

We present a fully Bayesian strategy to fit large hierarchical models that are computationally intractable under normal circumstances. We introduce the two main components of most parallelized Markov chain Monte Carlo approaches: embarrassingly parallel computations and reductions. We combine these components with a slice-sampling-within-Gibbs MCMC algorithm, and we harness the multi-core capabilities of GPUs. The CPU-GPU communication bottleneck is avoided by calculating running sums and sums of squares of relevant quantities. We demonstrate how these quantities can be used for convergence diagnostics and posterior inference. We exemplified these general approaches using a real RNA-seq dataset and satisfied standard convergence diagnostics in 3.89 hours of elapsed

runtime. In our simulation study based on the RNA-seq model, we found that total elapsed runtime scales linearly with the size of the data in each dimension within the range of sizes considered, and effective samples are hardest to obtain when the number of genes is high and the number of RNA-seq samples is low.

Major deterrents in the adoption of Bayesian methods are the development of computational machinery to estimate parameters in the model and the computation time required to estimate those parameters. General purpose Bayesian software such WinBUGS (Lunn et al., 2000), OpenBUGS (Lunn et al., 2009), JAGS (Plummer et al., 2003), Stan (Carpenter et al., 2016), and NIMBLE de Valpine et al. (2016) have lowered the development time by allowing scientists to focus on model construction rather than computational details. These software platforms are based on DAGs representations of Bayesian models and determine appropriate MCMC schemes based on these DAGs. Unfortunately, for analyses similar to the RNA-seq analysis presented here, estimation using these tools is computationally intractable. We hope the abstraction presented here and elsewhere, e.g. Beam et al. (2015), will inspire and spur the development of GPU-parallelized versions of these software enabling MCMC analyses of larger datasets and larger models.

CHAPTER 3. AN RNA-SEQUENCING CASE STUDY

Heterosis, or hybrid vigor, is the enhancement of the phenotype of hybrid progeny relative to their inbred parents. Heterosis is extensively used in agriculture, and the underlying mechanisms are unclear. To investigate the molecular basis of phenotypic heterosis, researchers search tens of thousands of genes for heterosis with respect to expression in the transcriptome. We present a general hierarchical model for count data and a fully Bayesian method in which an efficient parallelized Markov chain Monte Carlo algorithm ameliorates the computational burden. We use our method to detect gene expression heterosis in a two-hybrid plant breeding scenario, both in a real RNA-seq maize dataset and in simulation studies. In the simulation studies, we show our method has well-calibrated posterior probabilities and credible intervals when the model assumed in analysis matches the model used to simulate the data. Although model misspecification can adversely affect calibration, the methodology is still able to accurately rank genes. Finally, we show that hyperparameter posteriors are extremely narrow and an empirical Bayes (eBayes) approach based posterior means from the fully Bayesian analysis provides virtually equivalent posterior probabilities, credible intervals, and gene rankings to the fully Bayesian solution. This evidence of equivalence provides support for the use of eBayes procedures in RNA-seq data analysis if accurate hyperparameter estimates are obtained.

3.1 Introduction

Heterosis, or hybrid vigor, is the biological phenomenon in which hybrid progeny surpasses each of its inbred parents with respect to some characteristic. Ever since Darwin (1876) documented heterosis, the term has usually referred to traits at the phenotype level, and phenotypic heterosis has long been used to enhance crops and livestock. For example, one well-known maize hybrid described by Hallauer and Miranda (1981) and Hallauer

et al. (2010) has taller, faster-growing stalks with more grain yield than either inbred parent. Similar breeding techniques have used heterosis to improve rice (Yu et al., 1997), alfalfa (Riday and Brummer, 2002), tomatoes (Krieger et al., 2010), and fish (Wohlfarth, 1993). However, the underlying genomic mechanisms of phenotypic heterosis remain unclear (Coors and Pandey, 1999; Lippman and Zamir, 2007).

Researchers have hypothesized that the enhanced expression of one or more genes in the hybrid relative to both inbred parents, which we call gene expression heterosis, may help account for heterosis in the phenotype (Swanson-Wagner et al., 2006; Springer and Stupar, 2007). Gene expression heterosis has been measured with a variety of experimental techniques, including the microarray, along with its successor, RNA-sequencing (RNA-seq) (Wang et al., 2006, 2010; Oshlack et al., 2010). Both classes of experiment measure the relative expression levels of genes in organisms across multiple groups or experimental conditions. Relative to the microarray, RNA-seq has less noise and higher throughput, among other advantages (Landau and Liu, 2013). However, both the microarray and RNA-seq bring about serious statistical challenges. With a large number of expressed genes assayed only a handful of times each, the data analysis is a multiple testing scenario prone to false discoveries. The heterosis problem has the additional difficulty that gene detection uses composite null hypotheses, which some popular frequentist genomic data analysis methods do not accommodate easily (Robinson et al., 2010a; Anders and Huber, 2010; Lund et al., 2012).

Swanson-Wagner et al. (2006), Wang et al. (2006), and Bassene et al. (2010) confront the statistical problems in the context of microarray data analysis. In one recent development for the microarray setting, Ji et al. (2014) assess gene expression heterosis with respect to a single first-generation hybrid using a normal hierarchical model for transcript abundance. To mitigate the standard difficulties in multiple testing, they introduce an empirical Bayes framework that borrows information across genes to estimate means and variances. Under the framework, they estimate model hyperparameters, condition on fixed point-mass hyper-

parameter estimates, and then estimate the posterior distribution of gene-specific parameters. The estimated probability of heterosis of each individual gene is based on the integral under the relevant region of the posterior density, avoiding the problem of composite null hypotheses in heterosis detection.

Niemi et al. (2015) consider heterosis in the RNA-seq setting with a hierarchical negative binomial model for gene expression heterosis with respect to a single first-generation hybrid. Estimation is accomplished via another empirical Bayes procedure. After computing replicate-specific normalization factors with the TMM method by Robinson et al. (2010a), they estimate the hyperparameters (locations and scales of hierarchical distributions) using a central method of moments approach. Conditional on those hyperparameters, they estimate the joint posterior distribution of the other parameters using the Hamiltonian Monte Carlo routine in the software `RStan` (Stan Development Team, 2014) to obtain parameter samples.

Empirical Bayes schemes, which condition on data-driven hyperparameter estimates, are becoming increasingly popular in statistical genomics (Hardcastle and Kelly, 2010; Wu et al., 2012; Ji et al., 2014). The computational burden is reduced relative to fully Bayesian strategies, and the analyses retain many of the advantages of Bayesian methodology, including straightforward inference and the borrowing of information across genes to mitigate the effects of small sample sizes during estimation. Theoretically, however, empirical Bayes procedures risk lower quality estimation and posterior inference by ignoring the uncertainties of the hyperparameters.

This article is motivated by a two-hybrid maize heterosis RNA-seq dataset by Paschold et al. (2012), which Section 3.2 introduces. For the analysis, Section 3.3 presents a hierarchical RNA-seq model constructed by mixing a Poisson distribution over log-normal distributions. Section 3.3 also describes the fully Bayesian estimation procedure, which, as explained in detail by Chapter 2, would be computationally intractable without the massively parallelized Markov chain Monte Carlo algorithm implemented for general-purpose graphics processing units (GPUs). Section 3.4 expounds simulation studies based on a two-hybrid

plant breeding scenario that assessed our fully Bayesian approach in terms of estimation, inference, and heterosis gene detection, and we compare our method to two best-case-scenario empirical Bayes counterparts based on the same model. Finally, Section 3.5 details our analysis of the Paschold et al. data.

3.2 Two-hybrid plant breeding experiment for heterosis detection

We focus on the RNA-seq dataset from Paschold et al. (2012), which contains read counts of $G = 39656$ genes on $N = 16$ biological replicates divided evenly among four genetic varieties. In the underlying experiment, multiple maize seedlings from each variety were germinated according to a procedure by Hoecker et al. (2006). Three and a half days after germination, the primary roots of the seedlings were harvested. Within each variety, four pools of primary roots served as four biological replicates. Following a procedure by Winz and Baldwin (2001), the 16 collections of roots were ground under liquid nitrogen, and the RNA was isolated. Complementary DNA (cDNA) fragments were then synthesized in preparation for sequencing. Next the cDNA from the replicates was divided among two flow cells (i.e. removable compartments for genetic material in the RNA-sequencing platform) according to Supplementary Table S1 by Paschold et al. (2012). The two flow cells were placed into an Illumina Genome Analyzer II, where the cDNA fragments were read, amplified, and counted. The reads from the sequencing platform were mapped to the B73 reference genome (RefGen_v2) (Schnable et al., 2009), and the preprocessed and amplified read counts for each gene and biological replicate were collected into a data table. The original table has both allele-specific counts and total per-replicate counts, and we retain only the per-replicate counts in the analyses that follow. The $G = 39656$ by $N = 16$ count table we use is part of our Table S1.

The varieties are inbred variety B73, inbred variety Mo17, B73×Mo17 (a first-generation

hybrid created by pollinating B73 with Mo17), and Mo17×B73 (a first-generation hybrid created by pollinating Mo17 with B73). This is a special case of a more general plant hybrid scenario where there are two parent varieties and one first-generation hybrid variety for each direction of pollination. For the general scenario, we shall use P1, P2, H12, and H21 for the parents and the first-generation hybrids, respectively. For the Paschold et al. dataset, P1 is B73, P2 is Mo17, H12 is B73×Mo17, and H21 is Mo17×B73.

A major goal is to identify genes that have heterosis with respect to their expression levels: that is, those with significantly higher (in the case of high-parent heterosis) or significantly lower (low-parent heterosis) expression levels in one or both hybrids relative to their parents. For each of the high-parent and low-parent cases, we are interested in heterosis with respect to H12, H21, and the log-scale mean expression level of H12 and H21 together. Table 3.1 provides the six types of gene expression heterosis parameterized in terms of log-scale mean expression levels μ_{gx} specific to gene g and variety x. (The third column is discussed in Section 3.4.1.) "High (low)-parent H" indicates hybrid H has higher (lower) mean expression than both parents while "high (low)-parent mean" indicates that the average of the hybrids is higher (lower) than both parents. One can think of each kind of heterosis in the table as a statistical hypothesis that a frequentist would test for each individual gene. For us, a major objective is to provide a measure of the strength of evidence of each kind of heterosis for each gene, which we accomplish using posterior probabilities of heterosis under the model in Section 3.3.1.

3.3 Fully Bayesian methodology

3.3.1 Hierarchical model for RNA-seq

To look for heterosis genes, we use the general multi-level hierarchical model expounded by Chapter 2. Let y_{gn} be the RNA-seq count (i.e. the relative expression level) of gene g ($g = 1, \ldots, G$) in replicate n ($n = 1, \ldots, N$), and let y be the $G \times N$ matrix of the

Table 3.1: Heterosis hypotheses for a two-parent (P1 and P2), two-hybrid (H12 and H21) gene expression experiment represented in terms of the log-scale mean expression μ_{gx} for gene g and variety x and in terms of the parameters $\beta_{g\ell}$ corresponding to columns $\ell = 1, \ldots, L = 5$ of the model matrix X in Equation (3.1).

Heterosis	With log-scale group means	With $\beta_{g\ell}$ parameters
high-parent H12	$\mu_{g,\text{H12}} > \max(\mu_{g,\text{P1}}, \mu_{g,\text{P2}})$	$2\beta_{g2} + \beta_{g4},\quad 2\beta_{g3} + \beta_{g4} > 0$
low-parent H12	$\mu_{g,\text{H12}} < \min(\mu_{g,\text{P1}}, \mu_{g,\text{P2}})$	$-2\beta_{g2} - \beta_{g4}, -2\beta_{g3} - \beta_{g4} > 0$
high-parent H21	$\mu_{g,\text{H21}} > \max(\mu_{g,\text{P1}}, \mu_{g,\text{P2}})$	$2\beta_{g2} - \beta_{g4},\quad 2\beta_{g3} - \beta_{g4} > 0$
low-parent H21	$\mu_{g,\text{H21}} < \min(\mu_{g,\text{P1}}, \mu_{g,\text{P2}})$	$-2\beta_{g2} + \beta_{g4},\quad 2\beta_{g3} + \beta_{g4} > 0$
high-parent mean	$\mu_{g,\text{H12}} + \mu_{g,\text{H21}} > 2\max(\mu_{g,\text{P1}}, \mu_{g,\text{P2}})$	$\beta_{g2},\quad \beta_{g3} > 0$
low-parent mean	$\mu_{g,\text{H12}} + \mu_{g,\text{H21}} < 2\min(\mu_{g,\text{P1}}, \mu_{g,\text{P2}})$	$-\beta_{g2}, -\beta_{g3} > 0$

y_{gn}'s. Let X be an $N \times L$ model matrix that connects the N samples (i.e. RNA-seq replicates) to the genotypes, blocking factors, etc. Taking X_n to be the n'th row of X, we let $y_{gn} \overset{\text{ind}}{\sim} \text{Poisson}(\exp(h_n + \varepsilon_{gn} + X_n\beta_g))$. The constants h_n's, estimated from the data, play the role of normalization factors in other RNA-seq models, taking into account sample-specific nuisance effects such as sequencing depth (Si and Liu, 2013; Anders and Huber, 2010; Robinson et al., 2010a). The ε_{gn} parameters account for overdispersion, and we assume $\varepsilon_{gn} \mid \gamma_g \overset{\text{ind}}{\sim} \text{Normal}(0,\ \gamma_g)$ such that the γ_g parameters are analogous to the gene-specific negative-binomial dispersion parameters widespread in other RNA-seq data analysis methodology (Landau and Liu, 2013). We assumed $1/\gamma_g \overset{\text{ind}}{\sim} \text{Gamma}(\nu/2, \nu\tau/2)$, parameterized such that $E[1/\gamma_g] = 1/\tau$.

The gene-specific vector-valued parameters β_g account for the effects on gene expression of the experimental variables of interest. Aside from the normalization factor h_n, we interpret $X_n\beta_g$ to be the log-scale mean expression level of gene g in RNA-seq sample (replicate) n. For each fixed ℓ from 1 to L, we assign $\beta_{g\ell} \mid \theta_\ell,\ \sigma_\ell \overset{\text{ind}}{\sim} \text{Normal}(\theta_\ell,\ \sigma_\ell^2)$.

This model is similar to negative binomial regression models from other RNA-seq data analyses (McCarthy et al., 2012; Wu et al., 2012), with one difference being that we mix Poisson distributions over log-normal rather than gamma distributions. This choice is made primarily to ease computational implementation by reducing the number of distinct types of full conditionals.

3.3.2 Inference on gene-specific parameters and heterosis probabilities

To perform Bayesian analyses, we assigned independent priors for the hyperparameters. Specifically $\tau \sim \text{Gamma}(a, b)$, $\nu \sim \text{Uniform}(0, d)$, $\theta_\ell \overset{\text{ind}}{\sim} \text{Normal}(0, c_\ell^2)$, and $\sigma_\ell \overset{\text{ind}}{\sim} \text{Uniform}(0, s_\ell)$ for $\ell = 1, \ldots, L$. Before parameter estimation, we calculated the log-scale replicate-specific normalization constants h_n as follows. We first calculated log-scale counts $w_{gn} = \log(y_{gn} + 0.5 \cdot \text{I}(y_{gn} = 0))$, replicate-specific means $\overline{w}_{.n} = \frac{1}{G}\sum_{g=1}^{G} w_{gn}$, and the grand mean $\overline{w}_{..} = \frac{1}{N}\sum_{n=1}^{N} \overline{w}_{.n}$. Afterwards, we set $h_n = \overline{w}_{.n} - \overline{w}_{..}$ for $n = 1, \ldots, N$.

To estimate the full joint posterior distribution of the parameters, we used the parallelized slice-sampling-within Gibbs Markov chain Monte Carlo (MCMC) algorithm described in Chapter 2. Without parallel computing, the MCMC would be computationally intractable, and a fully Bayesian analysis would not be possible. However, with the strategy we employed, which uses massively parallel computing that takes advantage of general-purpose graphics processing units (GPUs), the approach is solidly tractable. The algorithm accelerates MCMC computation by executing conditionally independent Gibbs steps in parallel and using parallelized reductions to compute the full conditional distributions of the hyperparameters. Efficiency is increased by reducing the data transferred from GPU to CPU, and thus we limited posterior samples to all hyperparameters and a random subset of gene-specific parameters. For each parameter ψ, we also record $\overline{\psi} = \frac{1}{M}\sum_{m=1}^{M} \psi^{(m)}$ and $\overline{\psi^2} = \frac{1}{M}\sum_{m=1}^{M} \left(\psi^{(m)}\right)^2$, where $\psi_g^{(m)}$ is the m^{th} Monte Carlo sample of ψ_g, and approximate $p(\psi|y)$ with $N\left(\overline{\psi}, \overline{\psi^2} - \overline{\psi}^2\right)$. Finally, we assessed the posterior probabilities of heterosis in Table 3.1 via their ergodic averages, e.g.

$$P\left(\text{high-parent H12 heterosis for gene } g|y\right) \approx \frac{1}{M}\sum_{m=1}^{M} \text{I}\left(2\beta_{g2}^{(m)} + \beta_{g4}^{(m)} > 0 \text{ and } 2\beta_{g3}^{(m)} + \beta_{g4}^{(m)} > 0\right)$$

where $\text{I}(A)$ is 1 if A is true and 0 otherwise.

For each analysis of each dataset, we ran multiple chains, collectively totaling 2×10^5 Monte Carlo iterations. We monitored those chains for convergence using Gelman-Rubin

potential scale reduction factors $\widehat{R}$ (Gelman and Rubin, 1992) which were calculated using $\overline{\psi}$ and $\overline{\psi^2}$ (see Chapter 2). Specifically, we monitored $\widehat{R}$ on the $2L+2$ hyperparameters, the $G \times L$ parameters $\beta_{g\ell}$, and the G hierarchical variance parameters γ_g. In our experience, we found $\widehat{R}$ values near one for all but a few gene-specific parameters that varied when rerunning the MCMC. In addition, since we retained Monte Carlo samples of the hyperparameters, we monitored hyperparameter effective sample size, which we generally found to be well above the 10 to 100 effective samples recommended by Gelman et al. (2013).

For computation, we used the R packages `fbseq` and `fbseqCUDA` publicly available on GitHub. We also released `fbseqStudies`, an R package that replicates all the results of this paper. The `fbseqStudies` package is publicly available through the GitHub repository of the same name. Installing `fbseqStudies` according to the instructions in the package vignette and running the `paper_case()` function reproduces the computation, figures, tables, etc. shown in all the following sections.

3.4 Studies of simulated heterosis datasets

We assessed coverage of credible intervals (CIs), calibration of posterior probabilities, and the ability of our method to rank genes by constructing simulations with known values of the gene-specific parameters. For CIs, we calculated coverage, i.e. the proportion of genes whose true parameter value falls within the interval, across all genes and as a function of the parameter value, and compared this proportion to the intervals' credibility. To assess calibration of posterior probabilities, we constructed kernel-smoothed plots of the true heterosis status of each gene against its estimated posterior probability, and we refer to these figures as calibration curves throughout. For each calibration curve, we calculated the mean absolute vertical distance from the identity line, which we call calibration error. Posterior probabilities provide a ranking of genes of interest for each hypotheses in Table 3.1. To evaluate these rankings, we constructed receiver operating characteristic (ROC) curves and the areas under these curves (Landau and Liu, 2013).

For all the simulation studies in this article, we simulated RNA-seq count datasets under the plant hybrid scenario from Section 3.2. Each dataset contained count data on $G = 30000$ genes for $N = 16$ or $N = 32$ total replicates spread evenly over the P1, P2, H12, and H21 varieties. From left to right, the columns in each count data table y corresponded to P1, P2, H12, and then H21, respectively. Within each variety, the all columns for the first block (flow cell) preceded all the columns of the second block. Our model matrix, which we also used to analyze the Paschold et al. dataset in Section 3.5, is compactly represented as

$$X = \left(\begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix} \otimes J_{(N/4)\times 1} \qquad J_{(N/4)\times 1} \otimes \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \right) \tag{3.1}$$

where "$\otimes$" denotes the Kronecker product and $J_{m\times n}$ is the m by n matrix with all entries equal to 1. We chose the first $\ell = 1, \ldots, 4$ columns of the $N \times L$ model matrix ($L = 5$) to strategically model gene expression heterosis. For a maize dataset similar to that of Paschold et al., Lithio and Nettleton (2015) found strong correlations among gene-specific model coefficient parameters, a phenomenon that could potentially violate our model's conditional independence assumptions. To mitigate this effect among columns $\ell = 1, \ldots, 4$, we selected a slightly reparameterized, two-hybrid version of the parameterization used by Niemi et al. (2015) and Ji et al. (2014). Column $\ell = 5$ of X is a gene-specific experimental block effect, used in the analysis of the Paschold et al. data (Section 3.5) to account for the difference between the two flow cells of the sequencing platform in the original experiment. Table 3.2 provides interpretations for the parameters $\beta_{g\ell}$ in terms of the log-scale group means while Table 3.1 provides the method to evaluate each heterosis hypothesis using these $\beta_{g\ell}$ parameters.

For Section 3.4.1, we evaluated coverage, calibration, and ranking for simulations where the assumed model in the analysis matched the model used to generate data. Section 3.4.2 provides an assessment of robustness under two alternative data-generating scenarios as well

Table 3.2: For the model matrix in Equation (3.1), interpretations of the parameters $\beta_{g\ell}$ in terms of the group means μ_{gx} (gene g, group x). Group means and interpretations in the table are given on the natural logarithmic scale. β_{g5} cannot be expressed in terms of the group means, so only the prose interpretation is given.

$\beta_{g\ell}$	Using group means	Log-scale interpretation
$\ell = 1$	$\frac{\mu_{g,\text{P1}} + \mu_{g,\text{P2}}}{2}$	Parental mean
$\ell = 2$	$\frac{(\mu_{g,\text{H12}} + \mu_{g,\text{H21}})/2 - \mu_{g,\text{P2}}}{2}$	Half difference, hybrid mean versus parent 2
$\ell = 3$	$\frac{(\mu_{g,\text{H12}} + \mu_{g,\text{H21}})/2 - \mu_{g,\text{P1}}}{2}$	Half difference, hybrid mean versus parent 1
$\ell = 4$	$\frac{\mu_{g,\text{H21}} - \mu_{g,\text{H12}}}{2}$	Half the difference between hybrids
$\ell = 5$	–	Flow cell block effect

as a comparison to an empirical Bayes approach.

3.4.1 Assessing performance when the data-generation and analysis models agree

We generated 10 datasets from the model in Section 3.3.1 using $N = 16$ total replicates per dataset. To generate each dataset, we fixed hyperparameters ν, τ, $\theta_1, \ldots, \theta_5$, and $\sigma_1, \ldots, \sigma_5$ to values similar to the posterior modes of the real data shown in Figure 3.4. Conditioning on those fixed hyperparameter values, we generated the γ_g's and $\beta_{g\ell}$'s from their hierarchical distributions under the model. Similarly, we conditioned on those γ_g values to generate the ε_{gn}'s from their hierarchical distributions. Finally, with parameter values in hand and the model matrix X given by Equation (3.1), we generated RNA-seq counts y_{gn} using the Poisson $(\exp(\varepsilon_{gn} + X_n\beta_g))$ distribution from the model (with $h_n = 0$ for all $n = 1, \ldots, N$).

We used a single node of a computing cluster with a single NVIDIA K20 GPU, two 2.0 GHz 8-Core Intel E5 2650 processors, and 64 GB of memory. For each dataset, we ran 4 independent Markov chains with overdispersed starting values relative to the full joint posterior distribution of the parameters. For each chain, we used a burn-in period of 10^5 iterations (the first 50 of those iterations without tuning the slice sampler), and then 10^5

true iterations with a thinning interval of 20 so that 5000 samples are retained for a small subset of parameters of interest. Whereas serial execution would have been computationally intractable, parallelization with a GPU drove down the maximum total elapsed runtime per dataset to 3.2 hours. For each dataset, no more than 9 $\widehat{R}$ values were above 1.1 and these all correspond to $\beta_{g\ell}$ parameters. For the hyperparameters, the minimum effective sample size across all simulated datasets was ~ 600 (for σ_4^2). Evidence of lack of convergence was weak overall, though estimation and inference may have been poor for the few genes with $\widehat{R} > 1.1$.

With these results, we assessed the accuracy of posterior inference on the hyperparameters. Since the data were generated from the model, we expected credible intervals of the hyperparameters to be accurate overall. In particular, on average, around half of all 50%-credible intervals should contain the hyperparameter values used to generate the data. Figure S1 shows the estimated credible intervals for all datasets, along with the true values used in data generation. There appears to be no apparent overall bias in the location of the intervals, and for most of the hyperparameters, between 4 and 6 estimated 50% credible intervals cover the truth. The exceptions are ν, where 2 intervals cover the truth and 3 others barely miss, and θ_5, where 8 intervals cover the truth. Overall, the quality of posterior inference on the hyperparameters appears satisfactory.

We also assessed posterior inference on the parameters $\beta_{g\ell}$ because they are important for detecting heterosis genes. As described in Section 3.3.2, we retained full samples of only a few randomly selected $\beta_{g\ell}$ and otherwise approximate posteriors via their normal approximations. Across the simulations, coverage for normal-based 95% CIs ranged from 94.7% to 95.4% for $\ell \neq 4$ and from 92.9% to 96.7% for $\ell = 4$. Figure 3.1 displays the smoothed coverage proportions plotted against the true parameter values. From the figure, for each ℓ, coverage exceeded desired minimum near the overall mean true parameter value, but dropped abruptly for extreme parameter values. For $\ell > 1$, the low $\beta_{g\ell}$'s tended to be overestimated and the high $\beta_{g\ell}$'s tended to be underestimated, i.e. the CIs shrunk towards

the hierarchical mean.

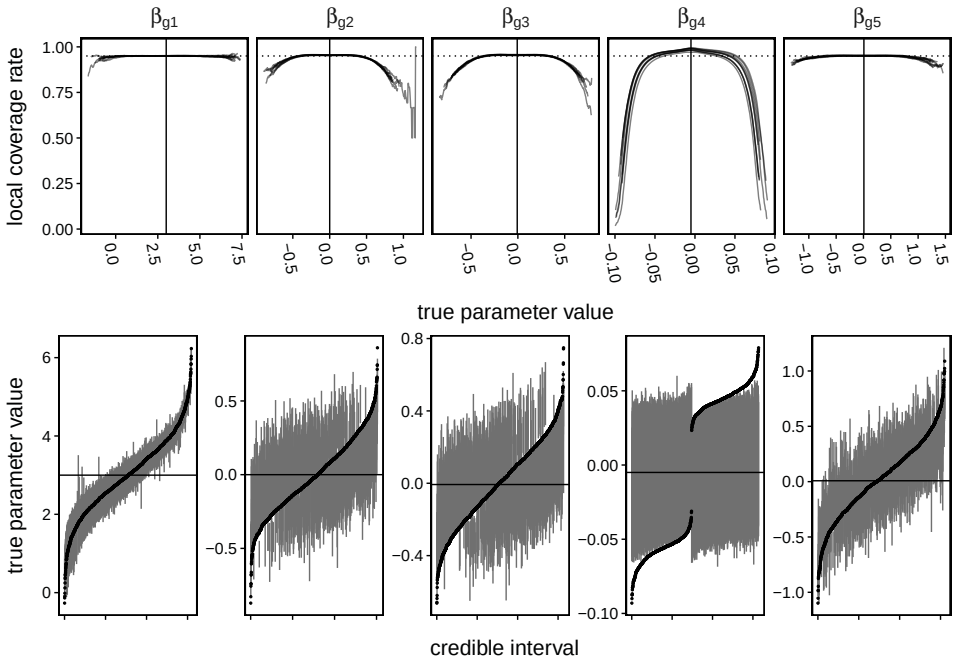

Figure 3.1: Posterior inference on the $\beta_{g\ell}$ parameters for Simulation Study 1 in Section 3.4.1. For each $\ell = 1, \ldots, 5$ and each dataset, the top row shows the kernel-smoothed local proportion of $\beta_{g\ell}$ parameters for which 95% CIs cover the true parameter values. The horizontal dashed lines are at 0.95, the desired coverage rate, and the solid black vertical lines indicate the respective true values of the hierarchical means θ_ℓ used to generate the count data. The bottom row shows, for $\ell = 1$ through 5 in the same order, the CIs (dark gray vertical lines) that do not cover the true parameter values (black points). Here, the solid black horizontal lines indicate the true hierarchical mean, θ_ℓ.

Finally, Figure S2 shows a receiver operating characteristic (ROC) curve for each kind of heterosis and each dataset. The results, all favorable, are extremely similar across datasets. With areas under the curves ranging from 0.916 to 0.922 for low-parent heterosis and from 0.930 to 0.936 for high-parent heterosis, our method competently filtered out the heterosis from the null genes. In addition, all the calibration curves in Figure S3 are extremely close to the identity line, so the estimated posterior probabilities of heterosis were extremely accurate and well-calibrated.

3.4.2 Robust comparison of full Bayes versus empirical Bayes

In RNA-seq analyses, empirical Bayes (eBayes) is relatively more common (Hardcastle and Kelly, 2010; Wu et al., 2012; Ji et al., 2014; Niemi et al., 2015) due to the reduced computational burden even though theoretically, eBayes procedures risk lower quality estimation and posterior inference by ignoring hyperparameter uncertainty. For this study, we considered two eBayes versions of our fully Bayesian approach: the *Oracle* approach fixed hyperparameters at the values used in data generation while the *Means* approach fixed hyperparameters at the posterior means estimated from the fully Bayesian approach. Thus, these methods provided a comparison under the best possible case for eBayes, and we did not address the question of how to obtain eBayes estimates of hyperparameters without running a fully Bayesian analysis.

For a robust comparison of our three methods, we simulated two datasets, one with $N = 16$ total replicates and the other with $N = 32$, under each of the three scenarios below. To generate counts, all scenarios used known values of the parameters $\beta_{g\ell}$, along with known γ_g's or negative binomial dispersions, depending on the data-generating mechanism. That way, parameter estimation and gene detection could be assessed as in Simulation Study 1 in Section 3.4.1.

Model Datasets were generated exactly as in Simulation Study 1. This was the only scenario where the true hyperparameter values were known, so it was the only scenario where we applied the Oracle empirical Bayes method.

edgeR In this scenario, the data-generating mechanism was a RNA-seq negative binomial generalized linear model that did not assume independence among $\beta_{g\ell}$ parameters for different genes. We obtained gene-specific parameter values by analyzing the Paschold et al. data using the `edgeR` R package (Robinson et al., 2010b). This software finds normalization factors to account for replicate-specific effects (Si and Liu, 2013; Anders and Huber, 2010; Robinson et al., 2010a), max-

imizes an adjusted profile likelihood to estimate negative binomial dispersions, and then uses maximum likelihood to find the parameters $\beta_{g\ell}$. Using the normalization factors, estimated negative binomial dispersions, and estimated $\beta_{g\ell}$ parameters as truth, we simulated counts using the model from `edgeR`.

Simple The purpose of this scenario was to produce datasets with exaggerated heterosis effects. The β_{g1} and β_{g5} parameter values were generated from normal distributions similar to their counterparts in the Model simulation. For $\ell = 2$, 3, and 4, the $\beta_{g\ell}$'s were drawn from discrete distributions in order to exaggerate the heterosis effect. We used $P(\beta_{g\ell} = 0) = 0.5$ and $P(\beta_{g\ell} = 1) = P(\beta_{g\ell} = -1) = 0.25$ for $\ell = 2$ and 3, $P(\beta_{g4} = 0) = 0.99$, and $P(\beta_{g4} = 1) = P(\beta_{g4} = -1) = 0.005$. All $\beta_{g\ell}$ parameters were generated independently across $g = 1, \ldots, G$ and $\ell = 1, \ldots L$. With the parameters in hand, count data were generated from a negative binomial model with a single common dispersion for all genes close in value to the dispersions obtained from the Paschold et al. dataset using `edgeR`. With respect to each of the six kinds of heterosis given in Section 3.2 and Table 3.1, roughly 6.5% of the simulated genes had some type of heterosis.

The fully Bayesian implementation was exactly the same as the previous simulation study with essentially the same results in terms of runtime, convergence diagnostics, and effective sample size for hyperparameters. The MCMC step of the empirical Bayes procedure was also performed using the software in the R packages `fbseq` and `fbseqCUDA` utilizing an option to skip sampling of hyperparameters. The runtime of this step was similar to the fully Bayesian analysis, e.g. up to 2.7 hours versus 3.2 hours for $N = 16$ and up to 4.3 hours versus 4.8 hours for $N = 32$.

Figure S4 shows the observed rates at which estimated 95% credible intervals cover parameters $\beta_{g\ell}$ for each method under comparison. The overall ability to capture these parameters in credible intervals hardly changed among methods. Overall, coverage was around the nominal 95% for the Model scenario, as well as for the Simple scenario, except

for slightly higher-than-nominal coverage of the β_{g4}'s. In the edgeR scenario, coverage was uniformly poor, ranging roughly from 50% to 90%.

The accuracy of estimated posterior heterosis probabilities also appeared nearly constant across methods. Figure 3.2 shows the calibration errors as defined in Section 3.4, which varied slightly between eBayes and fully Bayesian approaches. Calibration error was similar across sample sizes, but increased from the Model to the edgeR scenario and dramatically increased in the Simple scenario.

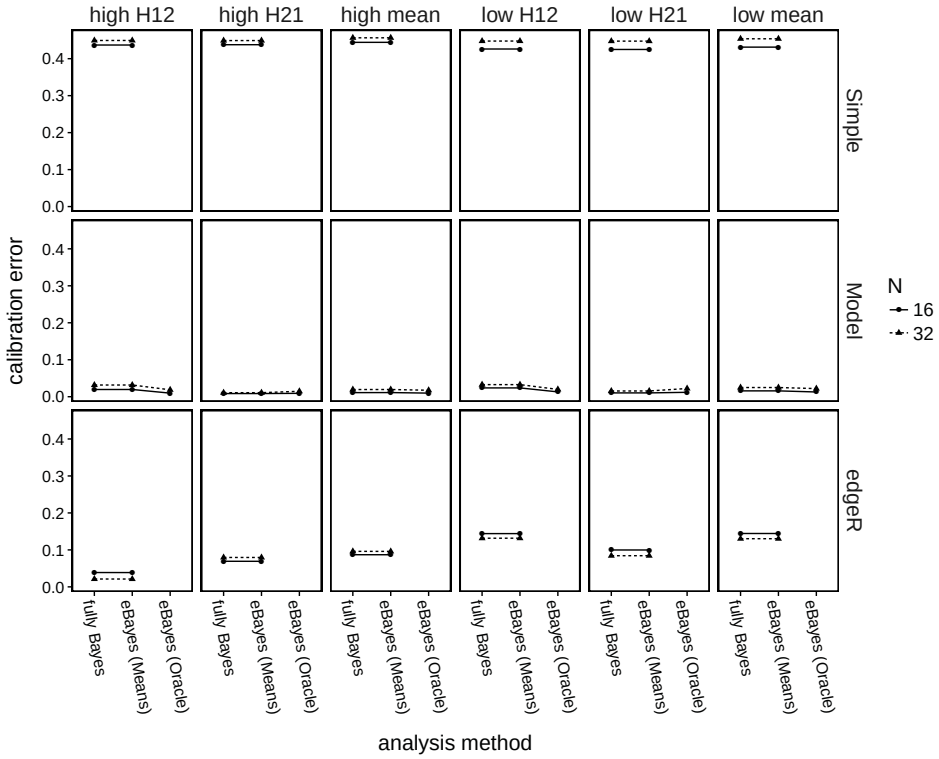

Figure 3.2: Mean absolute difference of each calibration curve in Figures 3.3 and S8 from the identity line. Triangle plotting symbols indicate simulated datasets with $N = 32$, and circles indicate datasets with $N = 16$. The type of heterosis is indicated above each column. For heterosis designations, we use the notation from the general plant breeding scenario from Section 3.2 and Table 3.1.

Figure 3.3 shows the calibration curves themselves for $N = 16$. (The results for $N = 32$, shown in Figure S8, are similar.) Compared to the Model scenario calibration was worse in the edgeR scenario, where many low probabilities were underestimated and high probabilities were overestimated for some types of high-parent heterosis. For the edgeR scenario, low-

parent heterosis probabilities tended to be overestimated overall. Calibration was egregiously poor in the Simple scenario, where posterior probabilities were heavily overestimated.

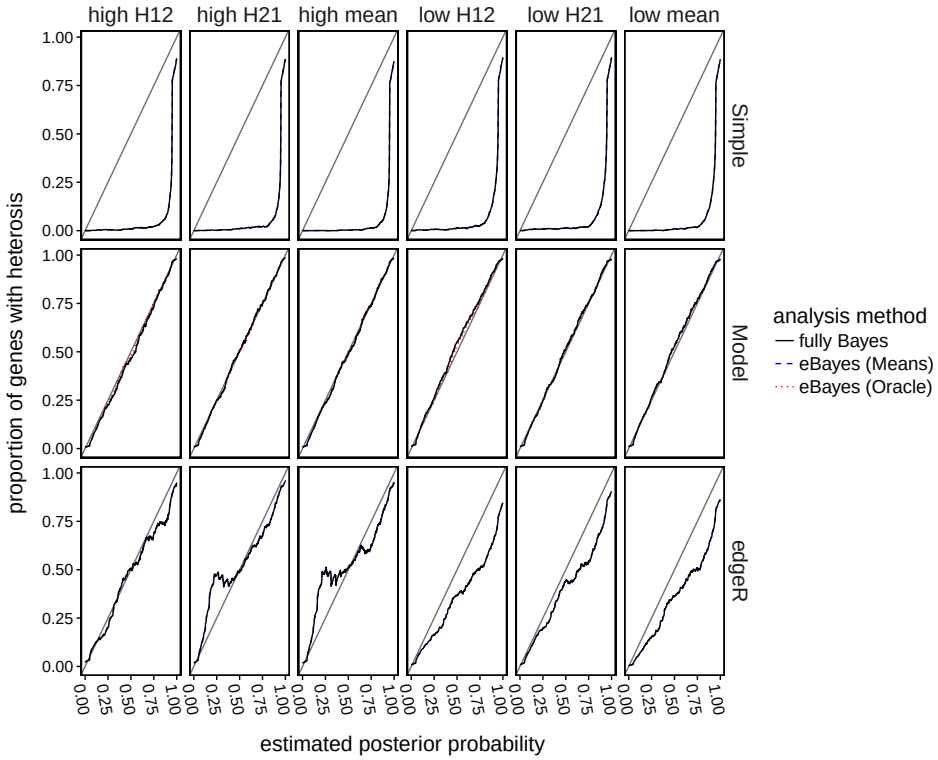

Figure 3.3: For datasets with $N = 16$ in Simulation Study 2 in Section 3.4.2, calibration curves for heterosis gene detection. The type of heterosis detected is indicated above each column, where we use the notation from the general plant breeding scenario from Section 3.2 and Table 3.1. The label to the right of each row indicates the method of simulating the data.

Figures S5 and S6 provide ROC curves for $N = 16$ and $N = 32$ while Figure S7 provides areas under the ROC curves (AUCs) for all simulations. The AUCs were around 0.85 (0.90) for edgeR, 0.94 (0.96) for Model, and 0.99 (0.99) for Simple with $N = 16$ ($N = 32$). The ROC curves and AUCs were almost identical for the fully Bayes and eBayes methods. Thus, despite a lack of coverage and poor calibration of posterior probabilities, the methodology appears to have provided reasonable rankings of genes even when the model assumed in the analysis disagreed with the data-generating mechanism.

3.5 A fully Bayesian analysis of the Paschold et al. dataset

Having assessed our methodology's estimation, inference, and gene detection abilities in the simulation studies in Section 3.4, we now turn back to the original motivating heterosis dataset in Section 3.2, where P1 is B73, P2 is Mo17, H12 is B73×Mo17, and H21 is Mo17×B73. The model matrix X and the interpretations of the parameters $\beta_{g\ell}$ are the same as in Section 3.4.

The dataset contains count data for $G = 39656$ genes on $N = 16$ biological replicates evenly spread over the four varieties. Roughly 7% of the genes have counts all equal to zero, 21% have mean counts less than 1, and 39% have mean counts less than 10. In other words, a large fraction of genes in the reference genome have low expression levels. Still, the mean count is around 255.5, the median is 37, the third quartile is 290, and the maximum is 38010. Figure S9 shows a kernel density estimate of the log of the counts after incrementing by 1. As the summary statistics suggests, the counts are multimodal, mainly split into low and high count groups.

We applied our fully Bayesian approach to the Paschold et al. dataset using the same number of chains, burn-in length, thinning, number of iterations, hardware, etc. as in Section 3.4.1, and the total elapsed runtime was 3.89 hours. As before, $\widehat{R}$ was less than 1.1 for all parameters except three $\beta_{g\ell}$ parameters and one γ_g parameter, and all the hyperparameter effective sample sizes were above 500.

Figure 3.4 shows posterior samples of all the hyperparameters. The marginal posterior distributions were approximately normal and extremely narrow, so uncertainty in these parameters was small. The marginal posteriors were so concentrated that the prior distributions, which were diffuse and uninformative, would just appear as horizontal lines near zero in the figure.

We also examined posterior samples of a random subset of parameters $\beta_{g\ell}$ in Figure 3.5. Along with kernel density estimates, the figure shows the respective approximate normal densities computed from the posterior means and mean squares of the samples. For each

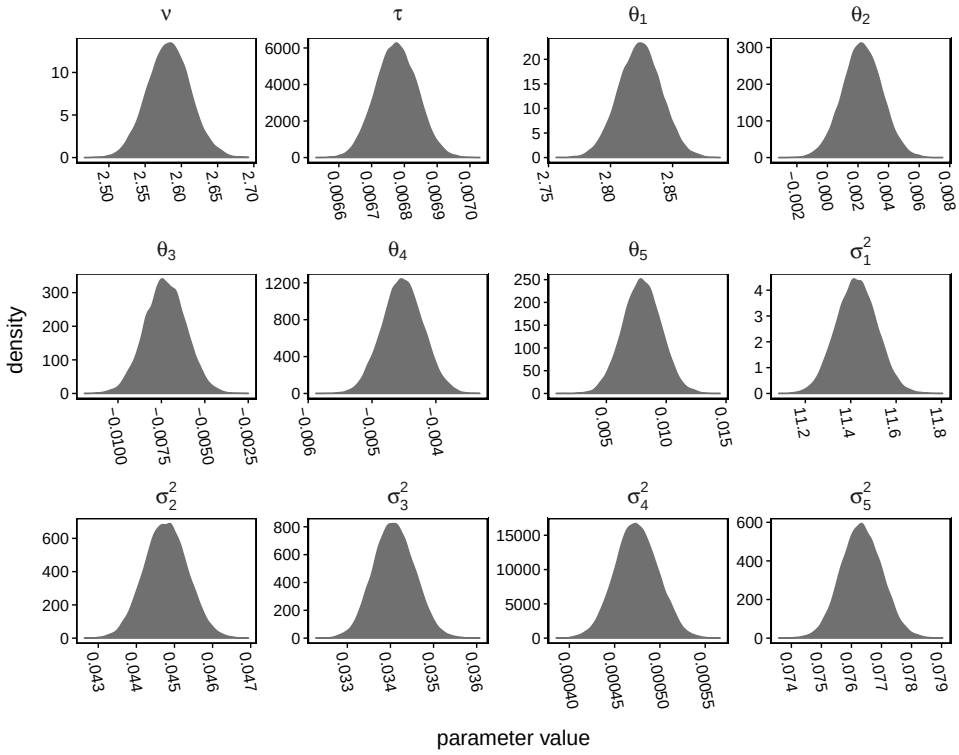

Figure 3.4: For the Paschold et al. data analyzed with the fully Bayesian approach in Section 3.5, kernel density estimates from MCMC samples of the hyperparameters.

parameter, the normal approximation closely matched the kernel density estimate, as did the equal-tail 95% CIs computed from each. This finding justifies the computational strategy recommended by Chapter 2, which, for the sake of computational tractability, discarded most MCMC parameter samples and retained only the estimated posterior means and mean squares of these parameters.

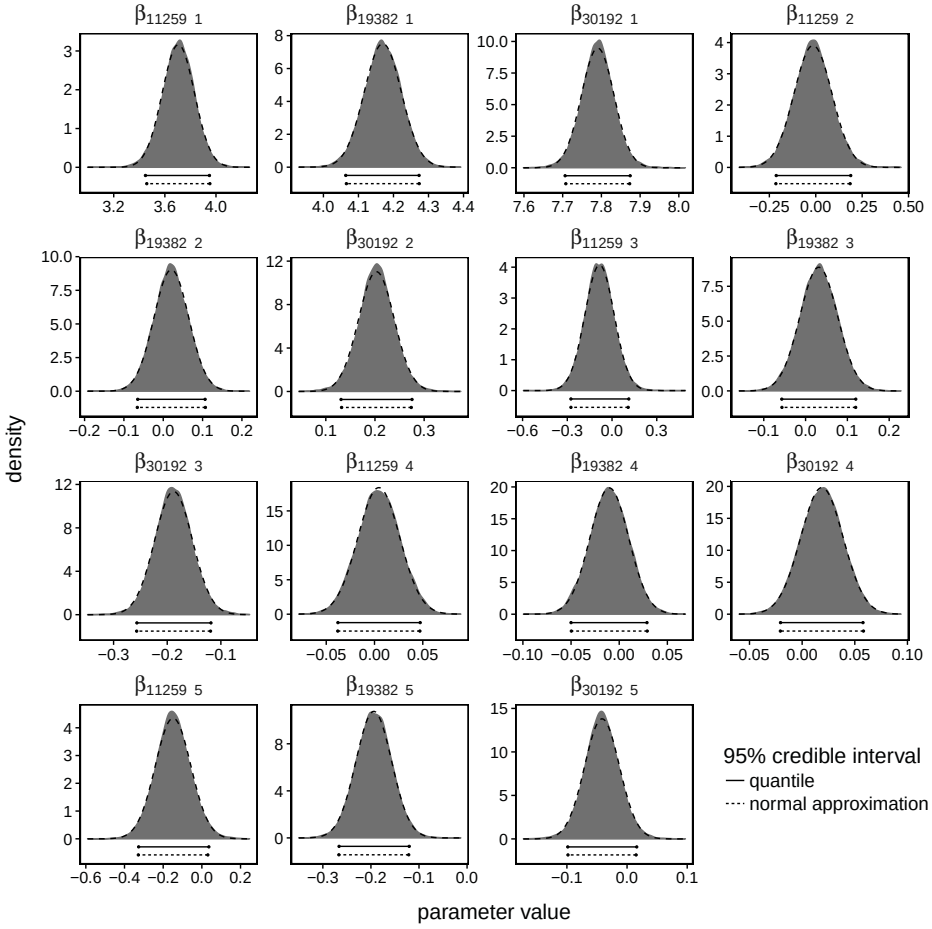

Figure 3.5: For the Paschold et al. dataset analyzed with the fully Bayesian approach in Section 3.5, kernel density estimates of the marginal posterior distributions of a small random subset of $\beta_{g\ell}$ parameters with estimated normal densities (dashed lines) based on mean and mean square of the MCMC parameter samples. The horizontal solid lines are 95% equal-tail credible intervals based on MCMC samples (solid) and normal approximation (dashed).

Figure 3.6 shows the estimated posterior probabilities of each kind of heterosis. Most probabilities are below 0.5, and there is a spike at 0 for each kind of heterosis, so gene-specific heterosis appears uncommon overall. In addition, there is a spike around 0.25 in

each histogram, which corresponds to unexpressed and barely expressed genes. The value 0.25 is the estimated predictive probability for a new gene $\tilde{g}$

$$P\left(\beta_{\tilde{g}2} > 0 \text{ and } \beta_{\tilde{g}3} > 0\right) \approx P(\beta_{\tilde{g}2}/\sigma_2 > 0)P(\beta_{\tilde{g}3}/\sigma_3 > 0) = 0.25$$

since θ_2 and θ_3 are close zero (see Figure 3.4), β_{g2} and β_{g3} are assumed independent, and the probability that a bivariate, independent normal is in the positive quadrant is 0.25.

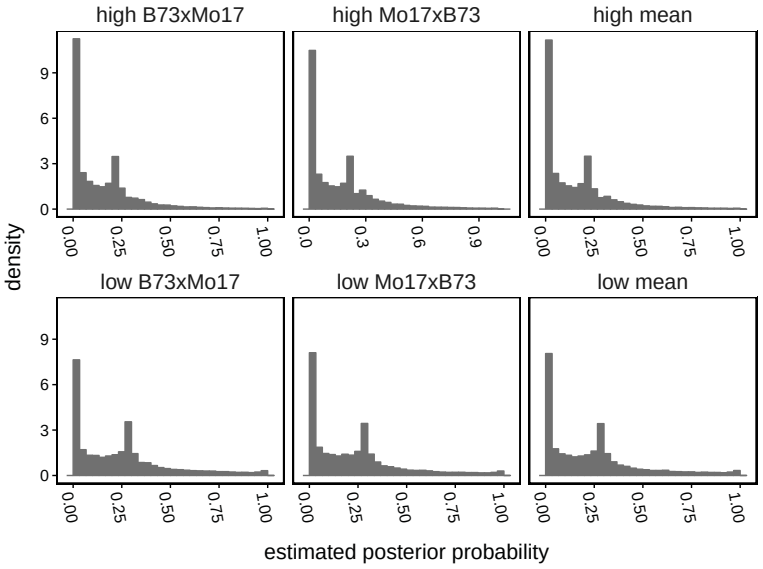

Figure 3.6: Histograms of estimated posterior probabilities of high (top row) and low (bottom row) heterosis for the B73×Mo17 hybrid (left column) and Mo17×B73 hybrid (middle column), and their mean (right column).

Without an objective standard to measure the correctness of our heterosis probabilities, we compared them to the equivalent gene detection results from the original paper (Paschold et al., 2012). Figure 3.7 showed the same posterior probabilities of heterosis as in Figure 3.6 (except for the "high mean" and "low mean" cases), but for this study, we separated genes by whether Paschold et al. labeled them heterosis genes in the "Table 1" section of their Supplementary Table S3. Overall, we saw general agreement between our method and theirs: probabilities are generally high for their discoveries and low for their null genes. However, there was also disagreement: low probabilities for some of their discoveries and high probabilities for some of their nondiscoveries.

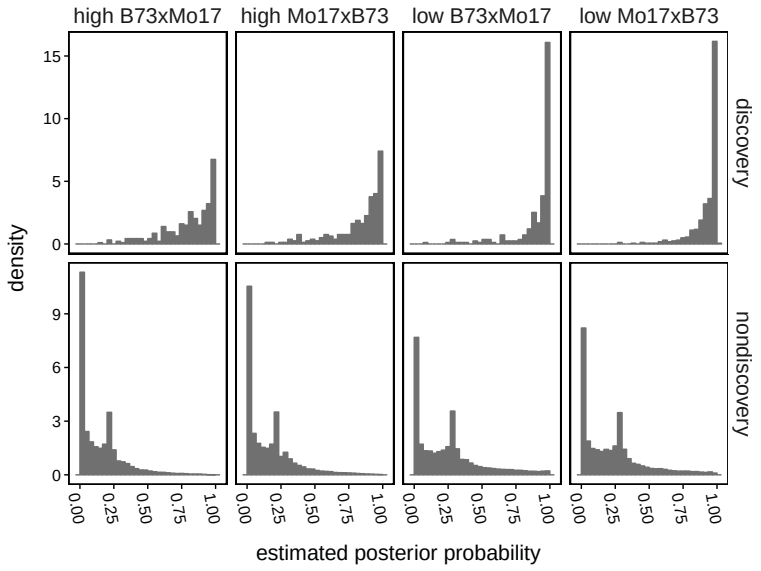

Figure 3.7: Histograms of fully Bayesian estimated posterior probabilities for the four heterosis types (columns) in Paschold et al. (2012) split by discovery (top row) and non-discovery (bottom row) as determined by Paschold et al. (2012).

3.6 Discussion

We presented a fully Bayesian strategy for modeling high-dimensional count data, a rare approach in fields such as RNA-sequencing data analysis due to the computational challenges. A fully Bayesian analysis of our data with our model would have been computationally intractable with conventional serial computing. However, our method proved solidly tractable because we used a massively parallelized Markov chain Monte Carlo algorithm designed to take advantage of the power of general-purpose graphics processing units. We applied our approach to the heterosis problem in RNA-sequencing data analysis, and we used simulation studies to assess the fully Bayesian approach and compared it to two empirical Bayes counterparts. From our simulations, we found that our fully Bayesian method strongly shrunk estimates of important gene-specific parameters towards common means, and although the method competently ranked genes in order of importance, the calibration of inferential quantities, such as posterior probabilities, suffered when the model's assumptions were not satisfied. The fully Bayesian and empirical Bayes methods performed equally well under metrics of gene detection ability, a finding that supports the use of empirical Bayes

methods in RNA-seq analyses. Finally, we turned back to the motivating RNA-seqdataset by Paschold et al. (2012), and we showed that our results generally agree with those of the original Paschold et al. paper.

Simulation Study 2 in Section 3.4.2 showed that with good enough hyperparameter values, our fully Bayesian approach is practically equivalent to its empirical Bayes counterpart. This finding is likely due to the fact that, as seen in Figure 3.4, the hyperparameters had extremely low uncertainty and are well-approximated by point masses. The low uncertainty, in turn, is likely due to the large number of genes used to estimate the hyperparameters. Our finding appears to justify the widespread use of empirical Bayes in RNA-seq (Niemi et al., 2015; Ji et al., 2014; Hardcastle and Kelly, 2010) as an approximation to a fully Bayesian procedure. However, we only used empirical Bayes in the best-case scenario: that is, when hyperparameters are fixed either at the true values used to generate the data or at estimated posterior means from a fully Bayesian analysis of the same dataset. If hyperparameter values are poor, the performance of empirical Bayes may still suffer. Determining a suitable hyperparameter estimation method for empirical Bayes is beyond the scope of this article, but such a technique could increase computational efficiency. We saw small reductions in runtime of empirical Bayes relative to fully Bayes, and these efficiency gains may improve if only a small subset of genes are analyzed.

Our results suggest some possible improvements to our model for future work. From Figure 3.1, the gene-specific parameters $\beta_{g\ell}$ were poorly estimated if their true values are extreme for a given index element ℓ, Specifically, low $\beta_{g\ell}$'s tended to be overestimated and high $\beta_{g\ell}$'s tended to be underestimated, i.e. the estimates of extreme $\beta_{g\ell}$ parameters were overly shrunk towards their hierarchical means. In addition, from Figure S4, overall estimation was poor when the model used to simulate data disagreed with the model used in the analysis. If we assumed hierarchical distributions with heavier-tailed distributions, such as Laplace, Student t, or horseshoe (Carvahlo et al., 2009) distributions, shrinkage should relax for extreme $\beta_{g\ell}$'s, and overall estimation, inference, and gene detection could improve.

We glean other potential improvements from Figure 3.6, in which the genes with the lowest counts are responsible for the probability spikes around 0.25. Genes with zero counts are not expressed at all and thus show no heterosis, yet our method assigned them heterosis probabilities of roughly 0.25. A point-mass mixture distribution, such as an additive mixture of a point mass at zero and a Student t distribution, may mitigate this issue. With point-mass mixture hierarchical distributions for $\beta_{g\ell}$'s with ℓ =2, 3, and 4, the model is free to give low-count genes heterosis probabilities of zero. This change to the model could also potentially improve estimation, inference, and gene detection. However, the computational cost may also increase. Given the large number of genes, it may be difficult for the Markov chains to traverse both components of each mixture distribution in each dimension.

One of our model's weaknesses is the poor accuracy of posterior probabilities when the model assumed in analysis is not the data-generating model. Calibration was worst for the Simple scenario in Simulation Study 2 (Section 3.4.2), in which our model overestimated probabilities. Perhaps one of the potential changes suggested above would mitigate calibration error, but the resulting model may still have assumptions that the data are likely to violate. For more flexibility and robustness, a semi-parametric approach, e.g. a Dirichlet process mixture (Liu et al., 2015; Muller and Mitra, 2013), could be employed to learn the distribution of the gene-specific parameters. Such a semi-parametric approach may also relax the assumptions, currently built into the model's hierarchical distributions, that gene-specific quantities are independent conditional on the hyperparameters.

Lacking complete certainty regarding the heterosis status of the genes in the Paschold et al. dataset, our real data analysis did not definitively determine whether our approach outperformed the one that Paschold et al. used. However, we suspect that our method had increased parameter estimation and gene detection abilities. The Paschold et al. analysis treated genes independently, while ours used a multilevel hierarchical model to borrow information across genes, and from Landau and Liu (2013), borrowing information tends to improve both estimation and detection. Altering an existing method to borrow informa-

tion can be cumbersome and complicated, but hierarchical modeling and Bayesian inference already accomplish this cleanly and elegantly without any additional effort.

3.7 Supplementary Materials

The supplementary figures are included in Appendix C, and the supplementary table for this chapter, Table S1, is available separately. Table S1 is a comma-separated values spreadsheet containing the total per-replicate counts of the Paschold et al. (2012) data, as well as posterior estimates from the fully Bayesian approach of the gene-specific heterosis probabilities, the means of the design parameters $\beta_{g\ell}$ and hierarchical means γ_g, and the standard deviations of the $\beta_{g\ell}$'s and γ_g's. In addition, the table contains information about which genes were classified as heterosis genes by Paschold et al. (2012), taken from the "Table 1" section of their Table S3.

CHAPTER 4. IMPROVING THE RNA-SEQ MODEL

In the analysis of high-dimensional count data, particularly datasets from RNA-sequencing experiments, new work is emerging on fully Bayesian methodology made tractable with high-performance computing. In one such approach from Chapter 3, a massively-parallelized slice-sampling-within-Gibbs Markov chain Monte Carlo algorithm estimates the full joint posterior distribution of a multilevel hierarchical model, where embarrassingly parallel Gibbs steps and parallelized reductions drive the acceleration. From past simulation studies, this approach suffers from poor estimation of extreme-valued gene-specific parameters, a lack of robustness in parameter estimation, and poor calibration of estimated gene-specific posterior probabilities. To address these issues, we use a scale mixture of normals to vary the marginal hierarchical distributions of key gene-specific parameters, and we compare the resulting models using simulation studies. We find that parameter estimation improves slightly for the newer models, but calibration does not.

4.1 Introduction

The hierarchical model from Chapter 3 was part of a fully Bayesian method for analyzing data from RNA-sequencing (RNA-seq) experiments. Parameter estimation and inference were accomplished with a slice-sampling-within-Gibbs Markov chain Monte Carlo algorithm accelerated with general purpose graphics processing unit (GPU) computing. The method was used to analyze a maize RNA-seq dataset by Paschold et al. (2012), who investigated heterosis at the level of gene expression, and the heterosis gene detection results generally agreed with those of Paschold et al. Simulation studies based on the plant breeding scenario of the Paschold dataset assessed the estimation, inference, and gene detection abilities of the method.

The purpose of this article is to address two major weaknesses exposed in the simula-

tion studies from Chapter 3. The first weakness was the estimation of the model coefficient parameters $\beta_{g\ell}$. Specifically, low $\beta_{g\ell}$'s tended to be overestimated and high $\beta_{g\ell}$'s tended to be underestimated, shrinking too heavily toward hierarchical means. In addition, when the data-generating mechanism disagreed with the model assumed in the analysis, the overall coverage of the $\beta_{g\ell}$ parameters suffered. The second major weakness we consider is the egregiously poor calibration (accuracy) of estimated posterior probabilities of gene-level heterosis when the data-generating mechanism disagreed with the analysis model. Even when these probabilities provided effective rankings of genes in order of importance, the estimates were still systematically overestimated in some cases and systematically underestimated in others.

In an attempt to address these two weaknesses, we focus on the hierarchical distributions of the parameters $\beta_{g\ell}$. In its original formulation by Chapter 3, the model assigned normal distributions to the $\beta_{g\ell}$ parameters. Because of their thin tails, we initially suspect that these normals may have contributed to the heavy $\beta_{g\ell}$ shrinkage, poor robustness of estimation, and bias in the posterior probability estimates when the model in the analysis and data-generating mechanism disagreed.

Our proposed solution is to change the hierarchical distributions of the $\beta_{g\ell}$ parameters. Rather than exclusively using normal distributions, we assign Laplace distributions and Student t distributions to some of the $\beta_{g\ell}$ parameters. These heavier-tailed distributions may relax shrinkage of extreme $\beta_{g\ell}$'s and improve flexibility and robustness.

The rest of the paper expounds the updated model and contrasts it with the one by Chapter 3 in terms of results from analyzing simulated data. Section 4.2 describes our fully Bayesian methodology. Section 4.2.1 introduces the new model, some derivations of which are relegated to Appendix B, and Section 4.2.2 reviews the Markov chain Monte Carlo method used for estimation and inference. Section 4.3 describe our simulation studies, which focused on determining if the changes to the model mitigated the original shortcomings. Section 4.3.1 lays out a simulation study in a two-group RNA-seq scenario to highlight the effects of the different hierarchical distributions. Finally, Section 4.3.2 shows how the new model

was used to analyze the exact simulated datasets from the simulation studies by Chapter 3 in order to investigate the behavior of the altered model relative to the old one in terms of estimation, inference, and heterosis gene detection.

4.2 Fully Bayesian methodology

4.2.1 The altered hierarchical RNA-seq model and priors for a Bayesian analysis

The original model, expounded by Chapter 2 and Chapter 3, was similar to negative binomial regression models from other RNA-seq methods (McCarthy et al., 2012; Wu et al., 2012). Whereas these other methods mixed a Poisson distribution over a gamma distribution, the model by Chapter 2 and Chapter 3 mixed a Poisson distribution over a log-normal. In the model, y_{gn} was defined as the RNA-seq count of gene g ($g = 1, \ldots, G$) in replicate n ($n = 1, \ldots, N$), and y was taken to be the G by N matrix of the y_{gn}'s. The model matrix X connected the N columns of the count data matrix y to elements of the model parameterization, and because X was a general N by L matrix, the model generalized beyond the general plant hybrid scenario used by Chapter 3.

In the original model, RNA-seq counts y_{gn} were given conditionally independent Poisson $(\exp(h_n + \varepsilon_{gn} + X_n\beta_g))$ distributions (where X_n was the n'th row of the model matrix, $\beta_g = (\beta_{g1}, \ldots, \beta_{gL})$), and the ε_{gn}'s were given conditionally independent Normal$(0, \gamma_g)$ distributions. That is, Poisson distributions were mixed over log-normal distributions to account for gene-specific overdispersion, a similar approach to the negative binomial models from other RNA-seq data analysis methods (McCarthy et al., 2012; Wu et al., 2012). The h_n's were constants estimated from the data, and they played the role of normalization factors in other RNA-seq models, taking into account sequencing depth and other nuisance effects (Si and Liu, 2013; Anders and Huber, 2010; Robinson et al., 2010a). The γ_g's played the role of gene-specific overdispersion parameters, analogous to the negative-binomial dispersions used in other RNA-seq methods (Landau and Liu, 2013). The γ_g's were assigned conditionally

independent Inverse-Gamma $(\nu/2, \nu\tau/2)$ distributions, where τ was a prior measure of center and ν was the degree to which the γ_g's shrunk towards τ. Also in keeping with Chapter 3, $\tau \sim \text{Gamma}(a, \text{rate} = b)$ and $\nu \sim \text{Uniform}(0, d)$, where constants $a = 1$, $b = 1$, and $d = 1000$ were set to make the priors diffuse (Gelman, 2006).

Our changes to the above model are with respect to the hierarchical distributions of the components $\beta_{g\ell}$ of $\beta_g = (\beta_{g1}, \ldots, \beta_{gL})$. Whereas the previous model assigned the $\beta_{g\ell}$'s conditionally independent Normal$(\theta_\ell, \sigma_\ell^2)$ distributions, the model for this article uses conditionally independent Normal$(\theta_\ell, \sigma_\ell^2 \xi_{g\ell})$ distributions. We still have prior centers θ_ℓ with conditionally independent Normal$(0, c_\ell^2)$ distributions and prior standard deviations σ_ℓ with conditionally independent Uniform$(0, s_\ell)$ distributions (where constants $c_\ell = 10$ and $s_\ell = 100$ $(\ell = 1, \ldots, L)$ are set so that these priors are diffuse). This time, however, we have new auxiliary parameters $\xi_{g\ell}$, to which we assign prior distributions $p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell)$, where $k_\ell = 1$, $q_\ell = 3$, and $r_\ell = 2$ are fixed constants for $\ell = 1, \ldots, L$. This creates a convenient *scale mixture of normals* (Carvahlo et al., 2009). By selecting the appropriate $p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell)$, we control $p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2)$ (i.e., $\int p(\beta_\ell|\theta_\ell, \sigma_\ell^2, \xi_{g\ell}) p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell) d\xi_{g\ell}$), which we call the *marginal hierarchical distribution* of $\beta_{g\ell}$. When the prior $p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell)$ is a point mass at 1, $p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2)$ is just Normal$(\theta_\ell, \sigma_\ell^2)$, and our model reduces to the model from Chapter 3. On the other hand, when $p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell)$ is Exp$(k_\ell = 1)$, then integrating out the $\xi_{g\ell}$'s gives

$$\begin{aligned} p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2) &= \int \text{Normal}(\theta_\ell, \sigma_\ell^2 \xi_{g\ell}) p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell) d\xi_{g\ell} \\ &= \text{Laplace}\left(\text{location} = \theta_\ell, \text{scale} = \sqrt{\frac{\sigma_\ell^2}{2k_\ell}}\right) \end{aligned}$$

i.e., a Laplace distribution with mean θ_ℓ. In the implementation, we choose $k_\ell = 1$ so that the variance is σ_ℓ^2. Similarly, when $p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell)$ is Inverse-Gamma$(q_\ell = 3, r_\ell = 2)$, then

$$\begin{aligned} p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2) &= \int \text{Normal}(\theta_\ell, \sigma_\ell^2 \xi_{g\ell}) p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell) d\xi_{g\ell} \\ &= t_{2q_\ell}\left(\text{location} = \theta_\ell, \text{scale} = \sqrt{\frac{\sigma_\ell^2 r_\ell}{q_\ell}}\right) \end{aligned}$$

i.e., a Student t distribution with $2q_\ell$ degrees of freedom and median θ_ℓ. In the implementation, we use $q_\ell = 3$ and $r_\ell = 2$ so that there are six degrees of freedom and the variance is σ_ℓ^2. See Appendix B for the derivation of $p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2)$ for each choice of prior distribution on $\xi_{g\ell}$.

4.2.2 Inference on gene-specific parameters and heterosis probabilities

Before parameter estimation, we calculated the log-scale replicate-specific normalization constants h_n using the method from Chapter 3. With the log-scale counts $w_{gn} = \log(y_{gn} + 0.5 \cdot \mathrm{I}(y_{gn} = 0))$, replicate-specific means $\overline{w}_{.n} = \frac{1}{G}\sum_{g=1}^{G} w_{gn}$, and the grand mean $\overline{w}_{..} = \frac{1}{N}\sum_{n=1}^{N} \overline{w}_{.n}$, we set $h_n = \overline{w}_{.n} - \overline{w}_{..}$ for $n = 1, \ldots, N$.

Next, to estimate the full joint posterior distribution of the parameters, we used the GPU-accelerated slice-sampling-within Gibbs Markov chain Monte Carlo algorithm by Chapter 2. By executing conditionally independent Gibbs steps in parallel and using parallelized reductions to compute the full conditional distributions of the hyperparameters, the algorithm accelerated the MCMC computation to a solidly tractable speed. Due to the severe memory transfer bottleneck in GPU computing, we limited data transferred from the GPU to the CPU as recommended by Chapter 2, so we only retain parameter samples for hyperparameters and a random subset of gene-specific parameters. For every parameter ψ, we also recorded $\overline{\psi} = \frac{1}{M}\sum_{m=1}^{M} \psi^{(m)}$ and $\overline{\psi^2} = \frac{1}{M}\sum_{m=1}^{M} \left(\psi^{(m)}\right)^2$, where $\psi^{(m)}$ was the m^{th} Monte Carlo sample of ψ, and approximated $p(\psi|y)$ with $N\left(\overline{\psi}, \overline{\psi^2} - \overline{\psi}^2\right)$. Finally, we assessed the posterior probabilities of gene-specific conditions using ergodic averages involving parameter samples. An example of such a probability was

$$P\left(\beta_{g2} > c|y\right) \approx \frac{1}{M}\sum_{m=1}^{M} \mathrm{I}\left(\beta_{g2}^{(m)} > c\right) \tag{4.1}$$

where c is a constant and $\mathrm{I}(A)$ is 1 if A is true and 0 otherwise. In this example, the probability above is the probability that gene g has high-parent heterosis in its expression level with respect to the H12 hybrid from Chapter 3.

For each analysis of each dataset, we ran four overdispersed chains, which collectively totaled 2×10^5 Monte Carlo iterations. We monitored those chains for convergence using Gelman-Rubin potential scale reduction factors $\widehat{R}$ (Gelman and Rubin, 1992), which we calculated using estimated posterior means and mean squares (see Chapter 2). Specifically, we monitored $\widehat{R}$ on the $2L + 2$ hyperparameters $(\nu,\ \tau,\ \theta_1, \ldots, \theta_L,\ \sigma_1, \ldots, \sigma_L)$, the $G \times L$ parameters $\beta_{g\ell}$, and the G hierarchical variance parameters γ_g. In our experience, we found $\widehat{R}$ values near one for all but a few gene-specific parameters that vary when rerunning the MCMC. We also monitored convergence using the effective sample sizes of the hyperparameters (since we retained Monte Carlo hyperparameter samples), which we generally found to be well above the 10 to 100 effective samples recommended by Gelman et al. (2013).

For computation, we used the R packages `fbseq` and `fbseqCUDA`, mentioned by Chapter 3 and publicly available on GitHub. We also added to the `fbseqStudies` R package, previously released by Chapter 3, to replicate all the results of this paper. The `fbseqStudies` package is publicly available through the GitHub repository of the same name. Running the `paper_priors()` function in `fbseqStudies` reproduces the computation, figures, tables, etc. shown in all the following sections. For hardware, we used a single NVIDIA K20 GPU, two 2.0 GHz 8-Core Intel E5 2650 processors, and 64 GB of memory.

4.3 Simulation studies

We used simulation studies to assess the merits of the altered RNA-seq model in 4.2.1 relative to its predecessor from Chapter 3. Specifically, the goal was to find out if the altered, heavier-tailed hierarchical distributions improved inference on the $\beta_{g\ell}$ parameters or the calibration of posterior probabilities. In the assessment, in addition to credible intervals of the parameters $\beta_{g\ell}$, we used calibration curves described by Chapter 2. A calibration curve plots the true status of each simulated gene against its estimated posterior probability of being a gene of interest, and kernel smoother is used so that the vertical axis shows the local proportion of true genes of interest. In addition to the calibration curves themselves,

we examined calibration error, the mean absolute vertical distance from the identity line.

4.3.1 Two-group simulation study

To simulate the data, we used a simple model matrix in order to focus on differences in $\beta_{g\ell}$ estimation when the hierarchical distribution was varied for a single ℓ. We simulated multiple datasets, each with with $N = 8$ replicates divided evenly between two groups and $G = 10000$ genes. The $N = 8$ by $L = 2$ model matrix X is compactly represented as

$$X = \begin{bmatrix} 1 & -1 \\ & \\ 1 & 1 \end{bmatrix} \otimes J_{4\times 1}$$

where "$\otimes$" denotes the Kronecker product and $J_{m\times n}$ is the $m \times n$ matrix with all entries equal to 1. With this X, for each gene g, β_{g1} is the log-scale mean and β_{g2} is the log-scale half-difference between the two groups.

When we generated datasets from the model, the hyperparameters underlying the simulation were similar to simulations from the model given by Chapter 3 for the plant breeding scenario, except that here, we used $\theta_1 = 3, \theta_2 = 0, \sigma_1^2 = 1$, and $\sigma_2^2 = 0.05$. Conditioning on these fixed hyperparameters, we generated the $\gamma_g's$, ε_{gn}'s, and $\beta_{g\ell}$'s using hierarchical distributions, and then generated counts y_{gn} using the gene-specific parameters and $h_n = 0$ for $n = 1, \ldots, N$. This is similar to the simulations from the model by Chapter 3, except that we varied the hierarchical distributions of the β_{g2}'s so that 10 datasets were generated using Normal(θ_2, σ_2^2) distributions, 10 using Laplace(mean $= \theta_2$, variance $= \sigma_2^2$), and 10 using t_6(mean $= \theta_2$, variance $= \sigma_2^2$). We call these simulation scenarios the normal, Laplace, and t scenarios, respectively.

To analyze each simulated dataset, we fit three versions of our model: one with normal distributions on the β_{g2} parameters, one with Laplace distributions, and one with shifted and scaled t_6 distributions. In this section, we call these models the normal, Laplace, and t models, respectively. In all cases, we set $h_n = 0$ for $n = 1, \ldots, N$ instead of estimating the

h_n's from the data in order to force the model assumed in the analysis to agree with the data-generating mechanism. For gene-specific inferential quantities, we focused on detecting genes with the highest expression levels in group 2 relative to group 1: i.e., genes for which β_{g2} was highest. Thus, we calculated gene-specific posterior probabilities $P(\beta_{g2} > c|y)$. In our case, we picked c to be the 90th percentile of the hierarchical Normal(0, 0.05) distribution for the β_{g2}'s (roughly 0.29) so that roughly 10% of the genes were deemed important.

We analyzed each dataset using the implementation and resources described in Section 4.2.2, and the overall maximum runtime for any analysis was about 42 minutes. For the majority of analyses, all computed Gelman factors fell below the commonly-recommended tolerance threshold of 1.1, indicating apparent convergence. The other 25 analyses each had no more than three Gelman factors above 1.1, all corresponding to $\beta_{g\ell}$ parameters, the highest of which was around 4.46 ($\beta_{2108,2}$, t simulation, t analysis). The minimum hyperparameter effective sample size over all analyses was around 3200 (ν, the t simulation, normal analysis). Convergence may not have been strictly reached, but estimation and inference should still be dependable for all but a small handful of poorly-analyzed genes.

One of our original goals was to mitigate the heavy shrinkage on the extreme values of the parameters $\beta_{g\ell}$. To assess shrinkage, we first examined estimated credible intervals of the β_{g2}'s. For one example dataset from each simulation scenario, Figure 4.1 shows example 95% credible intervals of β_{g2}'s that did not cover the true parameter values. The figure shows that in all cases, credible intervals shrunk towards the overall mean of the β_{g2}'s, which is consistent with the results of Chapter 3. The shrinkage appears to have varied little among the different analyses.

Figure 4.2 shows credible interval information for whole simulation study. For each dataset, the figure plots the local (kernel-smoothed) rate at which the 95% credible intervals covered the true β_{g2} values used to simulate the counts. As with Chapter 3, coverage was higher near the β_{g2} means and lower for extreme values. However, in all simulation scenarios, this loss in coverage appears to have been less severe for the non-normal analyses. The

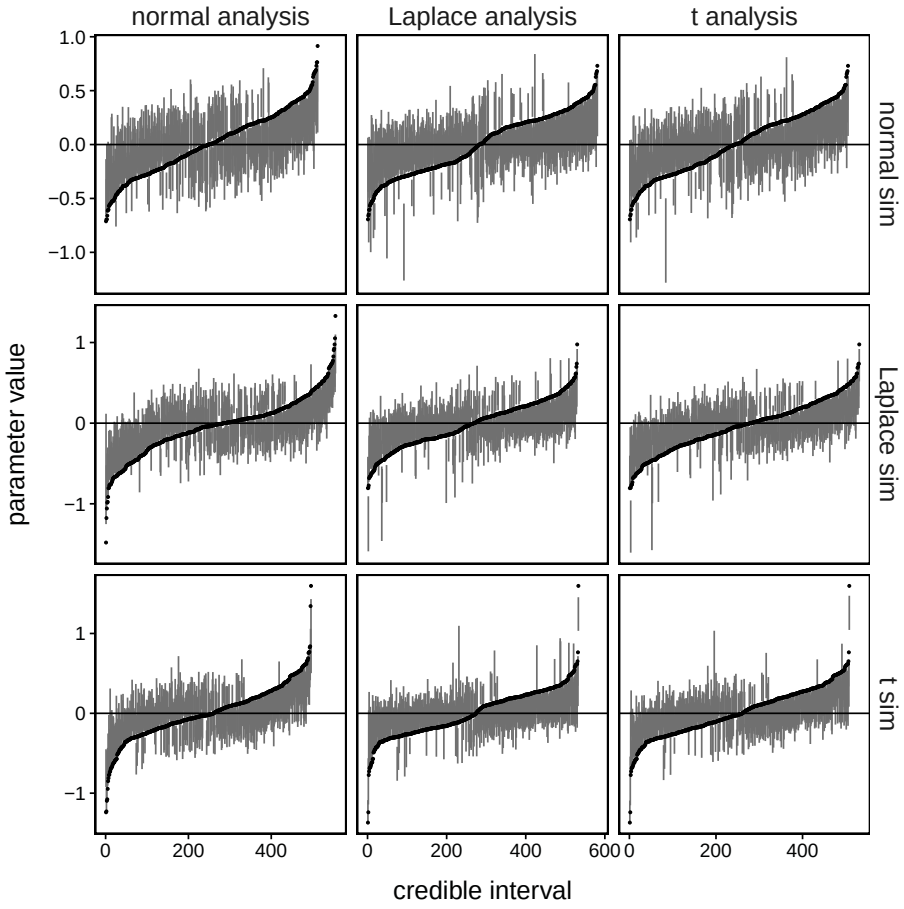

Figure 4.1: For three example datasets from the two-group simulation study in Section 4.3.1, estimated 95% credible intervals of the β_{g2} parameters that did not cover the true parameter values. The row labels indicate the simulation scenarios of the datasets, and column labels indicate the models used to analyze the data. Credible intervals are vertical dark gray lines, and the true values of the β_{g2}'s used to generate counts are shown as black dots in each panel. The horizontal black lines indicate zero, the true value of the hierarchical mean θ_2 used to generate count data.

Laplace and t hierarchical distributions on β_{g2} have apparently improved coverage of extreme $\beta_{g\ell}$'s in credible intervals relative to the normal distributions.

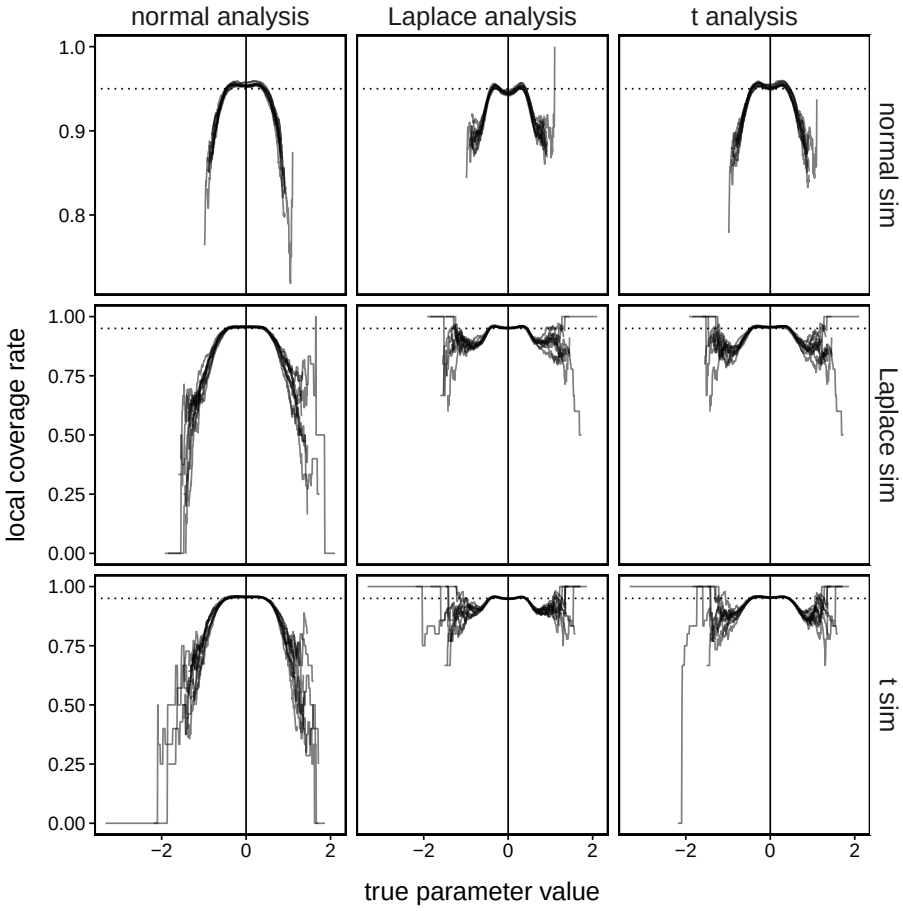

Figure 4.2: For the two-group simulation study in Section 4.3.1, local (kernel-smoothed) rates at which 95% credible intervals for the β_{g2}'s cover the true values used to simulate count data. The horizontal dotted lines are at 0.95, the desired coverage rate. The column labels indicate the models used for the analyses, and the row labels indicate the simulation scenario. Each semitransparent black curve corresponds to an analysis of a simulated dataset. The vertical solid black lines indicate zero, the true value of the hierarchical mean θ_2 used to generate the count data.

To address the poor robustness of posterior probability calibration from Chapter 3, Figure 4.3 shows calibration curves corresponding to the posterior probabilities $P(\beta_{g2} > c|y)$ (where c is around 0.29) described previously. From the individual calibration curves, accuracy appears high overall, and similar among different simulation scenarios and analyses. The corresponding calibration errors, shown in Figure 4.4, do show some slight systematic differ-

ences, however. As expected, calibration was systematically worse when the data-generating mechanism disagreed with the model assumed in the analysis. However, these differences in calibration were not severe, possibly because the three data-generating models were similar, only differing in the hierarchical distributions of the β_{g2}'s.

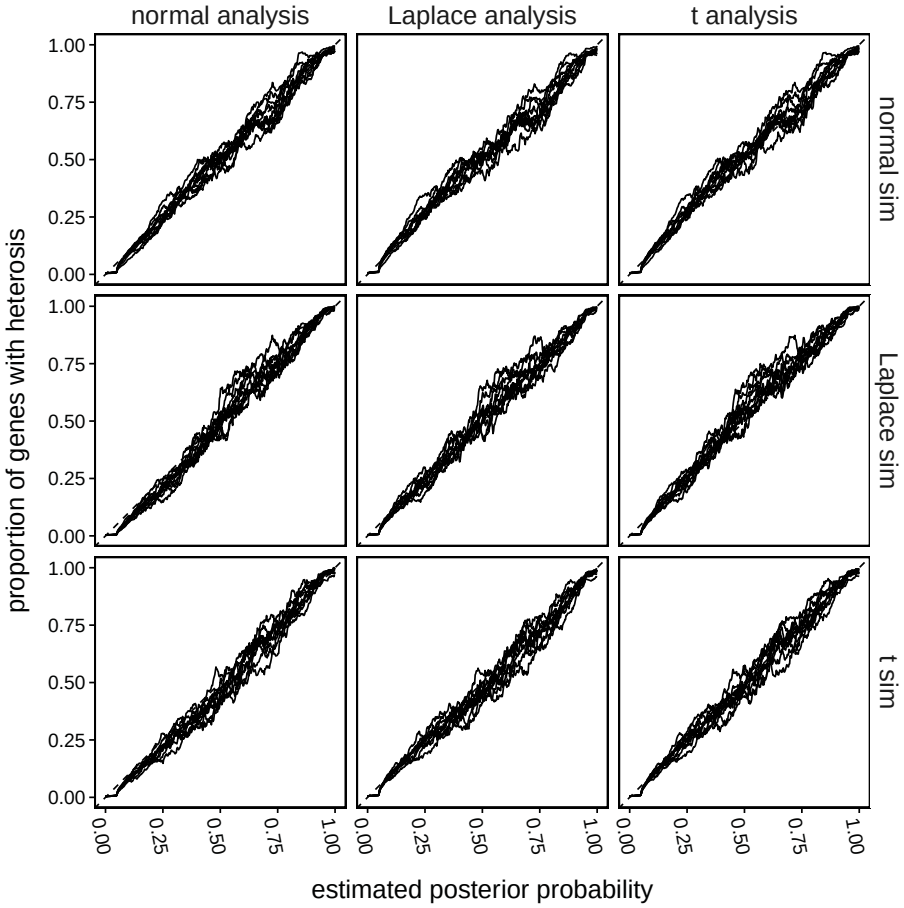

Figure 4.3: For the two-group simulation study in Section 4.3.1, calibration curves corresponding to the posterior probabilities $P(\beta_{g2} > c|y)$ (with c around 0.29) described in Section 4.2.2. The column labels indicate the models used in the analyses, and the row labels indicate the simulation scenario. Each black curve corresponds to the analysis of a simulated dataset.

4.3.2 Plant breeding simulations from Chapter 3

Now, we revisit the simulation studies by Chapter 3, where the main issues of estimation and calibration were originally spotted. By applying our alternate hierarchical distributions

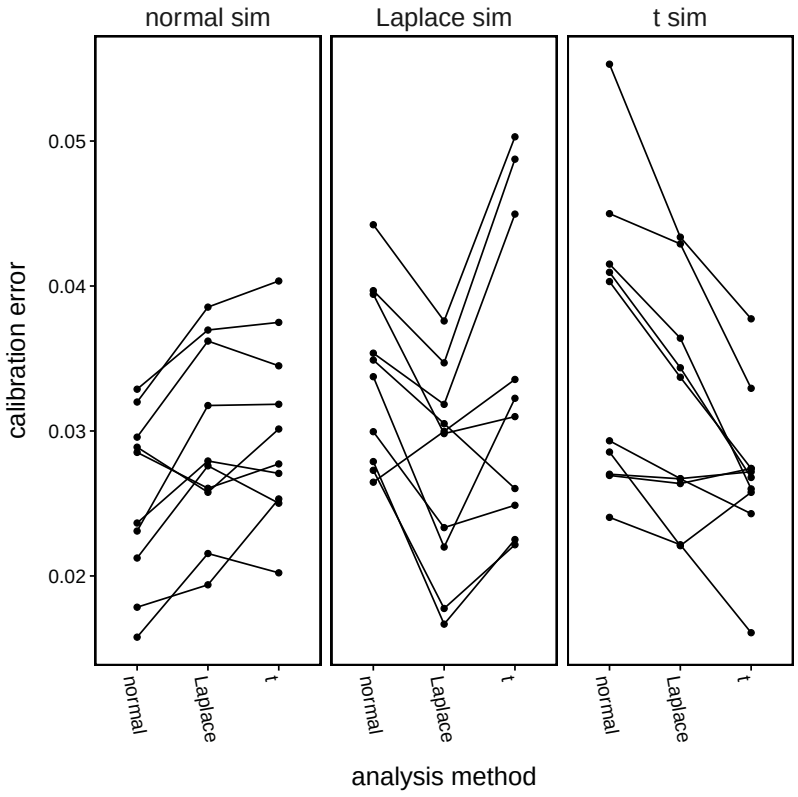

Figure 4.4: For the two-group simulation study in Section 4.3.1, calibration errors corresponding to the posterior probabilities $P(\beta_{g2} > c|y)$ (with c around 0.29) described in Section 4.2.2. The column labels indicate the simulation scenario. Each dot corresponds an analysis of a simulated dataset.

in new analyses of the original simulated datasets, we determine if these core problems are sufficiently addressed.

Simulation Study 1 from Chapter 3 generated 10 datasets with $N = 16$ replicates each, and Simulation Study 2 generated data using multiple data-generating mechanisms, with one dataset for each of $N = 16$ and $N = 32$ for each mechanism. When the model by Chapter 3 was used to generate data, the hierarchical distributions of all the $\beta_{g\ell}$'s were normal distributions. All simulations in Studies 1 and 2 used $G = 30000$ genes and the model matrix from the plant breeding scenario from Chapter 3 motivated by Paschold et al. (2012). For more details on the experiment by Paschold et al. and the general plant breeding scenario, please see Section 3.2 from Chapter 3. The $N \times 5$ model matrix is compactly represented as

$$X = \left(\begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix} \otimes J_{(N/4)\times 1} \qquad J_{(N/4)\times 1} \otimes \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \right) \tag{4.2}$$

where "$\otimes$" denotes the Kronecker product and $J_{m\times n}$ is the m by n matrix with all entries equal to 1. The left-most column is the intercept term, the right-most column is a block effect, and the middle three columns describe quantities of interest: i.e., quantities used to estimate gene-specific posterior probabilities of heterosis with respect to gene expression. For more details about gene expression heterosis and the posterior probabilities thereof, please refer to Chapter 3 Section 3.2, Section 3.4, and Table 3.1.

Chapter 3 provided a fully Bayesian analysis of each dataset with the normal model: that is, the model with normal marginal hierarchical distributions for the $\beta_{g\ell}$ parameters for $\ell = 1, \ldots, L$. For this article, we added analyses with the *Laplace model*, which had Laplace marginal hierarchical distributions for all the $\beta_{g\ell}$'s, and the *t model*, which had Student t_6

marginal hierarchical distributions for all the $\beta_{g\ell}$'s. Convergence results were comparable to those of the normal analyzes from Chapter 3. Nearly all computed Gelman factors fall below 1.1, and for any analysis, at most 11 $\beta_{g\ell}$'s and γ_g's had Gelman factors above 1.1. The maximum Gelman factor was around 7.28 (corresponding to $\beta_{26530,4}$ and the t analysis on a dataset from Simulation Study 1). In addition, the minimum hyperparameter effective sample size across all analyses was around 628 (corresponding to ν, $N = 16$, the Laplace model, and the Simple scenario from Simulation Study 2). Convergence appeared good enough to proceed with the results except for the small collection of genes with high Gelman factors. The maximum runtime over all analyses was about 5.33 hours, corresponding to the Laplace analysis of the $N = 32$ dataset from the edgeR simulation scenario from Section 3.4.2 of Chapter 3.

To compare the coverage of $\beta_{g\ell}$ parameters in credible intervals across models, Figure 4.5 shows, for one example simulated dataset from Simulation Study 1 of Chapter 3, 95% credible intervals for the $\beta_{g\ell}$ parameters that do not cover the true parameter values. Results appear mostly similar across models, with some of the credible intervals for some extreme $\beta_{g\ell}$'s slightly closer to the truth for the Laplace and t analyses than for the normal analyses. Figure 4.6 shows coverage information for all datasets in Simulation Study 1, displaying the local (kernel-smoothed) coverage rate of the 95% credible intervals versus true parameter values. Relative to the normal model, the Laplace and t models apparently relaxed the heavy shrinkage of extreme $\beta_{g\ell}$'s for $\ell = 2$ and 3. This effect is weaker for $\ell = 1$ and 5 and not completely discernible for $\ell = 4$.

To address the robustness of parameter estimation, we turn to Chapter 3 Simulation Study 2, which simulated datasets using multiple different data-generating mechanisms. Figure 4.7 shows, for the datasets from Simulation Study 2 in Section 3.4.2 of Chapter 3, the observed overall rates at which estimated 95% credible intervals covered parameters $\beta_{g\ell}$. Overall, coverage was close to the nominal 95% for the Model and Simple simulation scenarios, with coverage being slightly higher for the β_{g4}'s in the Simple scenario. As with

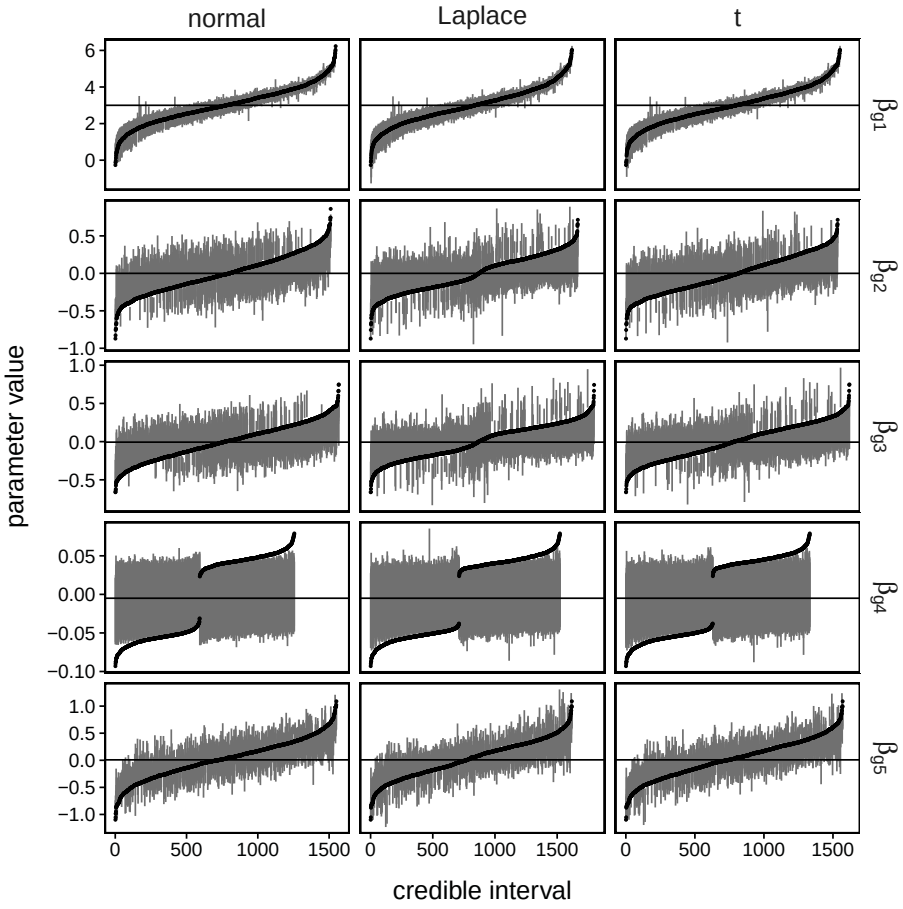

Figure 4.5: Estimated 95% credible intervals for the $\beta_{g\ell}$ parameters for Simulation Study 1 from Chapter 3 Section 3.4.1, with the addition of the Laplace and t analyses described in Sections 4.2.1 and 4.3.2. Only the credible intervals that do not cover the true parameter values are shown. The results here correspond to a single dataset simulated from the normal model with $N = 16$. The models for the analyses are indicated in the column labels, and the $\beta_{g\ell}$ parameters are indicated in the row labels. Each panel shows the estimated credible intervals as vertical gray lines, one for each gene. The black dots correspond to the true parameter values used to generate count data. The horizontal solid black lines indicate the true values of the respective hierarchical means θ_ℓ used in data generation.

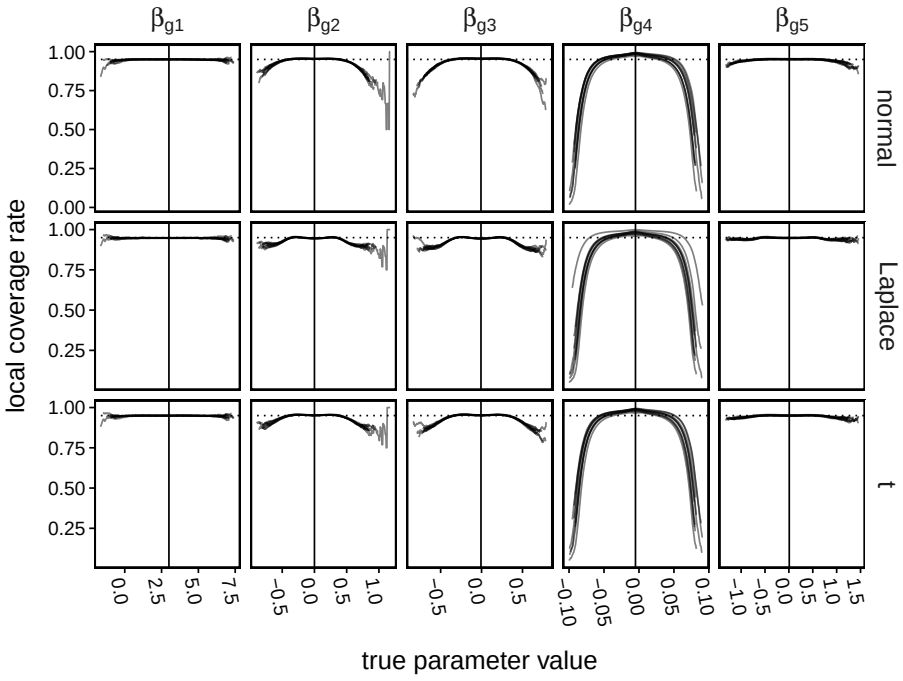

Figure 4.6: Posterior inference for the $\beta_{g\ell}$ parameters for Simulation Study 1 from Chapter 3 Section 3.4.1, with the addition of the Laplace and t analyses described in Sections 4.2.1 and 4.3.2. The row labels indicate the models for the analysis, and the column labels indicate the $\beta_{g\ell}$ parameters. Within each panel, each semitransparent black curve shows the local (kernel-smoothed) proportion of gene-specific 95% credible intervals that cover the true parameter values, plotted against the true parameter values used in data generation. There is one curve for each of the 10 simulated datasets. The horizontal dotted black line near the top of each panel indicates 0.95, the desired coverage rate. The vertical solid black lines indicate the true values of the respective hierarchical means θ_ℓ used in data generation.

the original study by Chapter 3, coverage was still uniformly poor in the edgeR scenario. In all scenarios, coverage rates barely varying across the different analyses. Relative to the normal model, we do not observe any practical impact of the Laplace or t models on the robustness of parameter estimation.

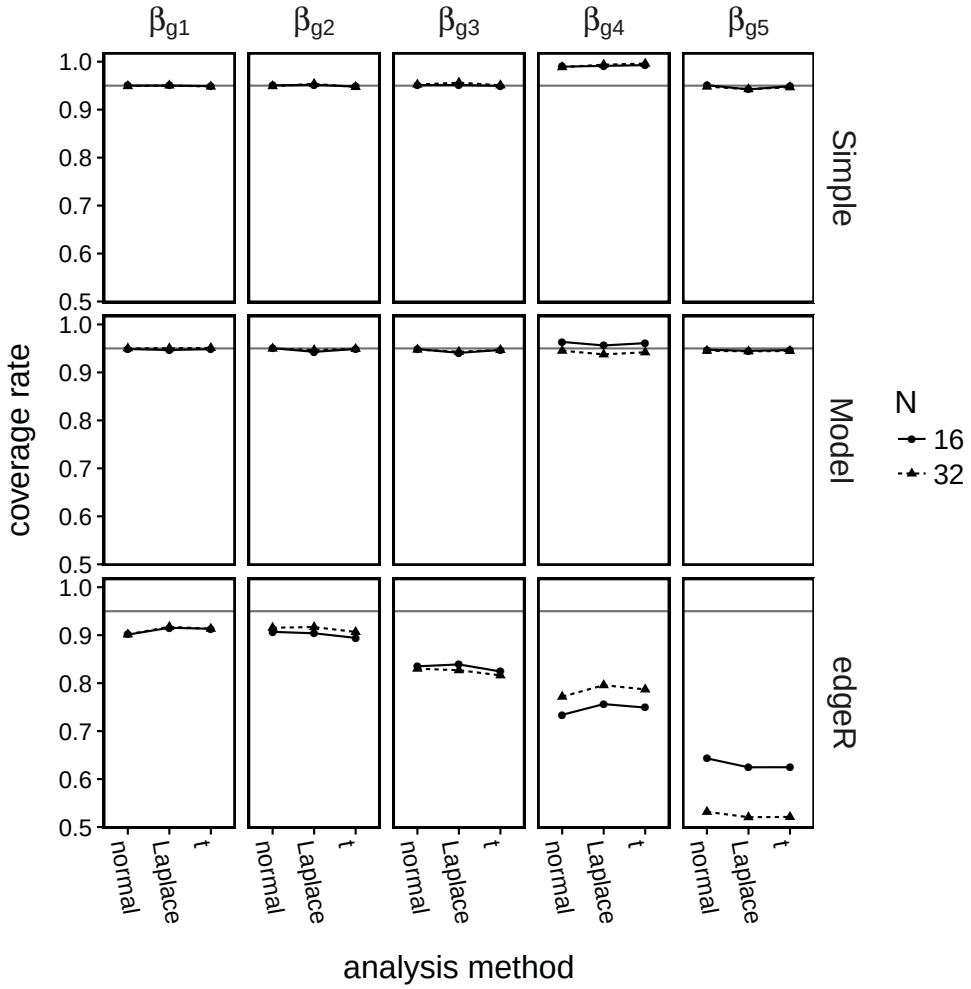

Figure 4.7: For the datasets from Simulation Study 2 in Section 3.4.2 of Chapter 3, observed overall rates at which estimated 95% credible intervals cover parameters $\beta_{g\ell}$. The column labels indicate the $\beta_{g\ell}$ parameters, and the row labels indicate the simulation scenarios. The gray horizontal lines indicate 0.95, the nominal coverage rate.

To evaluate the robustness of posterior probability calibration, we show calibration curves for each analysis of the datasets from Chapter 3 Simulation Study 2. Only the datasets with $N = 16$ are shown, and the results for $N = 32$ in Figure S10 were similar. In the Model scenario, where data were generated from the model with normal hierarchical distributions,

calibration appears to have been satisfactory, though posterior probabilities were underestimated slightly when the Laplace model was used in the analysis. Calibration worsened for the edgeR scenario, with some probabilities systematically overestimated and others systematically underestimated. Calibration was worst in the Simple scenario, where each type of probability was severely overestimated. Overall, calibration was roughly the same as in Chapter 3, and we see no improvements due to the Laplace or t models.

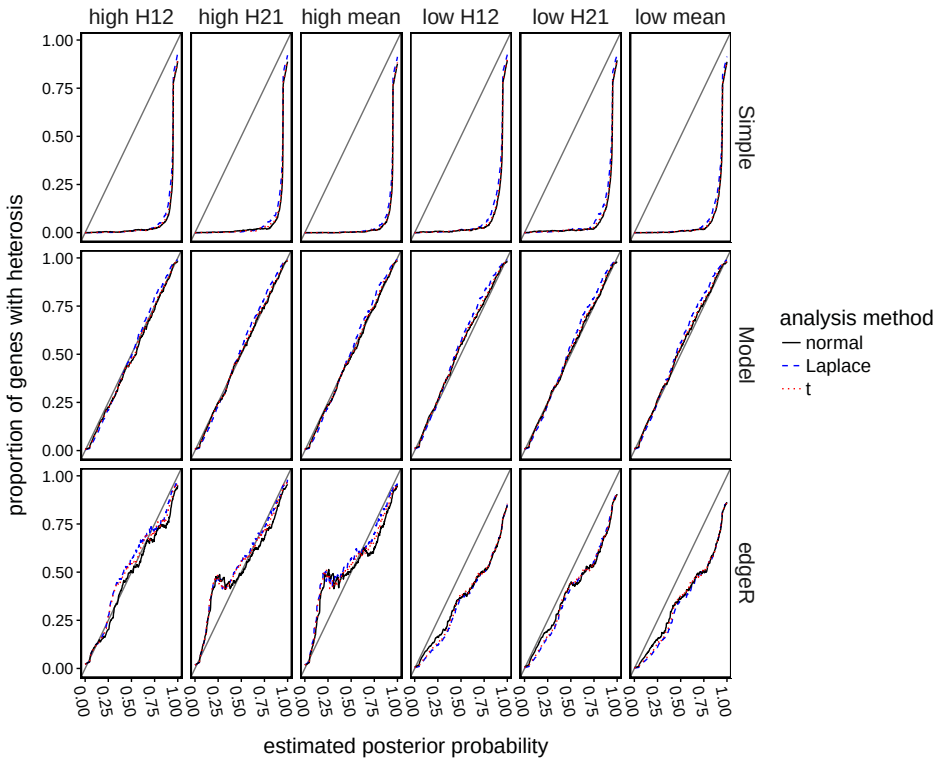

Figure 4.8: For datasets with $N = 16$ in Simulation Study 2 from Chapter 3 Section 3.4.2, calibration curves for heterosis gene detection. The row labels indicate the data-generating mechanisms explained by Chapter 3 Section 3.4.2. The types of posterior probabilities calculated, as explained by Section 3.2, Section 3.1, and Table 3.3.2 of Chapter 3 are indicated in the column labels.

4.4 Discussion

We addressed two major weaknesses of the fully Bayesian RNA-seq data analysis method from Chapter 3. First, the gene-specific parameters $\beta_{g\ell}$ were sometimes poorly estimated.

Specifically, coverage in credible intervals dropped below the nominal level for extreme-valued $\beta_{g\ell}$'s, and coverage was poor overall when the data-generating mechanism disagreed with the model used in the analysis. Second, the calibration of estimated posterior probabilities was poor when the data violated the model's assumptions. We attempted to rectify these issues by altering the model, replacing the previously normal marginal hierarchical distributions for the $\beta_{g\ell}$'s with Laplace and Student t distributions. To evaluate the models relative to one another, we used simulation studies. We found that with the Laplace and t models, the estimation of extreme $\beta_{g\ell}$'s improved slightly, but the robustness of parameter estimation and the calibration of posterior probabilities did not. Overall, the results were extremely similar among the three models.

It may be that the results were so similar across models because the models themselves were similar. Interchanging normal, Laplace, and Student t distributions may only be a slight adjustment with small practical consequences, and more extensive changes may be required to increase flexibility and robustness. One possible avenue for future work, also suggested by Chapter 3, is to model the $\beta_{g\ell}$ parameters with a semi-parametric approach, such as a Dirichlet process mixture (Liu et al., 2015; Muller and Mitra, 2013). Such a technique would allow the model to learn the distribution of the gene-specific parameters, and in addition, relax the current assumptions of conditional independence across genes.

CHAPTER 5. CONCLUSION

This work begins with a computationally efficient general strategy for fully Bayesian hierarchical modeling. The approach, which parallelizes Monte Carlo algorithms with a Gibbs sampling structure, is solidly tractable and could have a wide variety of applications. The application we explore is RNA-seq data analysis, an area where fully Bayesian techniques are still uncommon due to the computational expense but potentially helpful for addressing the problems of low sample size per gene and multiple-testing. The advantage is the borrowing of information across genes to improve detection, which shrinks gene-specific parameter estimates towards shared measures of center. The approach is successful according to many metrics of estimation, inference, and gene detection, with the best-case-scenario empirical Bayes versions equally effective. In an attempt to improve the model, we experiment with Laplace and Student t hierarchical distributions for gene-specific model coefficient parameters. As suggested in Chapters 3 and 4, other possibilities for improvement include a semi-parametric approach using a Dirichlet process mixture (Liu et al., 2015; Muller and Mitra, 2013), which may increase flexibility and robustness by learning the distributions of gene-specific parameters and relaxing conditional independence assumptions across genes.

In short, our RNA-seq model is currently undergoing the typical iterative process of fitting, testing, diagnosing, changing, refitting, etc. This workflow benefits from the computational strategy in Chapter 2, which decreases the time required to test the implementation of a given model. However, the process of creating an implementation in the first place is inefficient because currently, to use the strategy in Chapter 2, the Monte Carlo routine needs to be implemented from scratch, which could take several weeks or months.

For less cumbersome applications that do not require massively parallel computing, model development is already fast. General purpose Bayesian software such WinBUGS (Lunn et al., 2000), OpenBUGS (Lunn et al., 2009), JAGS (Plummer et al., 2003), Stan (Carpenter et al.,

2016), and NIMBLE de Valpine et al. (2016) have lowered the development time by allowing scientists to focus on model construction rather than computational details. The user need only write a few lines of code to specify the model, and an MCMC routine is already ready to run. If there were a similar black-box program for the strategy in Chapter 2, implementing both the parallelism and the memory transfer bottleneck workarounds, the development of large hierarchical models would be faster both conceptually and computationally. Existing platforms with modular structures, such as JAGS, may already be able to accommodate.

BIBLIOGRAPHY

Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Genome Biol*, 11(10):R106.

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2015). *Hierarchical Modeling and Analysis for Spatial Data*, chapter 9.4.1: Regression in the Gaussian case. CRC Press, 2nd edition.

Bassene, J. B., Froelicher, Y., Dubois, C., Ferrer, R. M., Navarro, L., Ollitrault, P., and Ancillo, G. (2010). Non- additive gene regulation in a citrus allotetraploid somatic hybrid between C. reticulata Blanco and C. limon (L.) Burm. *Heredity*, 105(3):299–308.

Beam, A. L., Ghosh, S. K., and Doyle, J. (in press 2015). Fast Hamiltonian Monte Carlo using GPU computing. *Journal of Computational and Graphical Statistics*.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A. (2016). Stan: a probabilistic programming language. *Journal of Statistical Software*. in press.

Carvahlo, C., Polson, N., and Scott, J. (2009). Handling Sparsity via the Horseshoe. *Proceedings of the 12th International Conference on Artifical Intelligence and Statistics*.

Coors, J. and Pandey, S. (1999). The genetics and exploitation of heterosis in crops. *Crop Science Society of America, Inc*.

Cruz, M. G., Peters, G. W., and Shevchenko, P. V. (2015). *Fundamental Aspects of Operational Risk and Insurance Analytics: A Handbook of Operational Risk*, chapter 7.6.2: Generic univariate auxiliary variable Gibbs sampler: slice sampler. Wiley.

Darwin, C. (1876). *The effects of cross and self fertilisation in the vegetable kingdom*. John Murray.

Datta, S. and Nettleton, D. (2014). *Statistical Analysis of Next Generation Sequencing Data*. Springer.

de Valpine, P., Paciorek, C., Turek, D., Anderson-Bergman, C., and Lang, D. T. (2016). nimble: Flexible bugs-compatible system for hierarchical statistical modeling and algorithm development. R package version 0.5. URL last visited April 14, 2016.

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1(3):515–533.

Gelman, A., Carlin, J. b., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. CRC Press, 3rd edition.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472.

Gramacy, R. B., Niemi, J., and Weiss, R. M. (2014). Massively parallel approximate gaussian process regression. *SIAM/ASA Journal of Uncertainty Quantification*, 2(1):564–584.

Gruber, L., West, M., et al. (2016). GPU-accelerated Bayesian learning and forecasting in simultaneous graphical dynamic linear models. *Bayesian Analysis*, 11(1):125–149.

Hallauer, A. and Miranda, F. (1981). Quantitative genetics in maize breeding. *Iowa St. Univ. Press, Ames, IA*.

Hallauer, A. R., Carena, M. J., and Miranda Filho, J. (2010). *Quantitative genetics in maize breeding*, volume 6. Springer.

Hardcastle, T. J. (2012). *baySeq: Empirical Bayesian analysis of patterns of differential expression in count data*. R package version 2.0.50.

Hardcastle, T. J. and Kelly, K. A. (2010). baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. *BMC Bioinformatics*, 11(1):422.

Hoecker, N., Keller, B., Piepho, H., and Hochholdinger F. (2006). Manifestation of heterosis during early maize (Zea mays L.) root development. *Theoretical and Applied Genetics*, 112(3):421–429.

Jacob, P., Robert, C. P., and Smith, M. H. (2011). Using parallel computation to improve independent Metropolis–Hastings based estimation. *Journal of Computational and Graphical Statistics*, 20(3):616–635.

Ji, T., Liu, P., and Nettleton, D. (2014). Estimation and testing of gene expression heterosis. *Journal of Agricultural, Biological, and Environmental Statistics*, 19(3):319–337.

Johnson, A. A., Jones, G. L., Neath, R. C., et al. (2013). Component-wise Markov chain Monte Carlo: Uniform and geometric ergodicity under mixing and composition. *Statistical Science*, 28(3):360–375.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Krieger, U., Lippman, Z., and Zamir, D. (2010). The flowering gene single flower truss drives heterosis for yield in tomato. *Nature Genetics*, 42(5):459–463.

Landau, W. M. and Liu, P. (2013). Dispersion estimation and its effect on test performance in RNA-seq data analysis: a simulation-based comparison of methods. *PLOS ONE*, 8(12).

Lee, A., Yau, C., Giles, M. B., Doucet, A., and Holmes, C. C. (2010). On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789. URL last visited March 25, 2016.

Ling, R. F. (1974). Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association*, 69(348):859–866.

Lippman, Z. and Zamir, D. (2007). Heterosis: revisiting the magic. *Trends in Genetics*, 23(2):60–66.

Lithio, A. and Nettleton, D. (2015). Hierarchical modeling and differential expression analysis for RNA-seq experiments with inbred and hybrid genotypes. *Journal of Agricultural, Biological, and Environmental Statistics*, 20(4):598–613.

Liu, F., Wang, C., and Liu, P. (2015). A semi-parametric bayesian approach for differential expression analysis of RNA-seq data. *Journal of Agricultural, Biological, and Environmental Statistics*, 20(4).

Lund, S. P., Nettleton, D., McCarthy, D. J., and Smyth, G. K. (2012). Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates. *Statistical Applications in Genetics and Molecular Biology*, 11(5).

Lunn, D., Spiegelhalter, D., Thomas, A., and Best, N. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in medicine*, 28(25):3049–3067.

Lunn, D. J., Thomas, A., Best, N., and Spiegelhalter, D. (2000). WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and computing*, 10(4):325–337.

McCarthy, D., Chen, Y., and Smyth, G. (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research*.

McCullagh, P. and Nelder, J. A. (1989). *Generalized linear models*, volume 37. CRC press.

Mortazavi, A., Williams, B., McCue, K., Schaeffer, L., and Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature methods*, 5(7):621–628.

Muller, P. and Mitra, R. (2013). Bayesian nonparametric inference - why and how. *Bayesian Analysis*, 8(2):269–302.

Murray, I. and Adams, R. P. (2014). Parallel MCMC with generalized elliptical slice sampling. *Journal of Machine Learning Research*, 15(1):2087–2112.

Neal, R. M. (2003). Slice Sampling. *The Annals of Statistics*, 31(3):705–767.

Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable parallel programming with CUDA. *ACM Queue*, 6(2).

Niemi, J., Mittman, E., Landau, W., and Nettleton, D. (2015). Empirical bayes analysis of RNA-seq data for detection of gene expression heterosis. *Journal of Agricultural, Biological, and Environmental Statistics*, 20(4):614–628.

NVIDIA (2015). Thrust. http://docs.nvidia.com/cuda/thrust/. URL last visited March 25, 2016.

Oshlack, A., Robinson, M. D., and Young, M. D. (2010). From RNA-seq reads to differential

expression results. *Genome Biology*, 11(220).

Park, T. and Casella, G. (2008). The Bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686.

Paschold, A., Jia, Y., Marcon, C., Lund, S., Larson, N. B., Yeh, C.-T., Ossowski, S., Lanz, C., Nettleton, D., Schnable, P. S., et al. (2012). Complementation contributes to transcriptome complexity in maize (*Zea mays L.*) hybrids relative to their inbred parents. *Genome research*, 22(12):2445–2454.

Plummer, M. et al. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd international workshop on distributed statistical computing*, volume 124, page 125. Technische Universit at Wien Wien, Austria.

Ramskold, D., Luo, S., Wang, Y.-C., Li, R., Deng, Q., Faridani, O. R., Daniels, G. A., Khrebtukova, I., Loring, J. F., Laurent, L. C., Schroth, G. P., and Sandberg, R. (2012). Full-length mRNA-seq from single-cell levels of RNA and individual circulating tumor cells. *Nat Biotech*, 30(8):777–782.

Riday, H. and Brummer, E. (2002). Heterosis of agronomic traits in alfalfa. *Crop science*, 42(4):1081–1087.

Robinson, M., Oshlack, A., et al. (2010a). A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*, 11(3):R25.

Robinson, M. D., McCarthy, D. J., and Smyth, G. K. (2010b). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26:139–40.

Schnable, P., Ware, D., Fulton, R., Stein, J., Wei, F., Pasternak, S., Liang, C., Zhang, J., Fulton, L., and Graves, T.A., e. a. (2009). The B73 maize genome: complexity, diversity, and dynamics. *Science*, 326(5956):1112–1115.

SHARCNET (2012). CUDA tips and tricks. https://www.sharcnet.ca/. URL last visited March 25, 2016.

Si, Y. and Liu, P. (2013). An optimal test with maximum average power while controlling fdr with application to rna-seq data. *Biometrics*, 69(3).

Springer, N. and Stupar, R. (2007). Allelic variation and heterosis in maize: How do two halves make more than a whole? *Genome research*, 17(3):264–275.

Stan Development Team (2014). RStan: the R interface to Stan, version 2.5.0.

Suchard, M., Wang, Q., Chan, C., Frelinger, J., Cron, A., and West, M. (2010). Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics*, 19(2):419–438.

Suchard, M. A. and Rambaut, A. (2009). Many-core algorithms for statistical phylogenetics.

Bioinformatics, 25(11):1370–1376.

Swanson-Wagner, R., Jia, Y., DeCook, R., Borsuk, L., Nettleton, D., and Schnable, P. (2006). All possible modes of gene action are observed in a global comparison of gene expression in a maize F1 hybrid and its inbred parents. *Proceedings of the National Academy of Sciences*, 103(18):6805–6810.

Tibbits, MM and Haran, M and Liechty, JC (2011). Parallel Multivariate Slice Sampling. *Statistics and Computing*, 21(3):415–430.

Wang, J., Tian, L., Lee, H., Wei, N., Jiang, H., Watson, B., Madlung, A., Osborn, T. C., Doerge, R. W., Comai, L., and Chen, Z. J. (2006). Genomewide nonadditive gene regulation in arabidopsis allotetraploids. *Genetics*, 172(1):507–517.

Wang, L., Li, P., and Brutnell, T. P. (2010). Exploring plant transcriptomes using ultra high-throughput sequencing. *Briefings in Functional Genomics*, 9(2):118–128.

Welford, B. P. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.

White, G. and Porter, M. D. (2014). GPU accelerated MCMC for modeling terrorist activity. *Computational Statistics & Data Analysis*, 71:643–651.

Winz, R. and Baldwin, I. (2001). Molecular interactions between the specialist herbivore Manduca sexta (Lepidoptera, Sphingidae) and its natural host Nicotiana attenuata. iv. insect-induced ethylene reduces jasmonate-induced nicotine accumulation by regulating putrescine N-methyltransferase transcripts. *Plant Physiology*, 125(4):2189–2202.

Wohlfarth, G. (1993). Heterosis for growth rate in common carp. *Aquaculture*, 113(1-2):31–46.

Wu, H., Wang, C., and Wu, Z. (2012). A new shrinkage estimator for dispersion improves differential expression detection in RNA-seq data. *Biostatistics*, 1(1):1–24.

Yu, S., Li, J., Xu, C., Tan, Y., Gao, Y., Li, X., Zhang, Q., and Maroof, M. (1997). Importance of epistasis as the genetic basis of heterosis in an elite rice hybrid. *Proceedings of the National Academy of Sciences*, 94(17):9226.

APPENDIX A. STEPPING-OUT SLICE SAMPLER

In the MCMC in Section 2.4.2, we repeatedly apply the univariate stepping-out slice sampler given by Neal (2003). The goal of slice sampling is to sample θ from an arbitrary univariate density proportional to some function $f(\theta)$. To do this, Neal's method samples from $g(\theta, u)$, the bivariate uniform density on the region under $f(\theta)$ (i.e., $\{(\theta, u) : -\infty < \theta < \infty, 0 < u < f(\theta)\}$). The marginal density of θ under $g(\theta, u)$ is $f(\theta)/\int_{-\infty}^{\infty} f(\theta)d\theta$, so the samples of θ come from the correct target.

To sample from $g(\theta, u)$, Neal's method uses a technique similar to a 2-step Gibbs sampler. Here, suppose the current state is $(\theta, u) = (\theta^{(m)}, u_0)$. The first step of this two-step Gibbs sampler is to draw a new value $u \sim \text{Uniform}(0, f(\theta^{(m)}))$, the full conditional distribution of u given $\theta = \theta^{(m)}$. The next step is to draw a new value $\theta^{(m+1)}$ of θ from the uniform distribution on $S = \{\theta : u < f(\theta)\}$, the conditional distribution of θ given u. Unfortunately, precisely determining the "slice" S is inefficient and not expedient in practice. The following explicit steps comprise a single stepping-out slice sampler iteration that moves from the current state $\theta = \theta^{(m)}$ to the next state $\theta = \theta^{(m+1)}$.

The tuning procedure in step 7 sets w to be a weighted average of the absolute differences between successive values of θ, giving precedence to later iterations. That way, w is calibrated according to the width of the "slice" $S = \{\theta : u < f(\theta)\}$. The popular black-box Gibbs sampler software, JAGS, uses this tuning method for its own slice sampler in version 4.0.1 (Plummer et al., 2003).

Algorithm 2 Univariate stepping-out slice sampler with tuning

Set the initial size of the step w, total number MCMC iterations (M), number of burn-in iterations (M_B), number of initial iterations where w is not tuned ($M_C < M_B$), maximum number of "stepping out" steps ($K \in \mathbb{N}^+$), and $w_{\text{aux}} = 0$. Let $\theta^{(m)}$ be the current value of θ at iteration m of the MCMC chain.

1. Sample $u \sim \text{Uniform}(0, f(\theta^{(m)}))$ distribution.
2. Randomly place an interval (L, R) of width w around $\theta^{(m)}$:
   (a) Sample $v \sim \text{Uniform}(0,\text{w})$.
   (b) Set $L = \theta^{(m)} - v$.
   (c) Set $R = L + w$.
3. Set upper limits on the number of steps to perform in each direction:
   (a) Sample K_L uniformly on $\{0, 1, \ldots, K\}$.
   (b) Set $K_R = K - K_L$.
4. "Step out" the interval $(L,\ R)$ to cover the "slice" $S = \{\theta : u < f(\theta)\}$:
   (a) For $k = 1, \ldots, K_L$, set $L = L - w$ if $u < f(L)$.
   (b) For $k = 1, \ldots, K_R$, set $R = R + w$ if $u < f(R)$.
5. Sample $\theta^* \sim \text{Uniform}(L, R)$ distribution.
6. If $u < f(\theta^*)$, set $\theta^{(m+1)} = \theta^*$. Otherwise, set $R = \theta^*$ if $\theta^* > \theta^{(m)}$ or $L = \theta^*$ if $\theta^* \leq \theta^{(m)}$, then go back to step 5.
7. If $m \leq M_B$, tune w as follows.
   (a) Increment w_{aux} by $m \cdot |\theta^{(m+1)} - \theta^{(m)}|$.
   (b) If $m > M_C$, set $w = w_{\text{aux}}/(0.5m(m+1))$.

APPENDIX B. DERIVATIONS OF DISTRIBUTIONS

The purpose of this section is to derive useful distributions, such as the full conditional distributions, of the parameters in the hierarchical RNA-seq model. The full conditionals are necessary in order to perform the massively-parallelized slice-sampling-within-Gibbs Markov chain Monte Carlo algorithm used to estimate the full joint posterior distribution. Recall that the hierarchical model for RNA-seq count data y_{gn} is summarized by

$$\begin{aligned}
y_{n,g} &\overset{\text{ind}}{\sim} \text{Poisson}\left(\exp\left(h_n + \varepsilon_{gn} + X_n\beta_g\right)\right) \\
\varepsilon_{gn} &\overset{\text{ind}}{\sim} \text{Normal}(0, \gamma_g) \\
\gamma_g &\overset{\text{ind}}{\sim} \text{Inverse-Gamma}\left(\frac{\nu}{2}, \frac{\nu\tau}{2}\right) \\
\nu &\sim \text{Uniform}(0, d) \\
\tau &\sim \text{Gamma}(a, \text{rate} = b) \\
\beta_{g\ell} &\sim \text{Normal}(\theta_\ell, \sigma_\ell^2\xi_{g\ell}) \\
\theta_\ell &\sim \text{Normal}(0, c_\ell^2) \\
\sigma_\ell &\sim \text{Uniform}(0, s_\ell) \\
\xi_{g\ell} &\sim p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell)
\end{aligned}$$

where $\overset{\text{ind}}{\sim}$ indicates conditional independence, $g = 1, \ldots, G$ is the index of a gene, $n = 1, \ldots, N$ is the index of a replicate, and X is the $N \times L$ model matrix, X_n is the n'th row of X, $\beta_g = (\beta_{g1}, \ldots, \beta_{gL})$, and the h_n's are normalization constants determined before parameter estimation. All Greek letters above denote model parameters, and all Roman letters are model constants.

For convenience, let $p(\psi|\ldots)$ be the full conditional distribution of parameter ψ given all the other parameters and the data. The full conditionals of all the parameters except the

$\xi_{g\ell}$'s are summarized below.

$$p(\theta_\ell|\ldots) = \text{Normal}\left(\frac{B}{2A},\ \frac{1}{2A}\right) \quad \left(A = \frac{1}{2}\left(\frac{1}{c_\ell^2} + \frac{1}{\sigma_\ell^2}\sum_{g=1}^{G}\frac{1}{\xi_{g\ell}}\right),\ B = \frac{1}{\sigma_\ell^2}\sum_{g=1}^{G}\frac{\beta_{g\ell}}{\xi_{g\ell}}\right)$$

$$p(\tau \mid \cdots) = \text{Gamma}\left(\text{shape} = a + \frac{G\nu}{2},\ \text{rate} = b + \frac{\nu}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right)$$

$$p(\gamma_g|\ldots) = \text{Inverse-Gamma}\left(\text{shape} = \frac{N+\nu}{2},\ \text{scale} = \frac{1}{2}\left(\nu\tau + \sum_{n=1}^{N}\varepsilon_{gn}^2\right)\right)$$

$$p(\sigma_\ell^2|\ldots) = \text{Inverse-Gamma}\left(\text{shape} = \frac{G-1}{2},\ \text{scale} = \frac{1}{2}\sum_{g=1}^{G}\frac{(\beta_{g\ell}-\theta_\ell)^2}{\xi_{g\ell}}\right)\mathrm{I}(\sigma_\ell^2 < s_\ell^2)$$

$$p(\nu|\ldots) \propto \exp\left(-G\log\Gamma\left(\frac{\nu}{2}\right) + \frac{G\nu}{2}\log\left(\frac{\nu\tau}{2}\right) - \frac{\nu}{2}\sum_{g=1}^{G}\left[\log\gamma_g + \frac{\tau}{\gamma_g}\right]\right)\mathrm{I}(0<\nu<d)$$

$$p(\varepsilon_{gn} \mid \cdots) \propto \exp\left(y_{gn}\varepsilon_{gn} - \frac{\varepsilon_{gn}^2}{2\gamma_g} - \exp\left(\varepsilon_{gn}\right)\exp\left(h_n + \sum_{i=1}^{L}X_{ni}\beta_{gi}\right)\right)$$

$$p(\beta_{g\ell} \mid \cdots) \propto \exp\left(\beta_{g\ell}\sum_{n=1}^{N}y_{gn}X_{n\ell} - \frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2\xi_{g\ell}}\right.$$
$$\left. - \sum_{x\in S_\ell}\exp\left(x\beta_{g\ell}\right)\sum_{n=1}^{N}\mathrm{I}(X_{n\ell}=x)\exp\left(h_n + \varepsilon_{gn} + \sum_{i\neq l}X_{ni}\beta_{gi}\right)\right)$$

Table B.1 summarizes the full conditional distributions of the $\xi_{g\ell}$ parameters for each choice of prior, along with the resulting marginal hierarchical distributions $p(\beta_\ell|\theta_\ell,\sigma_\ell^2)$ of $\beta_{g\ell}$. The following sections derive the individual full conditionals and marginal hierarchical distributions.

Table B.1: The full conditional distribution of each $\xi_{g\ell}$ parameter up to a proportionality constant (middle column) for each choice of prior (left column), along with the resulting marginal hierarchical distribution of $\beta_{g\ell}$ (right column). Here, $\mathrm{I}(\cdot)$ is the indicator function.

$\xi_{g\ell}$ prior	$\xi_{g\ell}$ full conditional	$\beta_{g\ell}$ marginal
$\mathrm{I}(\xi_{g\ell}=1)$	$\mathrm{I}(\xi_{g\ell}=1)$	$\text{Normal}(\theta_\ell,\ \sigma_\ell^2)$
$\text{Exp}(\text{rate}=k_\ell)$	$\exp\left(-\frac{1}{2}\log\xi_{g\ell} - \frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2}\frac{1}{\xi_{g\ell}} - k_\ell\xi_{g\ell}\right)$	$\text{Laplace}\left(\theta_\ell, \sqrt{\frac{\sigma_\ell^2}{2k_\ell}}\right)$
$\text{IG}(q_\ell, r_\ell)$	$\exp\left(-\left(q_\ell+\frac{3}{2}\right)\log\xi_{g\ell} - \left(\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2} + r_\ell\right)\frac{1}{\xi_{g\ell}}\right)$	$t_{2q_\ell}\left(\theta_\ell,\ \text{scale} = \sqrt{\frac{\sigma_\ell^2 r_\ell}{q_\ell}}\right)$

B.1 θ_ℓ, normal

Let $\text{Normal}(x|\mu, \sigma^2)$ be the normal density for random variable x with mean μ and variance σ^2.

$$\begin{aligned}
p(\theta_\ell|\ldots) &\propto \left[\prod_{g=1}^{G} \text{Normal}(\beta_{g\ell}|\theta_\ell,\ \sigma_\ell^2\xi_{g\ell})\right] \text{Normal}(\theta_\ell|0,\ c_\ell^2) \\
&= \text{Normal}(\theta_\ell|0,\ c_\ell^2) \prod_{g=1}^{G} \text{Normal}(\theta_\ell|\beta_{g\ell}, \sigma_\ell^2\xi_{g\ell}) \\
&\propto \exp\left(-\frac{\theta_\ell^2}{2c_\ell^2} - \sum_{g=1}^{G} \frac{(\theta_\ell - \beta_{g\ell})^2}{2\sigma_\ell^2\xi_{g\ell}}\right) \\
&= \exp\left(-\frac{\theta_\ell^2}{2c_\ell^2} - \sum_{g=1}^{G} \frac{\theta_\ell^2 - \theta_\ell \cdot 2\beta_{g\ell} + \beta_{g\ell}^2}{2\sigma_\ell^2\xi_{g\ell}}\right) \\
&\propto \exp\left(-\frac{\theta_\ell^2}{2c_\ell^2} - \sum_{g=1}^{G} \frac{\theta_\ell^2 - \theta_\ell \cdot 2\beta_{g\ell}}{2\sigma_\ell^2\xi_{g\ell}}\right) \\
&= \exp\left(-\theta_\ell^2 \frac{1}{2}\left(\frac{1}{c_\ell^2} + \sum_{g=1}^{G} \frac{1}{\sigma_\ell^2\xi_{g\ell}}\right) + \theta_\ell \sum_{g=1}^{G} \frac{2\beta_{g\ell}}{2\sigma_\ell^2\xi_{g\ell}}\right) \\
&= \exp\left(-\theta_\ell^2 \frac{1}{2}\left(\frac{1}{c_\ell^2} + \frac{1}{\sigma_\ell^2}\sum_{g=1}^{G} \frac{1}{\xi_{g\ell}}\right) + \theta_\ell \frac{1}{\sigma_\ell^2} \sum_{g=1}^{G} \frac{\beta_{g\ell}}{\xi_{g\ell}}\right)
\end{aligned}$$

Now, let

$$\begin{aligned}
A &= \frac{1}{2}\left(\frac{1}{c_\ell^2} + \frac{1}{\sigma_\ell^2}\sum_{g=1}^{G} \frac{1}{\xi_{g\ell}}\right) \\
B &= \frac{1}{\sigma_\ell^2} \sum_{g=1}^{G} \frac{\beta_{g\ell}}{\xi_{g\ell}}
\end{aligned}$$

Then,

$$\begin{aligned}
p(\theta_\ell|\ldots) &\propto \exp\left(-\theta_\ell^2 A + \theta_\ell B\right) \\
&\propto \exp\left(-A\left(\theta_\ell - \frac{B}{2A}\right)^2\right) \\
&= \exp\left(-\frac{1}{2(2A)^{-1}}\left(\theta_\ell - \frac{B}{2A}\right)^2\right)
\end{aligned}$$

which is proportional to a standard normal density. In summary,

$$p(\theta_\ell|\ldots) = \text{Normal}\left(\frac{B}{2A},\ \frac{1}{2A}\right) \quad \left(A = \frac{1}{2}\left(\frac{1}{c_\ell^2} + \frac{1}{\sigma_\ell^2}\sum_{g=1}^{G}\frac{1}{\xi_{g\ell}}\right),\ B = \frac{1}{\sigma_\ell^2}\sum_{g=1}^{G}\frac{\beta_{g\ell}}{\xi_{g\ell}}\right)$$

B.2 τ, gamma

Let Inverse-Gamma$(x|a,b)$ be the inverse-gamma density of random variable x with shape a and scale b. Define Gamma$(x|a,b)$ similarly for rate b.

$$\begin{aligned}
p(\tau \mid \cdots) &= \left[\prod_{g=1}^{G}\text{Inverse-Gamma}\left(\gamma_g \mid \frac{\nu}{2}, = \frac{\nu\tau}{2}\right)\right] \cdot \text{Gamma}(\tau \mid \text{shape} = a, \text{rate} = b) \\
&\propto \left[\Gamma\left(\frac{\nu}{2}\right)^{-G}\left(\frac{\nu\tau}{2}\right)^{G\nu/2}\left(\prod_{g=1}^{G}\gamma_g\right)^{-(\nu/2+1)}\exp\left(-\frac{\nu\tau}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right)\right] \cdot (\tau)^{a-1}\exp\left(-b\tau\right) \\
&\propto \left[(\tau)^{G\nu/2}\exp\left(-\tau\cdot\frac{\nu}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right)\right] \cdot (\tau)^{a-1}\exp\left(-b\tau\right) \\
&= (\tau)^{G\nu/2+a-1}\exp\left(-\tau\left(b + \frac{\nu}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right)\right)
\end{aligned}$$

Hence,

$$p(\tau \mid \cdots) = \text{Gamma}\left(\text{shape} = a + \frac{G\nu}{2},\ \text{rate} = b + \frac{\nu}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right)$$

B.3 γ_g, inverse gamma

$$\begin{aligned}
p(\gamma_g \mid \cdots) &= \left[\prod_{n=1}^{N}\text{Normal}(\varepsilon_{gn} \mid \gamma_g)\right] \cdot \text{Inverse-Gamma}\left(\gamma_g \mid \text{shape} = \frac{\nu}{2}, \text{scale} = \frac{\nu\tau}{2}\right) \\
&\propto \left[\prod_{n=1}^{N}(\gamma_g)^{-1/2}\exp\left(-\frac{1}{\gamma_g}\frac{\varepsilon_{gn}^2}{2}\right)\right] \cdot (\gamma_g)^{-(\nu/2+1)}\exp\left(-\frac{1}{\gamma_g}\frac{\nu\tau}{2}\right) \\
&= \left[(\gamma_g)^{-N/2}\exp\left(-\frac{1}{\gamma_g}\frac{1}{2}\sum_{n=1}^{N}\varepsilon_{gn}^2\right)\right] \cdot (\gamma_g)^{-(\nu/2+1)}\exp\left(-\frac{1}{\gamma_g}\frac{\nu\tau}{2}\right) \\
&= (\gamma_g)^{-((N+\nu)/2+1)}\exp\left(-\frac{1}{\gamma_g}\frac{1}{2}\left(\nu\tau + \sum_{n=1}^{N}\varepsilon_{gn}^2\right)\right)
\end{aligned}$$

which is the kernel of an inverse gamma distribution. Hence:

$$p(\gamma_g|\ldots) = \text{Inverse-Gamma}\left(\text{shape} = \frac{N+\nu}{2},\ \text{scale} = \frac{1}{2}\left(\nu\tau + \sum_{n=1}^{N}\varepsilon_{gn}^2\right)\right)$$

B.4 σ_ℓ^2, truncated inverse-gamma

$$\begin{aligned}
p(\sigma_\ell^2|\ldots) &\propto p(\sigma_\ell^2|s_\ell)\prod_{g=1}^{G}\text{Normal}(\beta_{g\ell}|\theta_\ell,\ \sigma_\ell^2\xi_{g\ell}) \\
&\propto p(\sigma_\ell^2|s_\ell)\prod_{g=1}^{G}(\sigma_\ell^2)^{-1/2}\exp\left(-\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2\xi_{g\ell}}\right) \\
&= p(\sigma_\ell^2|s_\ell)(\sigma_\ell^2)^{-G/2}\exp\left(-\frac{1}{\sigma_\ell^2}\frac{1}{2}\sum_{g=1}^{G}\frac{(\beta_{g\ell}-\theta_\ell)^2}{\xi_{g\ell}}\right)
\end{aligned}$$

Now, the density $p(\sigma_\ell|s_\ell)$ (the uniform density from 0 to s_ℓ) is equivalent to $p(\sigma_\ell^2|s_\ell) = (\sigma_\ell^2)^{-1/2}\text{I}(\sigma_\ell^2 < s_\ell^2)$. So,

$$\begin{aligned}
p(\sigma_\ell^2|\ldots) &\propto (\sigma_\ell^2)^{-1/2}\text{I}(\sigma_\ell^2 < s_\ell^2)(\sigma_\ell^2)^{-G/2}\exp\left(-\frac{1}{\sigma_\ell^2}\frac{1}{2}\sum_{g=1}^{G}\frac{(\beta_{g\ell}-\theta_\ell)^2}{\xi_{g\ell}}\right) \\
&= (\sigma_\ell^2)^{-G/2-1/2}\exp\left(-\frac{1}{\sigma_\ell^2}\frac{1}{2}\sum_{g=1}^{G}\frac{(\beta_{g\ell}-\theta_\ell)^2}{\xi_{g\ell}}\right)\text{I}(\sigma_\ell^2 < s_\ell^2) \\
&= (\sigma_\ell^2)^{-G/2+1/2-1}\exp\left(-\frac{1}{\sigma_\ell^2}\frac{1}{2}\sum_{g=1}^{G}\frac{(\beta_{g\ell}-\theta_\ell)^2}{\xi_{g\ell}}\right)\text{I}(\sigma_\ell^2 < s_\ell^2) \\
&= (\sigma_\ell^2)^{-(G-1)/2-1}\exp\left(-\frac{1}{\sigma_\ell^2}\frac{1}{2}\sum_{g=1}^{G}\frac{(\beta_{g\ell}-\theta_\ell)^2}{\xi_{g\ell}}\right)\text{I}(\sigma_\ell^2 < s_\ell^2) \\
&\propto \text{Inverse-Gamma}\left(\text{shape} = \frac{G-1}{2},\ \text{scale} = \frac{1}{2}\sum_{g=1}^{G}\frac{(\beta_{g\ell}-\theta_\ell)^2}{\xi_{g\ell}}\right)\text{I}(\sigma_\ell^2 < s_\ell^2)
\end{aligned}$$

B.5 ν, unknown closed form

Let Uniform$(x|a,b)$ be the uniform density for random variable x with lower bound a and upper bound b.

$$
\begin{aligned}
p(\nu \mid \cdots) &= \left[\prod_{g=1}^{G} \text{Inverse-Gamma}\left(\gamma_g \mid \text{shape} = \frac{\nu}{2},\ \text{scale} = \frac{\nu\tau}{2}\right)\right] p(\nu|0,\ d) \\
&= \prod_{g=1}^{G}\left[\Gamma\left(\frac{\nu}{2}\right)^{-1}\left(\frac{\nu\tau}{2}\right)^{\nu/2}(\gamma_g)^{-(\nu/2+1)}\exp\left(-\frac{1}{\gamma_g}\frac{\nu\tau}{2}\right)\right] p(\nu|0,\ d) \\
&= \Gamma\left(\frac{\nu}{2}\right)^{-G}\left(\frac{\nu\tau}{2}\right)^{G\nu/2}\left(\prod_{g=1}^{G}\gamma_g\right)^{-(\nu/2+1)}\exp\left(-\frac{\nu\tau}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right) p(\nu|0,\ d) \\
&\propto \Gamma\left(\frac{\nu}{2}\right)^{-G}\left(\frac{\nu\tau}{2}\right)^{G\nu/2}\left(\prod_{g=1}^{G}\gamma_g\right)^{-\nu/2}\exp\left(-\frac{\nu\tau}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right) p(\nu|0,\ d) \\
&= \exp\left(-G\log\Gamma\left(\frac{\nu}{2}\right) + \frac{G\nu}{2}\log\left(\frac{\nu\tau}{2}\right) - \frac{\nu}{2}\sum_{g=1}^{G}\log\gamma_g - \frac{\nu\tau}{2}\sum_{g=1}^{G}\frac{1}{\gamma_g}\right) p(\nu|0,\ d) \\
&= \exp\left(-G\log\Gamma\left(\frac{\nu}{2}\right) + \frac{G\nu}{2}\log\left(\frac{\nu\tau}{2}\right) - \frac{\nu}{2}\sum_{g=1}^{G}\left[\log\gamma_g + \frac{\tau}{\gamma_g}\right]\right) \text{Uniform}(\nu|0,\ d) \\
&= \exp\left(-G\log\Gamma\left(\frac{\nu}{2}\right) + \frac{G\nu}{2}\log\left(\frac{\nu\tau}{2}\right) - \frac{\nu}{2}\sum_{g=1}^{G}\left[\log\gamma_g + \frac{\tau}{\gamma_g}\right]\right) \text{I}(0 < \nu < d)
\end{aligned}
$$

B.6 ε_{gn}, unknown closed form

Let $\lambda_{gn} = \exp\left(h_n + \varepsilon_{gn} + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right)$.

$$
\begin{aligned}
p(\varepsilon_{gn} \mid \cdots) &= \text{Poisson}(y_{gn} \mid \lambda_{gn}) \cdot \text{Normal}(\varepsilon_{gn} \mid 0, \gamma_g) \\
&\propto \lambda_{gn}^{y_{gn}} \exp(-\lambda_{n,g}) \exp\left(-\frac{\varepsilon_{gn}^2}{2\gamma_g}\right) \\
&= \exp\left(y_{gn} \log \lambda_{gn} - \lambda_{n,g} - \frac{\varepsilon_{gn}^2}{2\gamma_g}\right) \\
&= \exp\left(y_{gn}\left(h_n + \varepsilon_{gn} + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right) - \exp\left(h_n + \varepsilon_{gn} + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right) - \frac{\varepsilon_{gn}^2}{2\gamma_g}\right) \\
&\propto \exp\left(y_{gn}\varepsilon_{gn} - \exp\left(\varepsilon_{gn}\right) \exp\left(h_n + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right) - \frac{\varepsilon_{gn}^2}{2\gamma_g}\right) \\
&= \exp\left(y_{gn}\varepsilon_{gn} - \frac{\varepsilon_{gn}^2}{2\gamma_g} - \exp\left(\varepsilon_{gn}\right) \exp\left(h_n + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right)\right)
\end{aligned}
$$

B.7 $\beta_{g\ell}$, unknown closed form

Let $\lambda_{gn} = \exp\left(h_n + \varepsilon_{gn} + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right)$.

$$
\begin{aligned}
p(\beta_{g\ell} \mid \cdots) &= \left[\prod_{n=1}^{N} \text{Poisson}(y_{gn} \mid \lambda_{gn})\right] \cdot \text{Normal}(\beta_{g\ell} \mid \theta_\ell, \sigma_\ell^2 \xi_{g\ell}) \\
&\propto \left[\prod_{n=1}^{N} \lambda_{gn}^{y_{gn}} \exp(-\lambda_{gn})\right] \cdot \exp\left(-\frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}}\right) \\
&= \exp\left(\sum_{n=1}^{N} \left[y_{gn} \log \lambda_{gn} - \lambda_{gn}\right] - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}}\right) \\
&= \exp\left(\sum_{n=1}^{N} \left[y_{gn}\left(h_n + \varepsilon_{gn} + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right) - \exp\left(h_n + \varepsilon_{gn} + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right)\right]\right. \\
&\quad \left. - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}}\right) \\
&\propto \exp\left(\sum_{n=1}^{N} \left[y_{gn} X_{n\ell}\beta_{g\ell} - \exp\left(h_n + \varepsilon_{gn} + \sum_{i=1}^{L} X_{ni}\beta_{gi}\right)\right] - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}}\right) \\
&= \exp\left(\sum_{n=1}^{N} y_{gn} X_{n\ell}\beta_{g\ell} - \sum_{n=1}^{N} \exp\left(h_n + \varepsilon_{gn} + X_{n\ell}\beta_{g\ell} + \sum_{i \neq l} X_{ni}\beta_{gi}\right)\right. \\
&\quad \left. - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}}\right) \\
&= \exp\left(\beta_{g\ell} \sum_{n=1}^{N} y_{gn} X_{n\ell} - \sum_{n=1}^{N} \exp\left(X_{n\ell}\beta_{g\ell}\right) \exp\left(h_n + \varepsilon_{gn} + \sum_{i \neq l} X_{ni}\beta_{gi}\right)\right. \\
&\quad \left. - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}}\right) \\
&= \exp\left(\beta_{g\ell} \sum_{n=1}^{N} y_{gn} X_{n\ell} - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}}\right. \\
&\quad \left. - \sum_{n=1}^{N} \exp\left(X_{n\ell}\beta_{g\ell}\right) \exp\left(h_n + \varepsilon_{gn} + \sum_{i \neq l} X_{ni}\beta_{gi}\right)\right)
\end{aligned}
$$

Now, let S_ℓ be the set of unique nonzero elements of $\{X_{1\ell}, \ldots, X_{N\ell}\}$. Then,

$$
\begin{aligned}
p(\beta_{g\ell} \mid \cdots) \propto \exp\Bigg(& \beta_{g\ell} \sum_{n=1}^{N} y_{gn} X_{n\ell} - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}} \\
& - \sum_{x \in S_\ell} \sum_{n=1}^{N} \mathrm{I}(X_{n\ell} = x) \exp\left(x \beta_{g\ell}\right) \exp\left(h_n + \varepsilon_{gn} + \sum_{i \neq l} X_{ni} \beta_{gi} \right) \Bigg) \\
= \exp\Bigg(& \beta_{g\ell} \sum_{n=1}^{N} y_{gn} X_{n\ell} - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}} \\
& - \sum_{x \in S_\ell} \exp\left(x \beta_{g\ell}\right) \sum_{n=1}^{N} \mathrm{I}(X_{n\ell} = x) \exp\left(h_n + \varepsilon_{gn} + \sum_{i \neq l} X_{ni} \beta_{gi} \right) \Bigg)
\end{aligned}
$$

B.8 $\xi_{g\ell}$, unknown closed form

The full conditional density of $\xi_{g\ell}$ up to a proportionality constant is given by

$$
\begin{aligned}
p(\xi_{g\ell} | \ldots) &\propto p(\beta_{g\ell} | \theta_\ell, \sigma_\ell^2, \xi_{g\ell}) p(\xi_{g\ell} | k_\ell, q_\ell, r_\ell) \\
&\propto \xi_{g\ell}^{-1/2} \exp\left(-\frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}} \right) p(\xi_{g\ell} | k_\ell, q_\ell, r_\ell)
\end{aligned}
$$

The choice of $p(\xi_{g\ell} | k_\ell, q_\ell, r_\ell)$ determines the full conditional of $\xi_{g\ell}$ and the marginal hierarchical distribution of $\beta_{g\ell}$. Suppose

$$
p(\xi_{g\ell} | k_\ell, q_\ell, r_\ell) = \text{Exponential}(\xi_{g\ell} | \text{rate} = k_\ell) \qquad (\text{with expectation } k_\ell^{-1})
$$

Then, the full conditional of $\xi_{g\ell}$ becomes

$$
\begin{aligned}
p(\xi_{g\ell} | \ldots) &\propto \xi_{g\ell}^{-1/2} \exp\left(-\frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}} \right) \exp\left(-k_\ell \xi_{g\ell} \right) && \text{(B.1)} \\
&= \xi_{g\ell}^{-1/2} \exp\left(-\frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}} - k_\ell \xi_{g\ell} \right) && \text{(B.2)} \\
&= \exp\left(-\frac{1}{2} \log \xi_{g\ell} - \frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2} \frac{1}{\xi_{g\ell}} - k_\ell \xi_{g\ell} \right) && \text{(B.3)}
\end{aligned}
$$

The resulting marginal hierarchical distribution of $\beta_{g\ell}$ is a Laplace distribution. Letting $s = \xi_{g\ell}$, $z = (\beta_{g\ell} - \theta_\ell)/\sqrt{\sigma_\ell^2}$, and $a = \sqrt{2k_\ell}$, Equation (B.1) becomes

$$p(\xi_{g\ell}|\ldots) \propto \frac{1}{\sqrt{s}} e^{-z^2/(2s)} e^{-a^2 s/2}$$

$$p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2, \xi_{g\ell}) p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell) \propto \frac{1}{\sqrt{2\pi s}} e^{-z^2/(2s)} \frac{a^2}{2} e^{-a^2 s/2}$$

The next step is an integration. Equation 4 by Park and Casella (2008) allows us to proceed from Equation (B.5) to Equation (B.6) below.

$$p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2) = \int_0^\infty p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2, \xi_{g\ell}) p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell) d\xi_{g\ell} \tag{B.4}$$

$$= \int_0^\infty \frac{1}{\sqrt{2\pi s}} e^{-z^2/(2s)} \frac{a^2}{2} e^{-a^2 s/2} ds \tag{B.5}$$

$$= \frac{a}{2} e^{-a|z|} \tag{B.6}$$

$$= \frac{\sqrt{2k_\ell}}{2} \exp\left(-\sqrt{2k_\ell} \left| \frac{\beta_{g\ell} - \theta_\ell}{\sqrt{\sigma_\ell^2}} \right| \right) \tag{B.7}$$

$$\propto \exp\left(-\sqrt{2k_\ell} \left| \frac{\beta_{g\ell} - \theta_\ell}{\sqrt{\sigma_\ell^2}} \right| \right) \tag{B.8}$$

$$= \exp\left(- \left| \frac{\beta_{g\ell} - \theta_\ell}{\sqrt{\sigma_\ell^2/2k_\ell}} \right| \right) \tag{B.9}$$

Hence,

$$p(\beta_{g\ell}|\theta_\ell, \sigma_\ell^2) = \text{Laplace}\left(\text{location} = \theta_\ell,\ \text{scale} = \sqrt{\frac{\sigma_\ell^2}{2k_\ell}} \right)$$

The mean is θ_ℓ, and the Laplace variance is twice the scale squared: i.e., $2\sqrt{\frac{\sigma_\ell^2}{2k_\ell}}^2 = \frac{\sigma_\ell^2}{k_\ell}$. Taking $k_\ell = 1$ as in our analyses, the variance becomes σ_ℓ^2.

Now, suppose $p(\xi_{g\ell}|k_\ell, q_\ell, r_\ell) = \text{Inverse-Gamma}(\text{shape} = q_\ell,\ \text{scale} = r_\ell)$. Then,

$$p(\xi_{g\ell}|\ldots) \propto \xi_{g\ell}^{-1/2} \exp\left(-\frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}} \right) \xi_{g\ell}^{-q_\ell - 1} \exp\left(-\frac{r_\ell}{\xi_{g\ell}} \right) \tag{B.10}$$

$$= \xi_{g\ell}^{-1/2 - q_\ell - 1} \exp\left(-\frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}} - \frac{r_\ell}{\xi_{g\ell}} \right) \tag{B.11}$$

$$= \xi_{g\ell}^{-q_\ell - 3/2} \exp\left(-\frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2 \xi_{g\ell}} - \frac{r_\ell}{\xi_{g\ell}} \right) \tag{B.12}$$

$$= \exp\left(-\left(q_\ell + \frac{3}{2} \right) \log \xi_{g\ell} - \frac{1}{\xi_{g\ell}} \left(\frac{(\beta_{g\ell} - \theta_\ell)^2}{2\sigma_\ell^2} + r_\ell \right) \right) \tag{B.13}$$

The resulting marginal hierarchical distribution of $\beta_{g\ell}$ is derived as follows.

$$
\begin{aligned}
p(\beta_{g\ell}|\theta_\ell,\sigma_\ell^2) &= \int_0^\infty p(\beta_{g\ell}|\theta_\ell,\sigma_\ell^2,\xi_{g\ell})p(\xi_{g\ell}|k_\ell,q_\ell,r_\ell)d\xi_{g\ell} \\
&\propto \int_0^\infty p(\xi_{g\ell}|\ldots)d\xi_{g\ell} \\
&\propto \int_0^\infty \xi_{g\ell}^{-q_\ell-3/2}\exp\left(-\frac{1}{\xi_{g\ell}}\left(\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2}+r_\ell\right)\right)d\xi_{g\ell} \qquad \text{(Equation B.12)} \\
&= \int_0^\infty x^{-a-1}\exp\left(-\frac{b}{x}\right)dx \quad \left(x=\xi_{g\ell},\ a=q_\ell+1/2,\ b=\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2}+r_\ell\right) \\
&= \Gamma(a)b^{-a} \qquad \text{(integrating an inverse-gamma distribution)} \\
&= \Gamma\left(q_\ell+\frac{1}{2}\right)\left(\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2}+r_\ell\right)^{-q_\ell-1/2} \\
&\propto \left(\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2}+r_\ell\right)^{-q_\ell-1/2} \\
&= \left(\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2}+r_\ell\right)^{-(2q_\ell-1)/2} \\
&= \left(\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2}+r_\ell\right)^{-(2q_\ell-1)/2} \\
&= r_\ell^{-(2q_\ell-1)/2}\left(\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2 r_\ell}+1\right)^{-(2q_\ell-1)/2} \\
&\propto \left(\frac{(\beta_{g\ell}-\theta_\ell)^2}{2\sigma_\ell^2 r_\ell}+1\right)^{-(2q_\ell-1)/2} \\
&= \left(\frac{1}{2q_\ell}\frac{(\beta_{g\ell}-\theta_\ell)^2}{\sigma_\ell^2 r_\ell/q_\ell}+1\right)^{-(2q_\ell-1)/2}
\end{aligned}
$$

which is the kernel of a shifted and scaled t_{2q_ℓ} distribution. The center is θ_ℓ, and the variance is

$$
\begin{aligned}
\mathrm{Var}(\beta_{g\ell}|\theta_\ell,\sigma_\ell^2) =& \mathrm{Var}(t_{2q_\ell})\cdot\sqrt{\frac{\sigma_\ell^2 r_\ell}{q_\ell}}^2 \\
=& \frac{2q_\ell}{2q_\ell-2}\frac{r_\ell}{q_\ell}\sigma_\ell^2 \\
=& \frac{r_\ell}{q_\ell-1}\sigma_\ell^2
\end{aligned}
$$

Taking $r_\ell = 2$ and $q_\ell = 3$ as in our analyses, $\mathrm{Var}(\beta_{g\ell}|\theta_\ell,\sigma_\ell^2)$ becomes σ_ℓ^2. For a summary of the full conditionals of $\xi_{g\ell}$ and the resulting hierarchical marginals of $\beta_{g\ell}$, please refer to Table B.1

APPENDIX C. SUPPLEMENTARY FIGURES

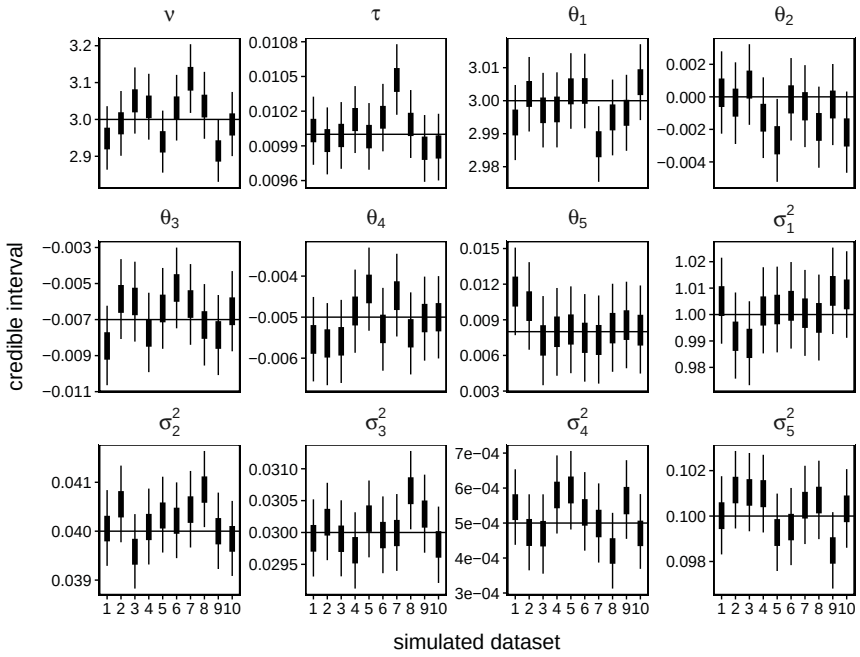

Figure S1: For Simulation Study 1 in Section 3.4.1, equal-tailed credible intervals for the hyperparameters, calculated from quantiles of MCMC samples. 50% credible intervals are shown as thick vertical lines, and 95% credible intervals are overlaid as narrow vertical lines.

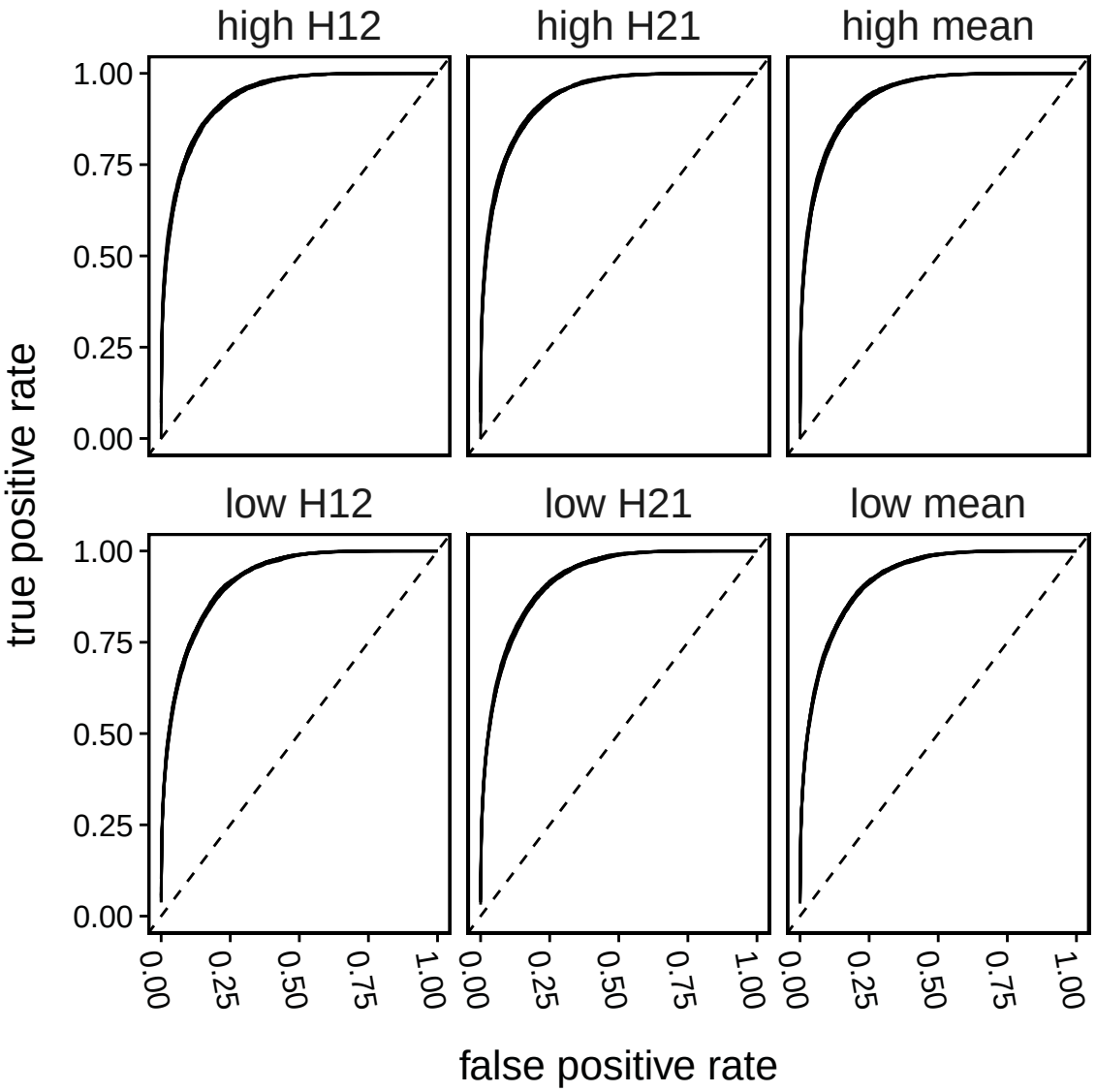

Figure S2: Receiver operating characteristic (ROC) curves for Simulation Study 1 in Section 3.4.1. There is one curve for each dataset and each kind of heterosis, and the dashed line is the identity line, the expected results of an ordering of genes completely at random. Areas under the curves range from 0.916 to 0.922 for low-parent heterosis and from 0.930 to 0.936 for high-parent heterosis.

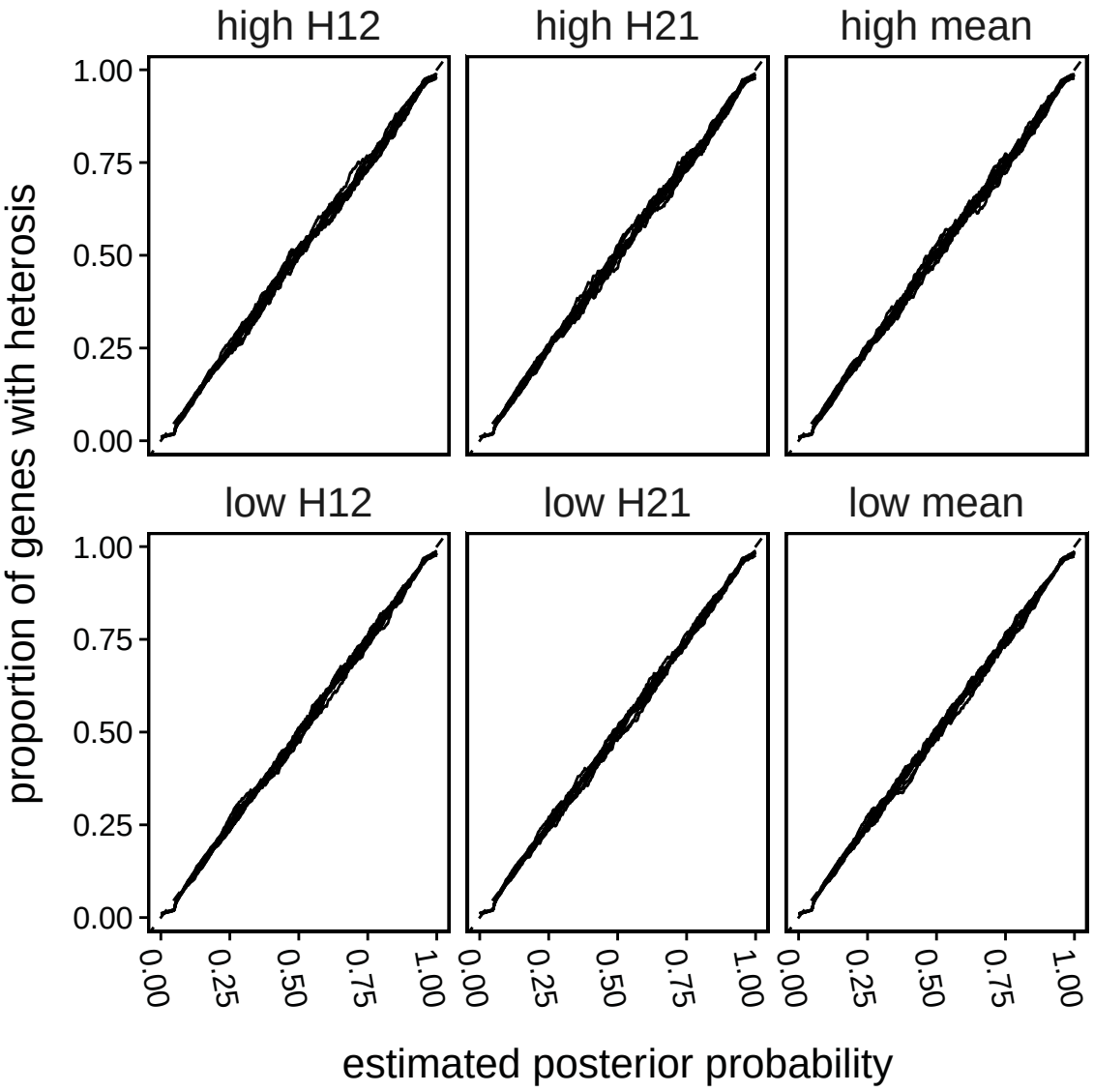

Figure S3: Calibration curves for Simulation Study 1 in Section 3.4.1. There is one curve for each dataset and each kind of heterosis. Each curve is the kernel-smoothed local proportion of true heterosis genes plotted against estimated probability from our fully Bayesian approach. The dashed line, hidden by the calibration curves, is identity line (the ideal calibration curve).

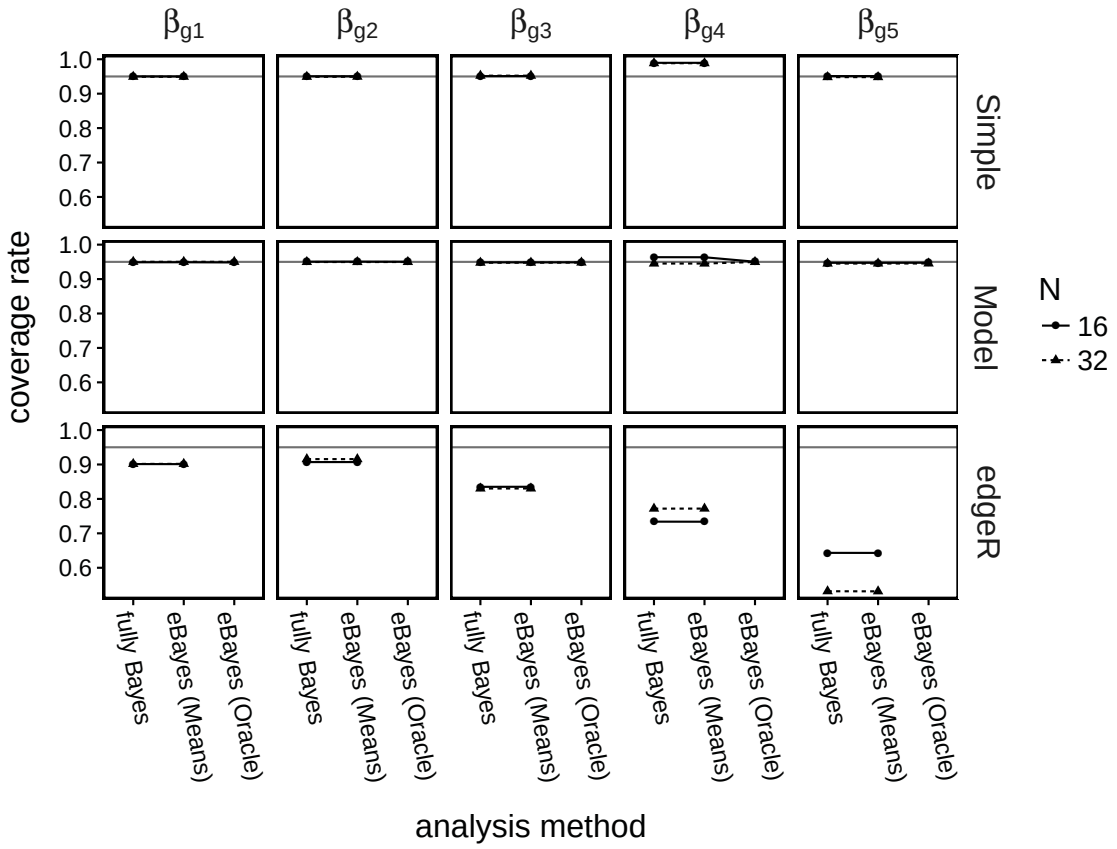

Figure S4: For Simulation Study 2 in Section 3.4.2, observed rates at which estimated 95% credible intervals cover parameters $\beta_{g\ell}$. The column labels indicate the $\beta_{g\ell}$ parameters, and the row labels indicate simulation scenarios. The gray horizontal lines indicate 0.95, the nominal coverage rate.

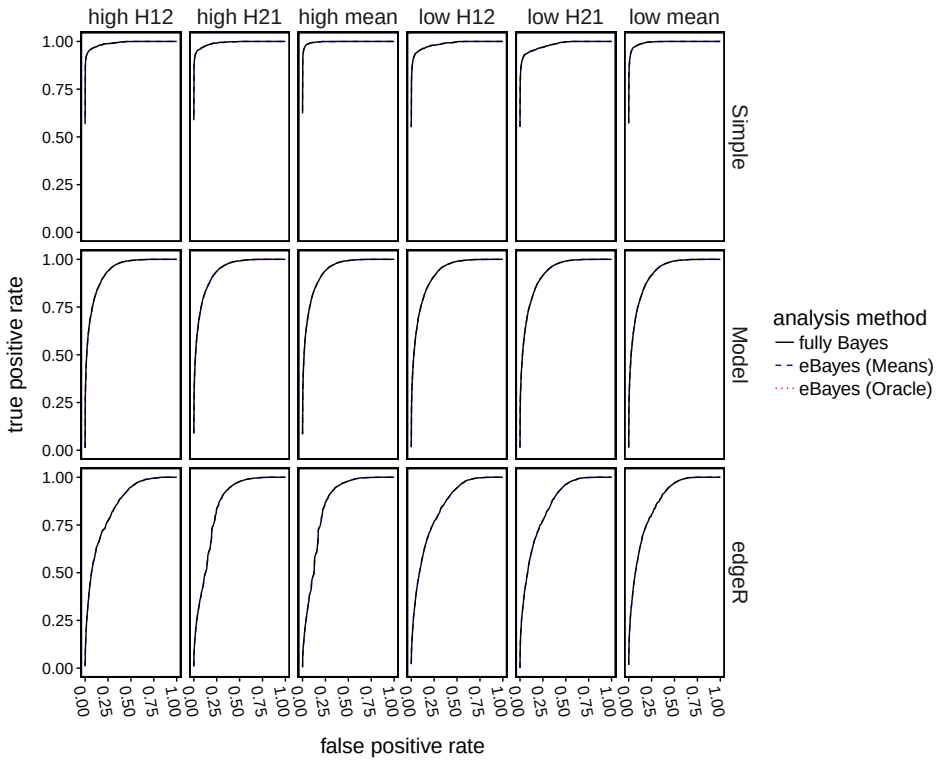

Figure S5: Receiver operating characteristic (ROC) curves for datasets with $N = 16$ in Simulation Study 2 in Section 3.4.2. The row labels indicate the method of simulating the data, and the column labels indicate the kind of heterosis detected. For heterosis, we use the notation from the general plant heterosis scenario from Section 3.2 and Table 3.1.

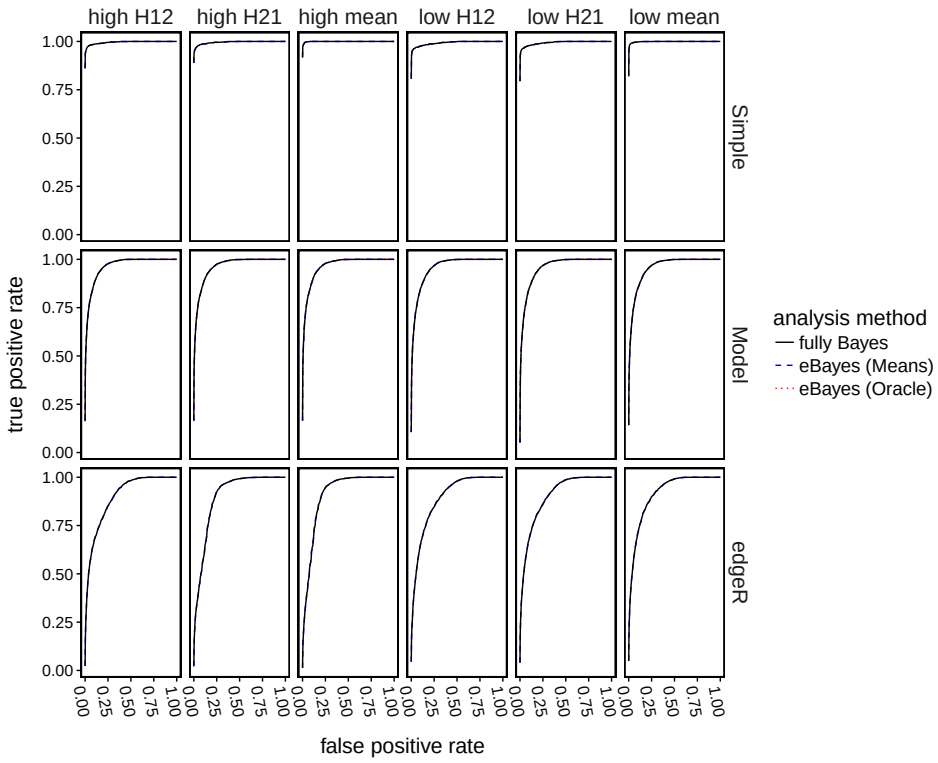

Figure S6: Receiver operating characteristic (ROC) curves for datasets with $N = 32$ in Simulation Study 2 in Section 3.4.2. The row labels indicate the method of simulating the data, and the column labels indicate the kind of heterosis detected. For heterosis, we use the notation from the general plant heterosis scenario from Section 3.2 and Table 3.1.

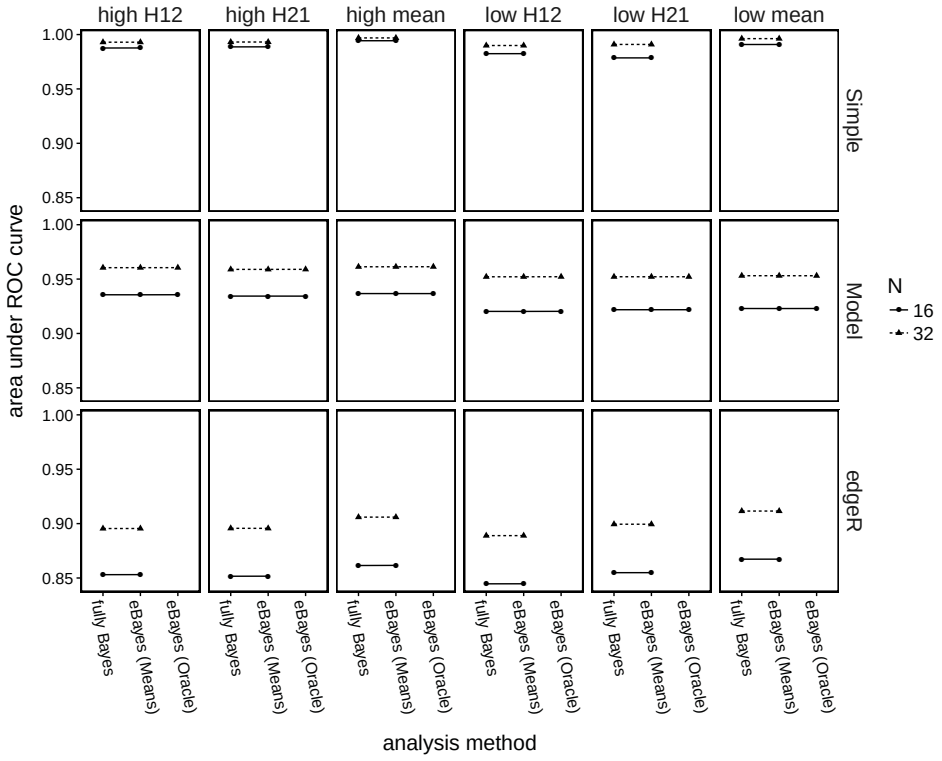

Figure S7: Areas under the receiver operating characteristic (ROC) curves for Simulation Study 2 in Section 3.4.2. The plotting shape denotes sample size ($N = 16$ or $N = 32$), the row labels indicate the method of simulating the data, the column labels indicate the kind of heterosis detected. For heterosis designations, we use the notation from the general plant breeding scenario from Section 3.2 and Table 3.1.

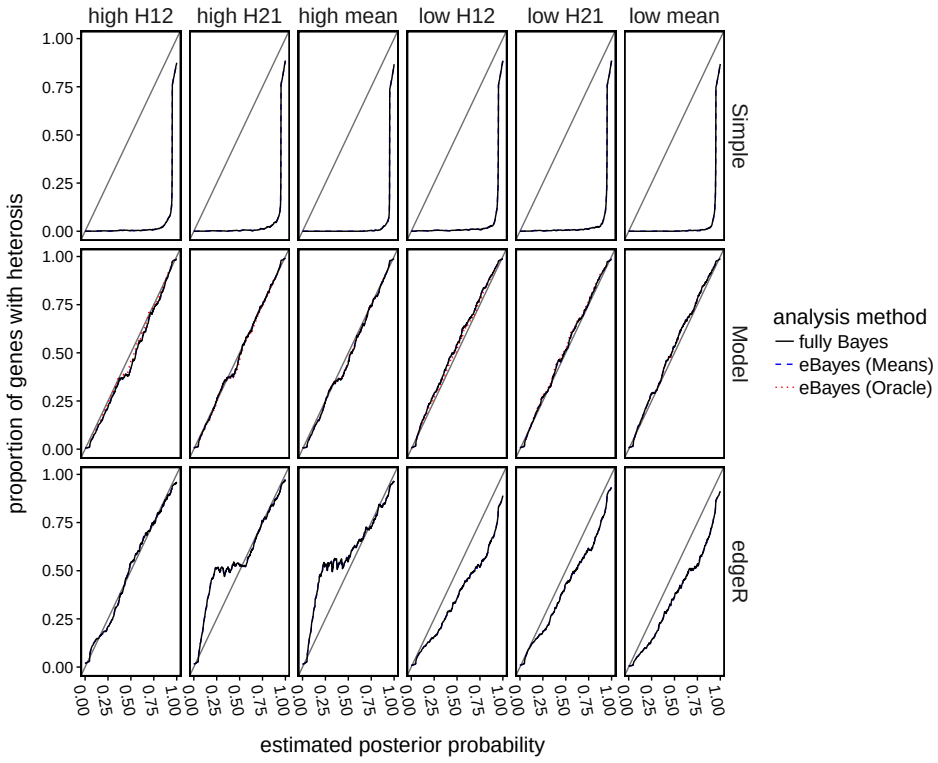

Figure S8: For datasets with $N = 32$ in Simulation Study 2 in Section 3.4.2, calibration curves for heterosis gene detection. A calibration curve, as explained in Section 3.4, is the smoothed local true proportion of heterosis genes plotted against posterior heterosis probability estimates from a statistical analysis. The identity line, plotted in solid gray in each panel, is the ideal calibration curve, which would result from perfectly accurate posterior probabilities. The type of heterosis detected is indicated above each column, where we use the notation from the general plant heterosis scenario from Section 3.2 and Table 3.1.The row labels indicate the method of simulating the data.

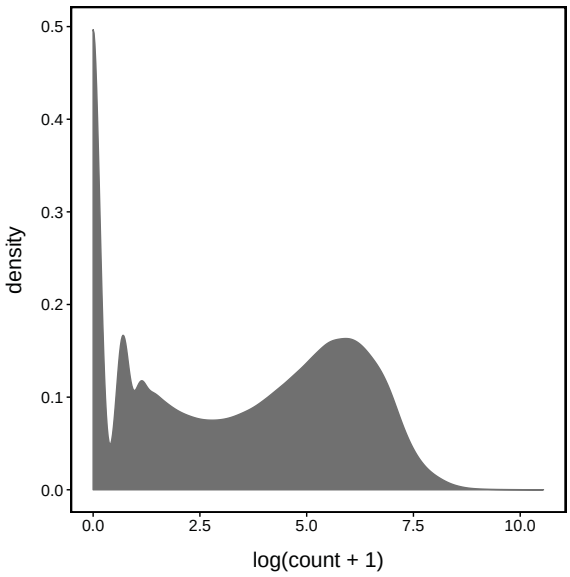

Figure S9: For the Paschold et al. dataset from Section 3.2, a kernel density estimate of the log of the counts after incrementing by 1.

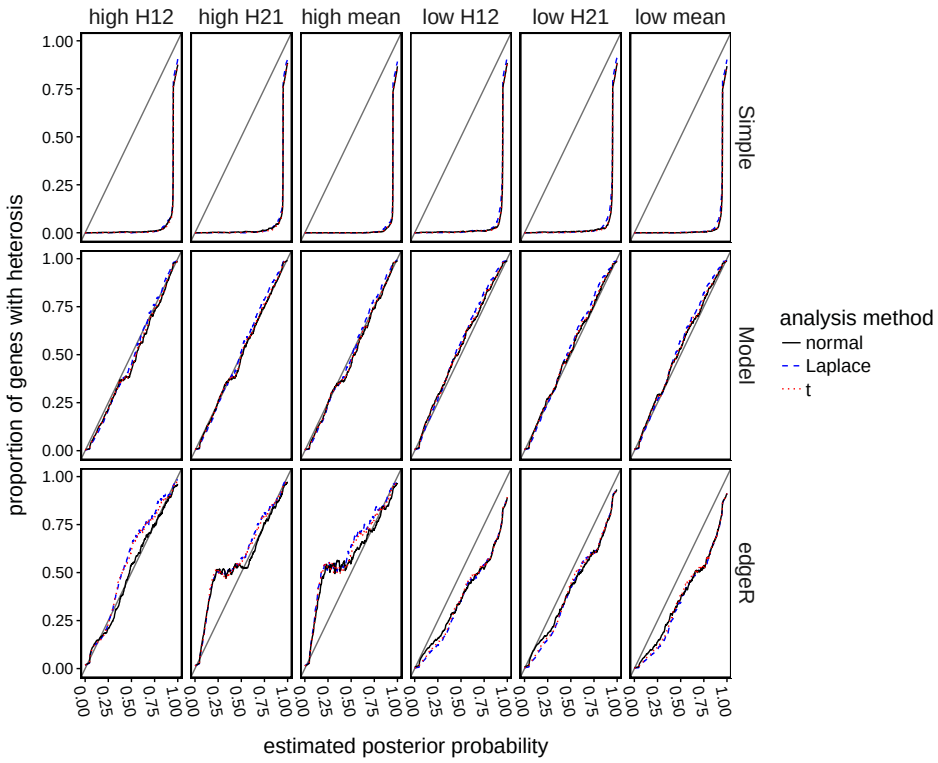

Figure S10: For datasets with $N = 32$ in Simulation Study 2 from Chapter 3 Section 3.4.2, calibration curves for heterosis gene detection. The data-generating mechanisms explained by Chapter 3 Section 3.4.2 are indicated in the row labels. The types of posterior probabilities calculated, as explained by Section 3.2, Section 3.1, and Table 3.3.2 of Chapter 3 are indicated in the column labels.

NOTES

Drs. Jarad Niemi and Dan Nettleton of Iowa State University oversaw and guided this work, the publication of which in peer-reviewed journals is pending.