Applications of Bayesian hierarchical models in gene expression and product reliability

by

Eric T. Mittman

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 Niemi, Major Professor

Peng Liu

William Q. Meeker

Dan Nettleton

Steve Vardeman

Iowa State University

Ames, Iowa

2017

DEDICATION

I dedicate this work to my wife, Jessa, and to our children: Jonah, Eliza and Ruth.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

Advances in modern computing have encouraged statisticians to fit larger and larger models to larger and more complex data sets. Bayesian hierarchical models are a class of models, suitable for a wide range of applications, that offer the analyst flexibility and for which general strategies for inference have been developed. In this work we present two such models, both motivated by real applications, and develop methodologies for performing inference.

First, we present a Bayesian nonparametric hierarchical regression model for gene expression profiling data. In gene profiling studies, a relatively small number of observational units produce data used to test hypotheses for tens of thousands of genes. This is a $n \ll p$ problem with the potential of producing many incorrect results, due to random noise. To mitigate this problem, we propose a nonparametric model which considers the set of regression parameters for each gene as independent, identically distributed random variables, having a joint distribution with an unspecified form.

Second, we present a method for estimation of lifetime for populations exhibiting heterogeneity due to infant mortality. Specifically, we consider the case where multiple such populations are of interest and information for some populations is limited by censoring and truncation. We demonstrate our method on a large set of field reliability data collected on hard drives.

CHAPTER 1. Introduction

The rapid progress in computational technologies has made feasible general computational methodologies for Bayesian statistical inference, especially Markov chain Monte Carlo (MCMC). The same conditions are also responsible for a huge increase in the amount of data available for analysis. In this work we consider two applications, gene expression and product reliability. While these applications are quite different, there exists a common problem for the analyst — namely, the desirable model suffers from low statistical power, due to small sample sizes — with a common solution: hierarchical modeling. Hierarchical, or multi-level, modeling is a general term for stochastic models where model parameters governing the assumed data-generation process are themselves given a model with some specified form. Hierarchical models are a well-known tool, commonly used across many disciplines, e.g., linear models with random effects.

This dissertation consists of three papers relating to two new applications of hierarchical models. In Chapter 2, we consider the feasibility of a flexible Bayesian nonparametric hierarchical model for gene expression data. Recent innovations in high-throughput RNA sequencing have made it possible to measure, within a single sample, the relative expression of tens of thousands of genes simultaneously (Wang et al., 2009). Many researchers are looking to use such gene expression data to learn which genes may be involved in biological phenomena. Because typical studies are limited in the number of samples they can afford to sequence, the statistical power for detecting differences between experimental groups tends to be quite low. Hierarchical modeling is useful in this context because it provides a mechanism by which partial pooling of information stabilizes estimation of differences and reduces the rate of false detection while not depending strongly on tuning parameters. Some popular methods use hierarchical models but make unrealistic assumptions, which in some cases could lead to inefficient partial pooling of information. We propose the application of a Bayesian nonparametric approach (Ishwaran and James, 2001; Liu et al., 2015) that avoids these assumptions and implement a Gibbs sampler on a GPU which allows posterior inference to be computationally feasible (Suchard et al., 2010; Landau and Niemi, 2016).

In Chapter 3, we apply our model from Chapter 2 to an experimental data set by Paschold et al. (2012), where key questions centered on the ordering of mean expression among four varieties of maize, two different inbred varieties and the two hybrids crosses. We conduct two simulation studies, both of which show that the new method produces estimates that are substantially closer to the truth, on average, than all competing methods that we considered (Smyth, 2005; Robinson et al., 2010; Love et al., 2014). We also used these studies to study the accuracy in ranking genes according with respect to a hypothesis; we found that the new method was at least as accurate as competing methods. An additional benefit of our approach is it is able to directly assess the probability of a particular ordering of expression levels. This is because we sample jointly from the full posterior. We find that this method achieves a greater degree of shrinkage than similar parametric approaches and that the pattern of shrinkage respects local patterns in the data which may represent relevant biological processes.

Chapter 4 considers the problem of modeling failure data for multiple populations where the failures exhibit multimodality and where much of the data is truncated and/or censored. Hierarchical modeling can help here because, by borrowing information across groups, we can more accurately assess uncertainty and also produce full inference for groups with very limited data. To deal with the multimodality, we present a hierarchical version of the generalized limited failure population model of Chan and Meeker (1999). We demonstrate fitting this model on a fairly large data set consisting of lifetime information on over 75,000 hard drives, representing 44 different drive-models. Here we illustrate the flexibility afforded by the Bayesian approach using posterior samples, obtained using Stan Development Team (2015), to perform model selection, model assessment, estimation and prediction.

Finally, in Chapter 5, I summarize our main findings and consider potential modifications or improvements to what we have done and directions for future work.

CHAPTER 2. A GPU accelerated nonparametric shrinkage model for high-dimensional data

2.1 Introduction

Experiments or observational studies that produce a large number of measurements for each of a small to moderate number of experimental/observational units present certain statistical challenges. Such problems are increasingly common as technologies produce more and more data. For example, gene expression profiling studies consider comparisons between experimental groups at thousands of genes on the genome.

When similar comparisons are to be made between subjects for each measurement component, the number of errors is likely to be large. Because of this, there is demand for statistical methods that reduce the number of 'false positives'. Shrinkage priors are often used to improve inference by borrowing information across components as a moderating influence given the limited information available for a single component. Correlations across measurement components may exist, e.g. interactions among genes, but in such scenarios as we are describing, any attempt at inference would be extremely dubious. Instead, the information being borrowed across components is with regard to between subject comparisons, such as treatment effects in the case of an experiment.

A common approach is to model component-specific parameters as independent, identically distributed according to some probability distribution, $\mathcal{P}$, belonging to a parametric family of distributions. Under this framework, $\mathcal{P}$ can then be estimated allowing for conditional inference, or a prior distribution can be given on its parameters allowing for fully Bayesian inference. While this approach can be useful in practice, the influence of the choice of a particular parametric family can be considerable. This choice is often not reflective of *a priori* information, but rather due to convention or convenience. Here, the choice of a ‘nonparametric prior’, i.e. a prior distribution over a ‘large’ set of probability distributions, allows one to avoid having to make parametric assumptions about $\mathcal{P}$.

Much effort has gone into researching methods in the case of gene expression. Several have been released as R packages (Robinson et al., 2010; Love et al., 2014; Law et al., 2014). A common feature of these methods is that they use between-gene information to estimate the the mean-variance relationship nonparametrically. This relationship can be key to achieving statistical power; it involves both biological and technological sources of variation and varies from dataset to dataset (Law et al., 2014). Empirically, we have observed structure to be present not only between the mean and variance, but also among regression parameter components (see Fig. 2.1). This suggests that between-gene information is available not only to regulate gene-level assessment of variance, but also the estimated effects themselves. While others used random effects models for individual regression components (Love et al., 2014; Landau and Niemi, 2016)], they depend on assumptions about the underlying distribution of effects, such as independence, which are contradicted by the data.

A special case of gene expression profiling concerning two populations is differential gene expression. Because expression is typically modeled on the log scale, the differential expression at a particular gene, termed ‘log-fold-change’, is the parameter of interest. Liu et al. (2015) proposed a semiparametric model treating the log-fold change parameters as random effects following an unknown distribution, $\mathcal{P}$, where $\mathcal{P}$ is a random distribution arising from a Dirichlet process (DP). In this paper, we modify and extend the approach taken by Liu et al. (2015) to allow nonparametric inference for a large class of gene profiling experiments. In our method, we use a DP to model jointly the distribution of the p-dimensional gene-specific regression coefficient, β_g, and variance, σ_g^2.

Common practice for Bayesian nonparametric applications is to sample from the joint posterior distribution using a Gibbs sampler. Several Gibbs sampling algorithms for DP mixture models have been proposed. These can be categorized into two types: ‘marginal,’ where full conditionals are with respect to the joint posterior with the unknown $\mathcal{P}$ integrated out, and ‘conditional’, where $\mathcal{P}$ is also sampled and conditioned on.

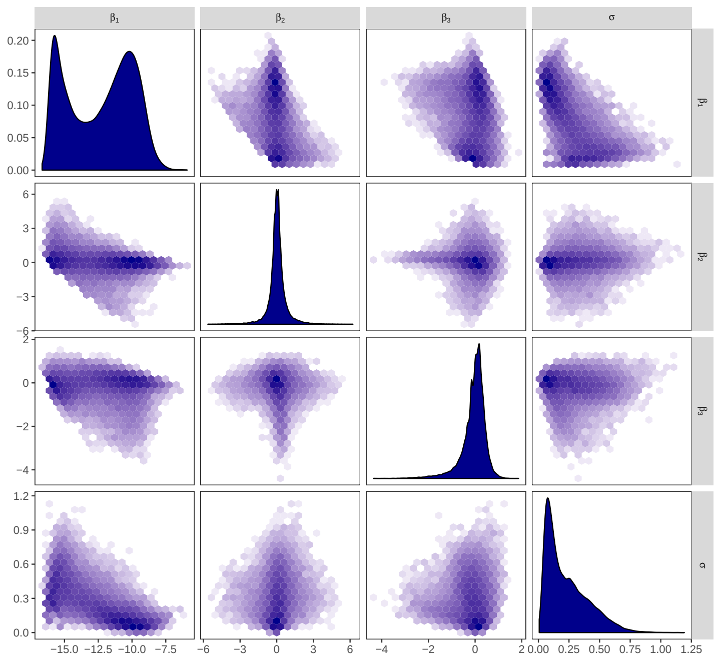


Figure 2.1: Bivariate histograms of independent estimates of effects and standard deviation obtained by ordinary least squares for 36,081 genes (data from Paschold et al. (2012)). This example shows that random effects models assuming normality and/or independence may not be suitable for modeling gene expression hierarchically. For more on the data, including definitions of β_1, β_2, β_3, and σ, see Section 2.7.

When $\mathcal{P}$ is not of particular interest, marginal approaches are often preferable. However, for large G, these algorithms are not computationally tractable as they require one-at-a-time updating of the cluster assignment for each g conditioning on all $g' \neq g$. On the other hand, conditional Gibbs methods, which depend on the "stick-breaking" representation of the DP Sethuraman (1994), are amenable to parallelism because cluster assignment is conditionally independent given $\mathcal{P}$. Suchard et al. (2010) argued for wider use of graphics processing units (GPUs) by statisticians for computationally demanding, parallelizable tasks to achieve large speed-ups in real time. We provide a brief introduction to computing in the massively parallel context, including some general guidelines that can help in designing implementations that are well-suited for the GPU. We demonstrate that using a Dirichlet process mixture as a 'shrinkage prior' is feasible in practice by utilizing currently available GPU technology and describe our implementation.

Section 3.2 introduces the structure of gene expression profiling data suitable for our method and recommended preprocessing steps. Section 3.3.2 presents our model for gene expression, which features a hierarchical model for the gene-specific parameters without parametric assumptions. Next, Section 2.4 outlines a procedure for posterior sampling based on the blocked Gibbs sampler of Ishwaran and Zarepour (2000). Section 2.5 introduces concepts related to programming for GPU parallelism and discusses details of two important parallel algorithms. The Gibbs sampler is revisited and GPU implementation details are provided. To investigate the properties of our algorithm, in Section 2.6 we discuss a study we conducted to assess the time requirement under variable conditions. Section 2.7 presents an analysis of differential expression data and contrasts our results with both a gene-independent analysis and a mixed-effects analysis assuming a multivariate normal distribution on the random effects. Finally, in Section 3.6 we put our proposed method in context, discuss potential applications and consider future directions for research.

2.2 Data

Our method is intended for cases where sample size is limited and each sample provides a high-dimensional response measurement. Such data can be represented in a tabular format, with each row associated with a component of the response and each column with an experimental/observational unit. An example is shown in Table 2.1. We assume that a design matrix encoding the important relationships between samples is known and can be applied equally to all response components.

Important cases include RNA-seq read counts and microarray data. For these cases, the design matrix for the samples could include information about treatments, experimental design, known genotypic and phenotypic relationships. Metadata for the genes themselves are not utilized.

Because the read count totals are right-skewed, we use a logarithmic transformation of the RNA-seq counts, performing the analysis instead with log counts normalized by the effective library size. The details are given in Section 2.7. Table 2.3 shows the result of this transformation applied to the data in Table 2.1.

Table 2.1: Sample of RNA-seq total read counts from Paschold et al. (2012). Columns names identify samples, including the genotype and a replicate number, from which we can infer the flow cell that was used for sequencing. Gene annotation information (regarding the rows), is not used for in our analysis. The first and last genes listed strongly indicate differential expression between the two inbred maize lines.

	$B73_1$	$B73_2$	$B73_3$	$B73_4$	$Mo17_1$	$Mo17_2$	$Mo17_3$	$Mo17_4$
$gene_1$	666	590	654	703	3	3	1	1
$gene_2$	414	422	383	416	392	346	402	351
$gene_3$	1525	1530	904	833	1688	1568	1413	1377
$gene_4$	12	11	5	2	8	20	9	6
$gene_5$	1	1	0	2	951	945	1157	867

2.3 Bayesian nonparametric hierarchical regression model

Let y_{gn} represent the observed expression (possibly after transformation) for component g, sample n. Let $x_n^\top$ be the row of the design matrix X corresponding to sample n. We will use the upper case letters G and N, to denote the number of components and samples, respectively. For the observed data, we assume the data model

$$y_{gn} \overset{ind.}{\sim} \mathrm{N}\left(x_n^\top \beta_g, \sigma_g^2\right), \tag{2.1}$$

We model jointly

$$\left(\beta_g^\top, \sigma_g^2\right) \overset{ind.}{\sim} \mathcal{P},$$

where we specify a Dirichlet process on $\mathcal{P}$, i.e.,

$$\mathcal{P} \sim \mathrm{DP}(\alpha \mathrm{Q}).$$

The use of this prior, due to Ferguson (1973), is a distribution over probability distributions, such that for any finite disjoint partition $\{A_i\}_{i>=1}^n$ on $\mathbb{R}^p$, $\mathcal{P}$ is a random measure such that the joint distribution $(\mathcal{P}(A_1), \ldots, \mathcal{P}(A_n)) \sim \mathrm{Dir}\left(\alpha Q(A_1), \ldots, \alpha Q(A_n)\right)$. The DP has two parameters: Q, the base measure, represents a prior guess at the distribution. α, the concentration parameter expresses the degree to which $\mathcal{P}$ will agree with Q on any set A. This follows from the definition given above and known properties of the Dirichlet distribution, i.e., $\mathrm{E}\left(\mathcal{P}(A)\right) = Q(A)$, and $\mathrm{V}\left(\mathcal{P}(A)\right) = \frac{Q(A)(1-Q(A)}{\alpha+1}$, showing that $\mathcal{P}(A) \xrightarrow{p} Q(A)$ as $\alpha \to \infty$ for any set A. By modeling $\mathcal{P}$ with a DP, one can be noninformative about the overall shape of $\mathcal{P}$, allowing for irregular shapes, multimodality and so forth. An argument can be made that by incorporating our uncertainty about these features of the distribution is required for coherent interpretation of the posterior distribution (Walker, 2010). For more information about the properties of the DP, see Ferguson (1973).

As shown by Sethuraman (1994), it follows from the definition of the DP that $\mathcal{P}$ is almost surely discrete and realizations of $\mathcal{P}$ can be produced by the following stick-breaking construction:

Let

$$\mathcal{P} = \sum_{k=1}^{\infty} \pi_k \delta_{(\tilde{\beta}_k^\top, \tilde{\sigma}_k^2)}, \tag{2.2}$$

where $\delta_{(.)}$ is the Dirac delta function. Note that although almost sure discreteness is a property applicable to "draws" from a DP, the posterior for $\mathcal{P}$ is in fact a mixture of DP (Antoniak, 1974). An implication of discreteness can be thought of as a "bet on sparsity"; that there are actually a finite (but unspecified) number of unique values that $(\beta_g^\top, \sigma_g^2)$ can take. The "atoms" distributed according to Q, specified by the product measure

$$\tilde{\beta}_k \overset{ind.}{\sim} \mathrm{N}(m_\beta, C_\beta), \quad \tilde{\sigma}_k^2 \overset{ind.}{\sim} \mathrm{IG}(a_{\sigma^2}, b_{\sigma^2}), \tag{2.3}$$

where "IG(a, b)" refers to the inverse gamma distribution which we parameterize by shape and scale parameters, a and b with density given by

$$p(x|a, b) = \frac{b^a}{\Gamma(a)} x^{a+1} e^{-b/x}.$$

The mixture weights, π_k, follow a stick-breaking process Sethuraman (1994). Under this reparameterization,

$$\nu_k = \frac{\pi_k}{1 - \sum_{l=1}^{k-1} \pi_l}, \tag{2.4}$$

where ν_k represents the weight for cluster k relative to the total probability remaining after $k - 1$ breaks. For the stick-breaking construction of the DP, the ν_k are modeled by:

$$\nu_k \overset{ind.}{\sim} \mathrm{Beta}(1, \alpha). \tag{2.5}$$

This assumption induces a stochastically decreasing ordering of the weights, so that $\sum_{k=1}^{K} \pi_k \xrightarrow{p} 1$ as $K \to \infty$. These facts suggest that the infinite mixture distribution, $\mathcal{P}$ can be well approximated by a finite mixture. We return to this idea in Section 2.4.

2.4 Model inference

We adopt a Bayesian approach for estimation and inference for the DP mixture model described in Section 3.3.2. This requires a specification the completion of a full probability model by specifying prior distributions. Looking at Figure 2.3, which shows the directed acyclic graphs representing the model, specifying a prior amounts to choosing values for model parameters which have no arrows pointing toward them, i.e. whose distribution is not otherwise specified in the model.

We choose values of m_β, C_β, a_{σ^2}, b_{σ^2} so that Q puts mass on reasonable values of the $p+1$ dimensional parameters $\left(\tilde{\beta}_g^\top, \tilde{\sigma}_g^2\right)$. First, estimates are computed for each gene independently by ordinary least squares. The hierarchical mean, m_β is chosen so the each component, $m_{\beta,j}$, is equal to the median of the estimates for $\beta_{g,j}$. The diagonal elements of the prior covariance matrix, C_{jj}, are chosen to be 4 times the sample variance of the $\beta_{g,j}$. The parameters of the inverse gamma are chosen by matching first and second moments, treating the estimates for σ_g^2 as data.

This can be thought of as an empirical Bayes approach. Alternatively, one could also select values based on prior experience. In any case, using informative or weakly informative priors for is important; if Q is chosen to be diffuse, that can be overly informative and lead to only few large π_k accounting for most of the total probability in the mixture, $\mathcal{P}$ (Gelman et al., 2014a).

The parameter α plays an important role in the model, since it helps to determine how quickly the successive elements of π decay, i.e. the number of clusters selected by the model. For computational convenience, we choose the conditionally conjugate prior

$$\alpha \sim \text{Ga}(a_\alpha, b_\alpha). \tag{2.6}$$

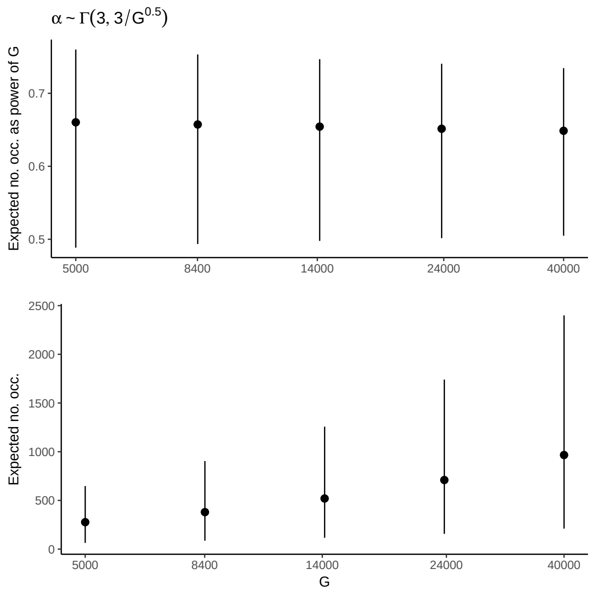


Figure 2.2: The number of clusters determined by the model is influenced by the parameter α, whose value determines the expected number of clusters prior to seeing the data. The figures above demonstrate the prior distribution of the prior expected number of clusters, $\mathrm{E}(K_{occ})$ for datasets with various number of response components, G.

This prior can be selected to express an a priori belief on the number of clusters, K_{occ}, in the data. Escobar (1994) give expression for the expected value of K_{occ} given α and G as,

$$\mathrm{E}(K_{occ}) = \sum_{g=1}^{G} \frac{\alpha}{\alpha + g - 1}. \tag{2.7}$$

In that paper, the authors use a table of values derived from this formula to defend their proposal for a prior over values of α ranging from G^{-1} to G^2, which admits values of $\mathrm{E}(K_{occ})$ anywhere from 1 to G. Due to computational limitations (we require $K_{occ} \ll G$), and our prior belief, based on scientific evidence, that there are more than just a few clusters, we choose $a_\alpha = 3$ and $b_\alpha = \frac{3}{G^{.5}}$ to express a prior that $\mathrm{E}(K_{occ})$ is probably between $G^{.5}$ and $G^{.75}$ (see figure 2.2).

2.4.1 Data augmentation and the TDP

Panels (a) and (b) in Figure 2.3 show two graphical representations of the model. As explained in Neal (2000), while sampling methods based on a) exist, it is more efficient to decouple the process partitioning the (β_g, σ_g) into clusters whose locations are realizations from the base measure, Q. This is done by introducing a latent variable, ζ_g, taking values on

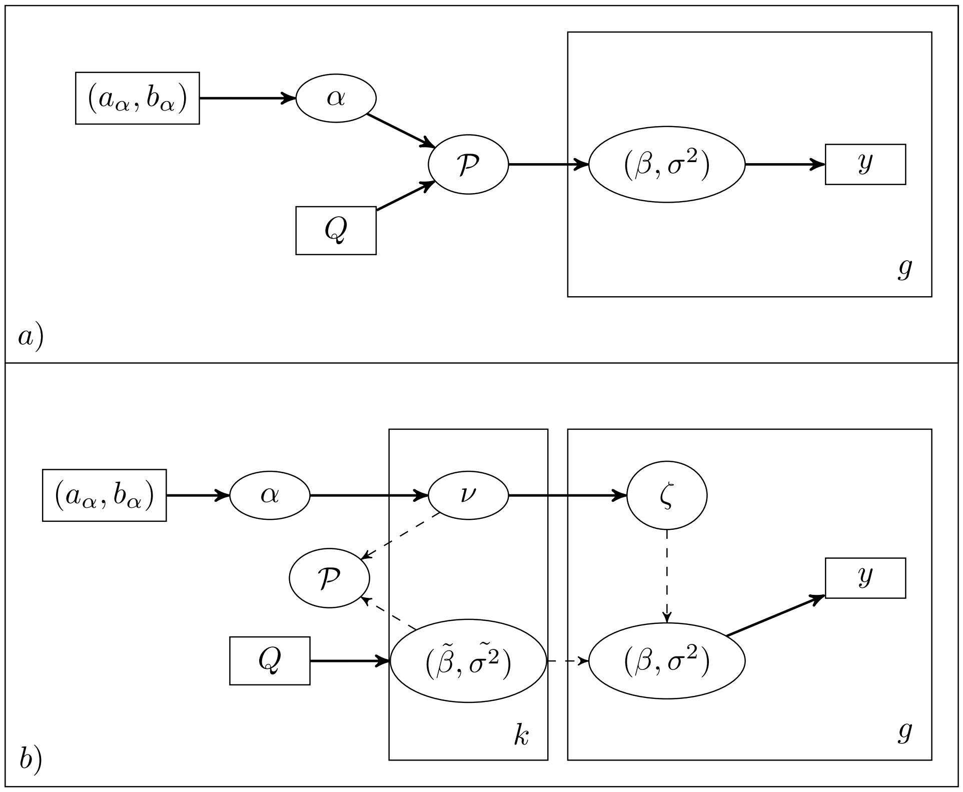


Figure 2.3: Directed acyclic graphs of the Bayesian nonparametric hierarchical regression model. Panel (b) introduces latent allocation parameters, ζ_g, which decouple the process which partitions the data into clusters from the distribution which provides the unique values, $(\tilde{\beta_k}, \tilde{\sigma}_k^2)$ of the cluster parameters. Solid lines indicate distributional dependency, dashed lines indicate deterministic functional relationships.

the positive integers with the discrete distribution given by $Pr(\zeta_g = k) = \pi_k$. These latent variables generate a random partition of the components into clusters, with $(\beta_g, \sigma_g^2) = (\tilde{\beta}_k, \tilde{\sigma^2}_k)$ for all g where $\zeta_g = k$. The expanded form of the model is shown in b).

Our Gibbs sampler is based on that proposed by Ishwaran and Zarepour (2000). In contrast to prior approaches to MCMC, detailed in Neal (2000), which can be classified as "marginal" methods, Ishwaran and Zarepour presented their blocked Gibbs sampler which allows approximate inference for DPM models. Unlike the marginal Gibbs samplers, which update the cluster allocation for each component, conditional on all other allocations, the blocked sampler jointly updates all cluster allocations independently. Being able to do so is advantageous when G is large, since it becomes possible to do a large portion of the necessary computation using many parallel processes which can be executed concurrently. As we can see from Figure 2.3, ζ_g are conditionally independent given $\mathcal{P}$. This is problematic since $\mathcal{P} = \sum_{k=1}^{\infty} \pi_k \delta_{(\beta_k, \sigma_k^2)}$ is an infinite mixture. Ishwaran and James (2001) showed that, due to the stochastic ordering of the π_k, $\mathcal{P}$ can be well-approximated by $\mathcal{P}_K = \sum_{k=1}^{K} \pi_k^* \delta_{(\beta_k, \sigma_k^2)}$, letting $\pi_k^* = \pi_k$ for $k < K$ and $\pi_K^* = 1 - \sum_{k=1}^{K-1} \pi_k$. The authors provide an approximate lower bound to the L_1 distance between $\mathcal{P}_K$ and $\mathcal{P}$ as a function of G, K and α and argue that because the bound decreases exponentially with K, it should be possible in practice to use a hierarchical model based on $\mathcal{P}_K$ that is virtually indistinguishable from one based on $\mathcal{P}$.

Applying this truncation in our model is achieved by setting $\nu_K = 1$. We then have

$$\zeta_g \sim \text{Cat}_K(\pi) = \sum_{k=1}^{K} \pi_k \delta_k, \text{ i.e. } Pr(\zeta_g = k) = \pi_k.$$

2.4.2 Posterior distribution

Applying Bayes' theorem, the posterior distribution, up to a constant, is

$$p(\beta, \sigma^2, \mathcal{P}, \alpha | y) = p(\zeta, \tilde{\beta}, \tilde{\sigma^2}, \nu, \alpha | y) \propto p(y | \tilde{\beta}, \tilde{\sigma^2}, \zeta) \times p(\zeta | \nu) \times p(\nu | \alpha) \times p(\tilde{\beta}, \tilde{\sigma^2})$$

$$= \prod_{g=1}^{G} \left[\text{N}(y_g;\, \beta_{\zeta_g}, \sigma_{\zeta_g}^2) \times \text{Cat}_K(\zeta_g;\, \pi(\nu)) \right] \times$$

$$\prod_{k=1}^{K} \left[\text{Be}(\nu_k;\, 1, \alpha) \times \text{N}(\tilde{\beta}_k;\, m_\beta, C_\beta) \times \text{IG}(\tilde{\sigma}_k^2;\, a_{\sigma^2}, b_{\sigma^2}) \right] \times \text{Ga}(\alpha; a_\alpha, b_\alpha)$$

The first equality follows from the invariance to reparameterization of the posterior distribution; that is, while we are interested in the posterior distribution of the β_g and σ_g (diagram a) in Figure 2.3), we lose nothing by formulating the problem in terms of ζ_g, $\tilde{\beta}_k$ and $\tilde{\sigma}_k$ (diagram b)). The last equality follows from the product rule of conditionally independent random variables.

2.4.3 Full conditionals

Our blocked Gibbs sampler is constructed as follows:

Step 1: Both the allocation parameters, ζ_g and the DP concentration parameter, α, are *mutually conditionally independent* given π, β, σ^2 and the full conditional distribution is given by

$$p(\zeta, \alpha|\cdot) \propto \prod_{g=1}^{G} \left[\mathrm{N}(y_g;\, \beta_{\zeta_g}, \sigma^2_{\zeta_g} W_g)\ \mathrm{Cat}_K(\zeta_g;\, \pi(\nu))\right] \prod_{k=1}^{K} \left[\mathrm{Be}(\nu_k;\, 1, \alpha)\ \mathrm{Ga}(\alpha; a_\alpha, b_\alpha)\right], \tag{2.8}$$

with conditional independence being implied by the product rule. For ζ we get

$$p(\zeta|\cdot) = \mathrm{Cat}_K(\hat{\pi}_k),\ \text{with} \tag{2.9}$$

$$\hat{\pi}_k \propto \pi_k\, \mathrm{N}(y_g; X\tilde{\beta}_k, \tilde{\sigma}^2_k W_g).$$

We see that α depends only on ν:

$$p(\alpha|\cdot) \propto \prod_{k=1}^{K-1} \mathrm{Be}(\nu_k; 1, \alpha)\ \mathrm{Ga}(\alpha; a_\alpha, b_\alpha) \propto \alpha^{(K-1)+a_\alpha-1} \left(\prod_{k=1}^{K-1}(1-\nu_k)\right)^{\alpha} e^{-b_\alpha \alpha} \tag{2.10}$$

$$= \alpha^{(K-1)+a_\alpha-1} e^{-(-\log \pi_K + b_\alpha)\alpha}$$

$$\implies p(\alpha|\cdot) = \mathrm{Ga}(K + a_\alpha - 1, -\log \pi_K + b_\alpha)$$

The full conditionals for $\tilde{\beta}_k$ and $\tilde{\sigma}^2_k$ are straightforward, due to conjugacy. However, due to Equation 2.3, the full conditional for $\tilde{\beta}_k$ depends on $\tilde{\sigma}^2_k$, and visa versa. An alternative specification for the distribution of $\tilde{\beta}_k$, $\tilde{\beta}_k \overset{ind.}{\sim} \mathrm{N}\left(m_\beta, \tilde{\sigma}^2_k \mathrm{diag}(c_1, \ldots, c_p)\right)$ would allow for joint updates of $(\tilde{\beta}_k, \tilde{\sigma}_k)$ with conjugacy. We opted for our approach, finding it more intuitive to specify C_β than $\mathrm{diag}(c_1, \ldots, c_p)$.

Step 2: The full conditional for cluster location, $\tilde{\beta}_k$:

$$p(\tilde{\beta}_k|\cdot) \propto \prod_{g:\zeta_g=k} \mathrm{N}(y_g;\, X\tilde{\beta}_k, \sigma_g^2 W_g^{-1})\ \mathrm{N}(\tilde{\beta}_k; m, C) \tag{2.11}$$

$$\implies p(\tilde{\beta}_k|\cdot) = \mathrm{N}(\tilde{\beta}_k; \hat{m}_k,\, \hat{C}_k),$$

where $\hat{C}_k = \left(\sigma_g^{-2} \sum_{g:\zeta_g=k} X^\top W_g X + C^{-1}\right)^{-1}$, and $\hat{m}_k = \hat{C}_k \left(\sum_{g:\zeta_g=k} X^\top W_g y_g + C^{-1} m\right)$.

Step 3: The full conditional for cluster variance, $\tilde{\sigma}_k^2$, is:

$$p(\tilde{\sigma}_k^2|\cdot) \propto \prod_{g:\zeta_g=k} \mathrm{N}(y_g;\, X\tilde{\beta}_k, \sigma_g^2 W_g^{-1})\, \mathrm{IG}(\tilde{\sigma}_k^2; a_{\sigma^2}, b_{\sigma^2}) \tag{2.12}$$

$$\implies p(\tilde{\sigma}_k^2|\cdot) = \mathrm{IG}(\hat{a}_k, \hat{b}_k),$$

where $\hat{a}_k = a_{\sigma^2} + \frac{NM_k}{2}$, and $\hat{b}_k = b_{\sigma^2} + \sum_{g:\zeta_g=k} y_g^\top W_g y_g - 2\beta_g^\top X^\top W_g y_g + \beta_g^\top X^\top W_g X \beta_g$

Here M_k is the number of g for which $\zeta_g = k$.

Step 4: The full conditional for ν_k, $k = 1, \ldots, K-1$ depends only on $M_k(\zeta)$. The full conditional is:

$$p(\nu_k) \propto \prod_{l \geq k} \prod_{g:\zeta_g=k} \pi_k\ \mathrm{Be}(\nu_k; 1, \alpha) \propto \nu_k^{M_k+1}(1-\nu_k)^{\sum_{l>k} M_l + \alpha} \tag{2.13}$$

$$\implies p(\nu_k) = \mathrm{Be}(M_k + 1, \sum_{l>k} M_l + \alpha)$$

2.5 Computation on the GPU

2.5.1 General remarks

Modern GPUs offer hundreds to thousands of cores and are capable of handling over a million concurrent processes. Compared with multi-core CPUs, which typically boast 16 or fewer processors, this suggests a large benefit to using GPUs to exploit parallelism. We now direct attention to some differences in implementation, compared to traditional CPU programming, that are required when porting code to a GPU platform.

GPUs are usually subservient to a host CPU: the executing program is run on a CPU which turns over control periodically to the GPU to run specific tasks. These tasks, or “kernels”, follow the single instruction, multiple data (SIMD) paradigm. Each core on the GPU is assigned to work on a specific chunk of memory, but all cores execute the same program. Each instance of the program is called a “thread”. It is desirable to avoid branching logic in kernels, in part because the GPU cores are relatively slow, so branching can easily lead to high latency. The best results are obtained by having all threads proceed in lockstep.

The GPU has its own memory system. Because copying memory from host (CPU) to device (GPU) and visa versa is relatively slow, care should be taken to excessive copying from host to device and back. Ideally, necessary inputs are copied to device memory once, and any large output objects produced by the device are kept in device memory until they are complete, at which point they are copied back to the host all at once.

When programming for the CPU, memory accesses tend to be fast. On the GPU, while the thread-local (“shared”) memory is fast, reading and writing from global memory is slow. Therefore, when writing kernels, one should avoid both unnecessary reading and writing from global device memory. This can often be accomplished by copying values needed more than once to variables or “shared” arrays, for fast access. However, thread-local memory is quite limited. A good rule of thumb is to write kernels to be relatively simple, so that the memory requirements are low.

Threads are themselves organized into “warps”. When data is read from global memory, it reads not one address at a time, but in chunks to minimize overhead costs. To take advantage of this, consecutively indexed threads should use data from memory at consecutive addresses. When this happens, the reads are “coalesced”. If consecutive threads access addresses that are distant to one another, reads are not coalesced, which will make memory transfer inefficient, hence the program will tend to be slow.

For more specifics on GPU computing for statistics, with an emphasis on mixture models, see Suchard et al. (2010). In the following subsections, we use some examples to demonstrate some instances where parallelism can help to accelerate routine computation.

2.5.2 Routines

2.5.2.1 Reductions

On the GPU, individual processors are slow, and asynchronous tasks can result in many idle threads. In order to achieve speed-ups, algorithms must exploit parallelism and do so in a way that respects the limitations of the hardware. A basic example, which we use heavily, is reduction. Given some data in memory, $a_1, a_2, \ldots, a_n$, and an *associative* binary operator $+$, the problem is to compute $s_{1:n} = \sum_{i=1}^{n} a_i$. The parallelized reduction algorithm is given below in Algorithm 2.5.1, and is illustrated in the top half of the diagram shown in Figure 2.4.

Algorithm 2.5.1: PARALLEL-REDUCE(a)

```
comment: Parallel reduction of $n = 2^D$ elements. Upon completion, $a[2^D] = s_{1:n}$
for $d \leftarrow 1$ to $D$
    do for $i \leftarrow 1$ to $n$ in parallel
        do if $i \bmod 2^d = 0$
            do $a[i] \leftarrow a[i - 2^{d-1}] + a[i]$;
```

To see why Parallel-Reduce works, let a be an array of size $n = 2^D$, $a = (a_1, \ldots, a_n)$. At the end of iteration d of the outer loop, the memory in the i position, where $i \mod 2^d = 0$, contains the sum, $s_{(i-2^{d-1}+1):i}$, which reduces problem of computing $s_{1:n}$ to $\sum_{j=1}^{2^{D-d}} (s_{((j-1)2^d+1):(j \cdot 2^d)})$. In the last iteration, the target is computed simply by adding the partial results in $a_{2^{D-1}}$ and a_{2^D}. Upon completion of the routine, modified positions in memory contain partial sums of the original values. These "side effects" of algorithm turn out to be useful for another purpose, as we will see shortly.

2.5.2.2 Parallel scans

Blelloch (1990) described a general algorithm for parallel scans. These include parallel cumulative sums as a special case. Parallel scans can be divided into two general types: exclusive scans produce $s = (0, s_{1:1}, s_{1:2}, \ldots, s_{1:(n-1)})$ and inclusive scans produce $s = (s_{1:1}, s_{1:2}, \ldots, s_{1:n})$.

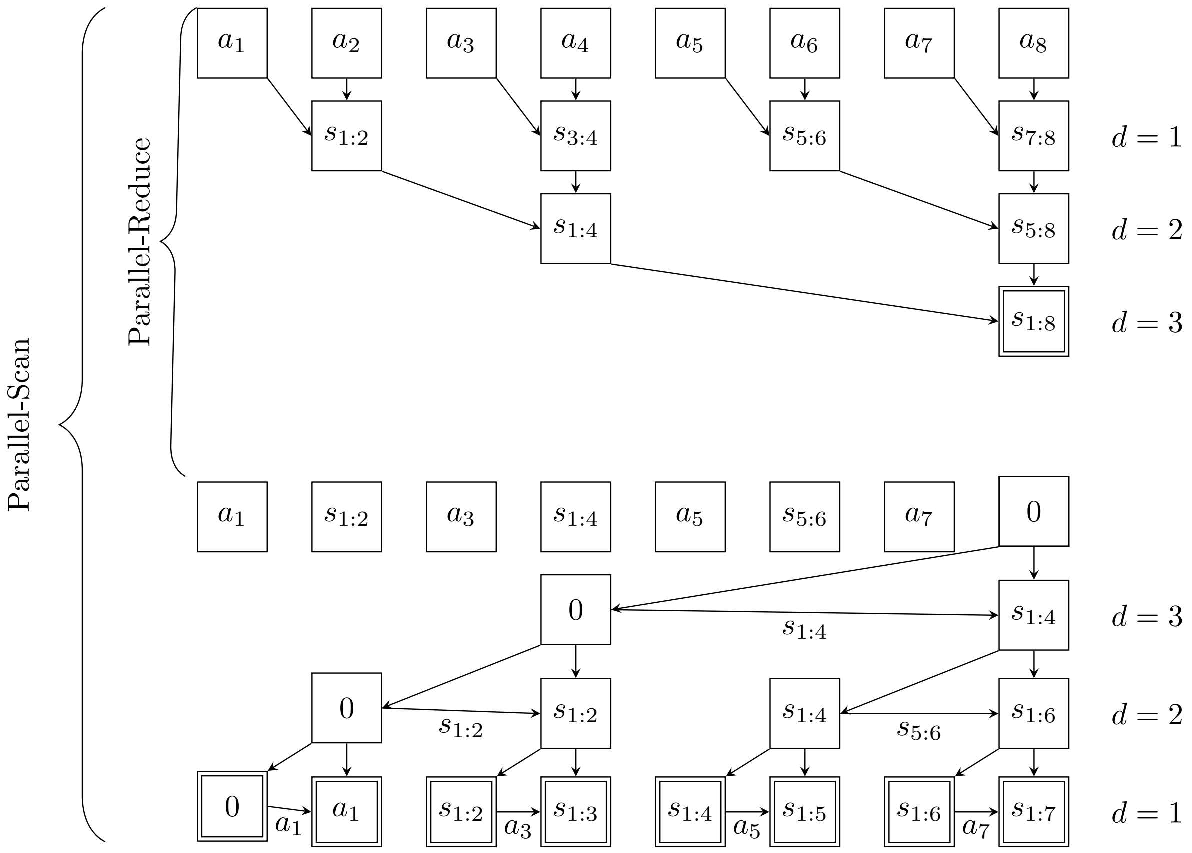

Parallel-Scan
Parallel-Reduce
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8
$s_{1:2}$ $s_{3:4}$ $s_{5:6}$ $s_{7:8}$ $d = 1$
$s_{1:4}$ $s_{5:8}$ $d = 2$
$s_{1:8}$ $d = 3$
a_1 $s_{1:2}$ a_3 $s_{1:4}$ a_5 $s_{5:6}$ a_7 0
0 $s_{1:4}$ $s_{1:4}$ $d = 3$
0 $s_{1:2}$ $s_{1:2}$ $s_{1:4}$ $s_{5:6}$ $s_{1:6}$ $d = 2$
0 a_1 a_1 $s_{1:2}$ a_3 $s_{1:3}$ $s_{1:4}$ a_5 $s_{1:5}$ $s_{1:6}$ a_7 $s_{1:7}$ $d = 1$

Figure 2.4: Diagram illustrating Parallel-Reduce and Parallel-Scan for an array of length $2^3 = 8$. Values in each row indicate the value after completing a step in the outer loop indicated by "$d = x$". $s_{i:j}$ denotes a sum of the original elements, $s_{i:j} = \sum_{k=i}^{j} a_k$. Arrows pointing down indicate that the previous value is carried forward from the previous step. Arrows pointing right indicate that the value in the left position increments the value previously contained in the right position. Arrows pointing left indicate that the value is copied from right to left. Parallel-Reduce results in the sum of all the original elements being contained in the right-most position. Parallel-Scan uses the partial sums produced as a side-effect of Parallel-Reduce to produce the result that each position holds the sum of the original elements to the left.

Just as with Parallel-Reduce, '+' can be replaced by any associative binary operator. Clearly, an exclusive scan can be generated from an exclusive scan by dropping the first element and appending $s_{1:n}$, and similarly an exclusive scan can be generated from an inclusive scan by dropping the last element and prepending a zero. While readers are likely more familiar with the inclusive scan, the parallel algorithm we discuss here is for an exclusive scan.

For simplicity, we describe the case for an array with 2^d elements and 2^{d-1} processors. The algorithm is composed of two steps: Parallel-Reduce (referred to in Blelloch (1990) as"upsweep"; Algorithm 2.5.1), which is followed by a second stage (which Blelloch refers to as "downsweep"). Given an input array a, an Parallel-Reduce modifies the array a, defined above, so that the memory in position i contains $s_{1:(i-1)} = \sum_{j=0}^{i-1} a_j$. By convention, $a_0 = 0$, i.e. the identity element. Pseudocode is given in Algorithm 2.5.2.

Algorithm 2.5.2: PARALLEL-SCAN(a)

```
comment: Parallel cumulative (prefix) sum of $n = 2^D$ elements.
$a \leftarrow$ REDUCE($a$)
$a[n] \leftarrow 0$
for $d \leftarrow D$ to 1
  do
    for $i \leftarrow 1$ to $n$ in parallel
      do
        if $i \bmod 2^d = 0$
          do
            $tmp \leftarrow a[j - 2^{d-1}]$;
            $a[i - 2^{d-1}] \leftarrow a[i]$;
            $a[i] \leftarrow tmp \oplus a[i]$;
return ($a$)
```

The second stage of Parallel-Scan works with the partial results in left in memory after Parallel-Reduce. First, note that a maximum of $\lfloor \log_2(i) \rfloor$ partial sums are required for computing $s_{1:(i-1)}$, thus the number of steps in the second stage is the minimum number required. After setting the right-most position to zero, the second stage of Parallel-Scan works on the same sets of pairs of positions as Parallel-Reduce, only in reverse order. Within each pair, the value in the left position is stored in a temporary buffer, then the copy from the right position is copied to the left position. Last, the value in the right position is incremented by the value in the buffer. Intuitively, this works because the accumulated value from previous steps is a sum of values to the left of the current left position while the sum in the current left position is required by to compute the right position's target value. Blelloch (1990) provides a proof by induction of the correctness of the result.

In practice, there are physical constraints that require modifications to Parallel-Scan. For example, because the size of a may not be a power of 2 and also because processors are typically much less than the size of a, the algorithm requires modification. One solution is to break the work into chunks, each with a size that is a power of 2. Each chunk is scanned separately by a single processor (in serial) and the scan totals written to a buffer. Next, the scan totals are themselves scanned in parallel. These scanned totals can be mapped by to the chunks, and are used as offsets for the values obtained from the original scan.

Both Parallel-Reduce and Parallel-Scan presented above require $O(\log_2(n))$ steps compared to the $O(n)$ required for a serial implementation. The actual speed-up attained depends on hardware and implementation. Harris et al. (2007) implemented Parallel-Scan for GPUs based on the implementation in Blelloch (1990) and observed maximum speedups of 5 times over an optimized serial CPU implementation. The advantage of the GPU implementation increased with n and then leveled off, achieving its best performance on arrays with over 1 million elements. Sengupta et al. (2008) implemented a variant of Parallel-Scan that is less efficient in that it requires a larger number of operations, but reduces the number of steps from $2\log_2(n)$ to $\log_2(n)$. They reported speedups of nearly $2\times$ compared to similar implementations based on Parallel-Scan. More recently, Ha and Han (2013) offered an implementation based on an algorithm that is a hybrid of the two mentioned. They reported speedups of about $1.5\times$ when compared to the implementation of Sengupta et al. (2008).

Parallel scans are used in various ways in our MCMC algorithm. For example, the calculation of cumulative probabilities to sample from categorical distributions in step 1 computes the cumulative sum – on the log scale for numerical stability – using $\log(e^{v_1} + e^{v_2})$ as the binary operator.

2.5.3 Gibbs sampler revisited: exploiting parallelism

Owing to conjugacy, the full conditional distributions of our Gibbs sampler are known distributions and are easy to simulate from. By simulating a drawing from each full conditional per iteration of the Markov chain, the sequence of draws converge to a set of draws from the posterior distribution. There are opportunities to exploit parallelism within some of these Gibbs steps because of conditional independence among blocks of parameters.

Insofar as the computation for these full conditionals depends on the data, it is only through summary statistics, which can be partially pre-computed (at the component level). Therefore, $y_g^\top W_g y_g$, $y_g^\top W_g X$, and $X^\top W_g X$ are computed once, prior to sampling, and saved in device memory. Also, in order to coalesce memory accesses for different steps, $\{y_g^\top W_g X\}_{1:G}$ is stored in two ways; row-wise and column-wise.

Another strategy that we use in an effort to coalesce memory accesses is to adhere to the structure of arrays (”SoA”) approach (Hoberock and Bell, 2008). In order to do so, it is necessary to organize the inputs and outputs for a kernel computing $f(A_i, B_i) = C_i$ so that A, B and C can be accessed like a “zipped” set of ranges with a common index, i.e. $(A, B, C)_i = (A_i, B_i, C_i)$. Because of large number of operations required, we wanted to avoid sorting and/or copying to achieve the organization required for SoA. The solution was to use fancy iterators which enable arbitrary user-specified access patterns. If, for example, we wanted to calculate $f(A_i, B_j) = C_{ij}$ for $i = 1, \dots, I$, $j = 1, \dots, J$, we can recast the problem as $f(\tilde{A}_k, \tilde{B}_k) = \tilde{C}_k$ so that $\tilde{A}_k = A_{\text{floor}(k/J)+1}$, $\tilde{B}_k = B_{(k+1) \mod I+1}$ and $\tilde{C}$ is a 1-d array with IJ elements where $\tilde{A}, \tilde{B}$ are just virtual arrays which actually just provide a map to the memory stored in A and B.

Allocation parameters The computational complexity for this step is $O(GKp^2)$. Fortunately, computation of the unnormalized weights is an embarassingly parallel problem. For numerical stability, these weights are computed on the log scale. From (2.9),

$$\log Pr(\zeta_g = k|\cdot) = constant + \log \pi_k - N \log \sigma_k - \frac{1}{2\sigma_k^2}\left(y_g^\top W_g y_g - 2 y_g^\top W_g X \beta_k + \beta^\top X^\top W_g X \beta\right). \tag{2.14}$$

In our implementation, first, separate kernels compute a) $y_g^\top W_g X \beta_k$ and b) $\beta^\top X^\top W_g X \beta$. a) reduces to a matrix multiplcation problem, for which optimized software has been developed for the GPU. To see this, let $B^\top D$, where $\tilde{\beta}_k$ form the columns of B, and $X^\top W_g y_g$ form the columns of D. b) is accomplished using the SoA approach. Once a) and b) have been computed and stored in global memory, (2.14) is computed, again using an SoA approach.

To perform the categorical sampling of ζ, we first perform a cumulative sum/scan of the weights (using log-sum-exp for numerical stability) for each component. Define $S_{g,k}$ to be the k^{th} partial log sum for component g. Next, we draw $logU_g$, $U_g \sim \mathrm{U}(0,1)$ add add the respective total log sum, $S_{g,K}$, obtaining $V_g = \log U_g + S_{g,K}$. Next, we have a kernel evaluate the comparison, $S_{g,k} < V_g$, returning 1 if true and 0 if false. The resulting $G \times K$ array is then reduced over each value of g, the result updating ζ.

Cluster atoms A prerequisite to drawing from the full conditional distribution for the cluster atoms is to compute cluster summary statistics. Although this seems fairly straightforward, consisting of a series of reductions, there is an additional complication in that the values to reduce are not contiguous, as the groups being determined by the current value of ζ. Rather than sorting (and copying) the data for each summary, we sort ζ storing both the sorted vector as well as the permutation that produced it. Using the stored permutation as a map for the summands, such as $y_g^\top W_g y_g$, we compute the corresponding summaries, $\sum_{g:\zeta_g=k} y_g^\top W_g y_g$, as if they were sorted using a specialized reduce algorithm, which reduces elements with a common key, the key being given by (sorted) ζ. Note that this produces cluster summaries only for occupied clusters. Since the conditional distribution for cluster k depends on "updated" prior parameters that represent a combination of the data and the priors, it is convenient to fill a K-dimensional parameter vector, do the required computation for the occupied clusters, and then update the K-dimensional vector so that draws can be performed for all clusters in parallel.

Cluster regression coefficients Computation of $\hat{C}_k^{-1}$, the precision matrix, is calculated (using AoS), followed by Cholesky decomposition. For the sizes of p we consider ($p < 8$), local memory limitations do not seem to be a problem, so we simply instruct each thread to decompose one of the K matrices. Next, $\hat{m}_k$ is computed by solving $\hat{C}_k^{-1}\hat{m}_k = \hat{C}_k^{-\frac{1}{2}}\left(\hat{C}_k^{-\frac{1}{2}}\hat{m}_k\right) = X^\top W_g y_g$, twice using a routine for solving triangular systems of equations. The full conditional draw for $\tilde{\beta}_k$ is accomplished by first generating multivariate standard normal draws, Z_k, and setting $\tilde{\beta}_k = \hat{C}_k^{-\frac{1}{2}} Z_k + \hat{\beta}_k$. The scaling operation is done by solving a system of equations as before. These computations are parallelized across clusters, $k = 1, \dots, K$.

Cluster variance For $\tilde{\sigma}_k^2$, the main hurdle is the computation of $\hat{b}_k$. This is done in three steps, each parallel over k, using the SoA approach. First, we compute the dot products, $\beta_k^\top \sum_{g:\zeta_g=k} X^\top W_g y_g$, followed by the quadratic forms, $\beta_k^\top \left(\sum_{g:\zeta_g=k} X^\top W_g X\right) \beta_k$, and, lastly, parallel summation of these with $\sum_{g:\zeta_g=k} y_g^\top y_g$.

Cluster weights The conjugate beta draws for ν_k each require two parameters. The first shape parameters are computed by parallel elementwise addition; the second can be computed using a parallel scan, in reverse order, of M_k.

Mass parameter The scalar parameter α depends only on the constants K, a_α and b_α, and the scalar quantity π_K. This step can be conveniently performed by the CPU.

2.5.4 Initialization

Papaspiliopoulos and Roberts (2008) identified an issue with conditional samplers for DP mixture models that rely on the stick-breaking construction. They noted that, although the cluster weights are stochastically ordered, the ordering of cluster labels is only weakly identifiable, so that the posterior distributions for the weight at any given index is multi-modal. This is due to many configurations of the allocation parameters being nearly equivalent in the posterior. We suspect that, because of this weakness, poor initialization can lead to very slow convergence. We initialize our chains as follows. Using the ranges of independent estimates of the component-specific parameters, we set a regular $(p + 1)$-dimensional grid. Using the base measure, we sort the grid values by their prior densities in descending order, retaining the first K. We set $\pi_k = 1/K$ for all $k = 1, \dots, K$ and α to 1. Next we run the sampler for several thousand iterations. The initialization is taken to be the last iteration of the first chain, but with the indices reordered in descending order by π. The purpose for reordering is that, in the first chain, it is common for the last cluster to become occupied early and never become unoccupied. This is undesirable, as it undermines the quality of the truncation approximation. Reordering the indices after most of the clusters have become established seems to correct the problem (provided that K is selected to be large enough.)

2.5.5 Output

Because of the dimensionality of the problem is large, it is cumbersome to save all of the MCMC samples. Following the approach in Landau and Niemi (2016), we preselect a small number of parameters for which we do save each iteration, but for the rest we first determine which functions of those parameters are of interest and use online algorithms that use running sums to update our estimates of those functions. The class of estimates we calculate fall into two categories,

1. expectations of the component-specific parameters and their squares, i.e. $\mathrm{E}\,\beta_g$, $\mathrm{E}\,\beta_g^2$, $\mathrm{E}\,\sigma_g$, $\mathrm{E}\,\sigma_g^2$ and
2. component-specific posterior probabilities of conjunctions of linear inequalities of the elements of β_g, i.e., $\Pr\left(c_1\beta_g > 0 \wedge \ldots \wedge c_t\beta_g > 0\right)$.

Updates of these quantities are embarrassingly parallel across g, so are well suited to the GPU. The update in 1) is done elementwise, based on a running sum, and the update in 2) can be decomposed into four tasks, a matrix multiplication to compute $C\beta$, an elementwise evaluation of the sign of the value, a parallel set-wise reduction using the minimum to evaluate each conjunction, and finally an update of the estimator (using the same functionality as 1)).

We also save and return several low-dimensional parameters which provide some information on the overall behavior of the sampler: the number of occupied clusters, the maximum id of the occupied clusters and α, the concentration parameter. We also optionally return samples of selected component-specific parameters and a user-specified number of draws of $\mathcal{P}$.

2.6 Simulation Study

To assess the time per iteration and time to convergence, we simulated data and sampled from the posterior using our algorithm. Seconds per iteration was calculated by dividing the total time during the sampling phase (after initialization and burn-in) by the number of sampling iterations to obtain one measure of this variable for each simulated data set. The time required to obtain a sufficient number of samples depends not only on the time per iteration, but also the number of MCMC samples required to obtain an "effective sample" after correcting for the autocorrelation of the chain. Therefore, we also considered seconds per effective sample, where the number of effective samples was calculated in R using the 'coda' package Plummer et al. (2010).

We identified several factors that could potentially impact the running time: the dimension of the regression coefficients, p, the number of groups, G, and truncation limit, K. The sample size was chosen to be proportional to the p, $N = 4p$. Two simulated data sets were generated and analyzed for each combination of $p = 2, 4, 6$, $G = 2^12, 2^13, 2^14$ and $K = 2^12, 2^11, 2^12$.

For the simulations, we first generated a distribution P (as in Eq. 2.2). We did this by first producing $K_t = \lfloor\sqrt{G}\rfloor$ cluster components, $\tilde{\beta}_k$, which were generated independently from a multivariate normal distribution with diagonal covariance C. We chose to have the variance decrease with the dimension, $C_{jj} = 3/j^2$. This decision was somewhat arbitrary, but is consistent with empirical observation that the proportion of significant effects is often inversely related to the dimension of the design matrix in regression. The cluster variances, $\tilde{\sigma^2}_k$ were generated independently from a gamma distribution. The component-specific coefficients, β_g and variances, σ^2_g, were chosen by drawing an index k uniformly with replacement from the set $\{1, ..., K_t\}$. Finally, conditional on (β_g, σ^2_g), log-expression data were generated from the normal distribution as in Equation 2.1.

The prior distributions used to analyze the simulated data were set to the true values for $m_\beta = m$, and $C_\beta = C$. The inverse gamma parameters a^2_σ and b^2_σ were both set to 1. Note that $\tilde{\sigma}^2_k$ come from a gamma distribution, rather than an inverse gamma, to avoid very large values, as the inverse gamma has a much heavier tail. The prior for α is the one described in section

3.3.2, with the prior expectation close or equal to K_t.depend on the independent estimates of the component-specific parameters and the number of components as described in Section 2.4.

For simplicity, we chose to focus on the effective number of samples for α because it depends on π_K, a parameter which is sensitive to changes in the number of occupied clusters and the maximum occupied cluster index. Thus, we expect that the efficiency of sampling for alpha is a reasonable proxy for the overall efficiency of the sampler.

Cluster parameters Let $K_t = \text{floor}(G^{0.5})$. For $k = 1 \ldots K_t$, draw $\tilde{\beta}_k \sim \text{Normal}(0, C)$, $\tilde{\sigma^2}_k \sim \text{Gamma}(1, 1)$, where $C = \text{diag}(3, 3/2^2, \ldots, 3/p^2)$.

Allocation to clusters For $g = 1, \ldots G$, draw $\zeta_g \overset{ind.}{\sim} \text{Discrete-Uniform}(1, K_t)$.

Simulated data Let $X = \begin{pmatrix} 1_{4\times 1} & 0_{1\times(p-1)} \\ 1_{4(p-1)\times 1} & I_{(p-1)\times(p-1)} \otimes 1_{4\times 1} \end{pmatrix}$, where $\otimes$ is the Kronecker product. Draw $y_g \sim \text{Normal}(X\tilde{\beta}_{\zeta_g}, \tilde{\sigma}^2_{\zeta_g})$, where $y_g = (y_{g1}, \ldots, y_{gN})^\top$.

The left panel of Figure 2.5 shows the average time per iteration at each simulation setting. Note that both axes are on the log scale. As expected, there is an increase in cost associated with larger data sets. We see a substantially larger increase in running time for doubling the truncation limit, K, than for doubling the number of groups, G; this is likely because more of the computational steps depend on K than depend on G. The cost associated with increasing the dimension of the regression, p, appears to be relatively small. Increases in p are associated primarily with a larger amount of work required for individual threads, while increases in G and K incur a cost of many additional threads.

While the trends in time per iteration are fairly obvious, the results for time per effective sample are less so. This is expected, since the number of effective samples is an estimate, introducing another source of variability. The right panel of Figure 2.5 shows the average time per effective sample at simulation setting. While some trends reflect those seen in the raw time per iteration, there are some notable differences. The most important factor is still K, but p has a much larger impact. Also, contrary to the previous figure, increases in p are associated with a reduction in time required.

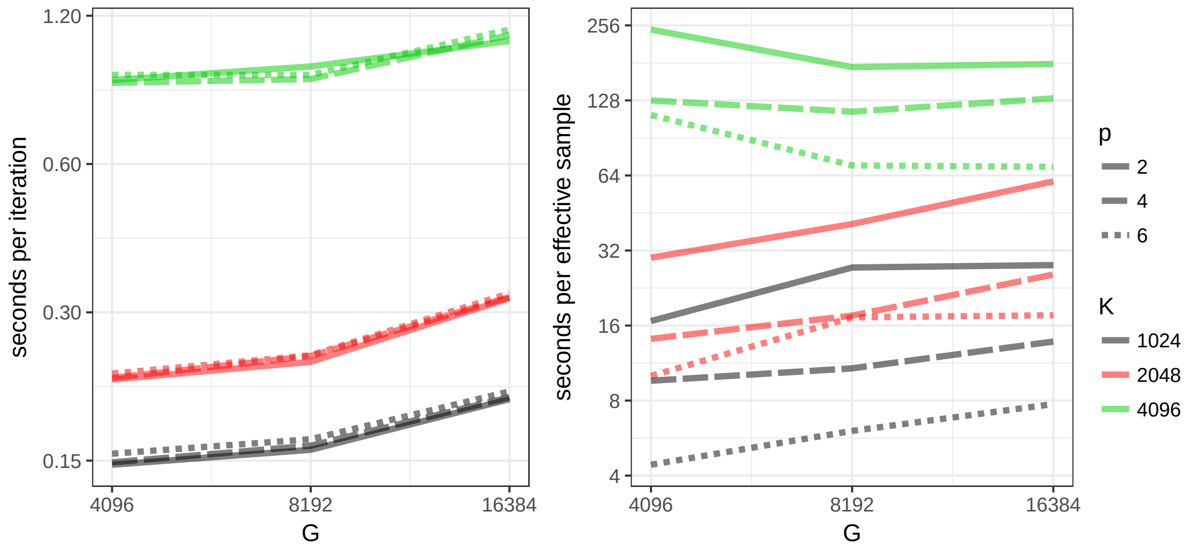


Figure 2.5: Left: Average seconds required per MCMC sample across for two simulated datasets at each level of K, the truncation limit, G, then number of components, and p , the dimension of each regression coefficient, β_g. K has the largest impact on computation time, followed by G. The effect of p is small over the range considered. Right: Average seconds per effective sample across for two simulated datasets at each level of K, the truncation limit, G, then number of genes, and p , the dimension of regression coefficient. Interestingly, the efficiency in sampling for α increases with p in our simulations. This might be explained by an increased separation between clusters, leading to greater stability in the partition defined by the allocation parameters.

A possible explanation for this relates to our simulation setup: the true number of clusters grows with G, but not with p. Also, the total number of samples does increase with p. By holding the number of clusters fixed, while moving the clusters apart (by increasing the dimension of the parameter space), one expects that the partition of groups into clusters becomes more precise. Since we consider the effective sample size for α to be a proxy for the stability of the random partition separating groups into clusters, it follows then that increases in p should increase the stability of both the random partition, as well as the parameter α.

Figure 2.6 shows normalized trace plots for α. These show shorter and less extreme excursions at higher values of p, which is consistent with this hypothesis.

Table 2.2 shows the result of fitting a linear regression of $\log_2$(seconds/effective sample on $\log_2(K)$, $\log_2(G)$ and $\log_2(p)$. The estimated coefficients of this model suggest that seconds per effective sample grows sub linearly in G, approximately quadratic in K and inversely with V.

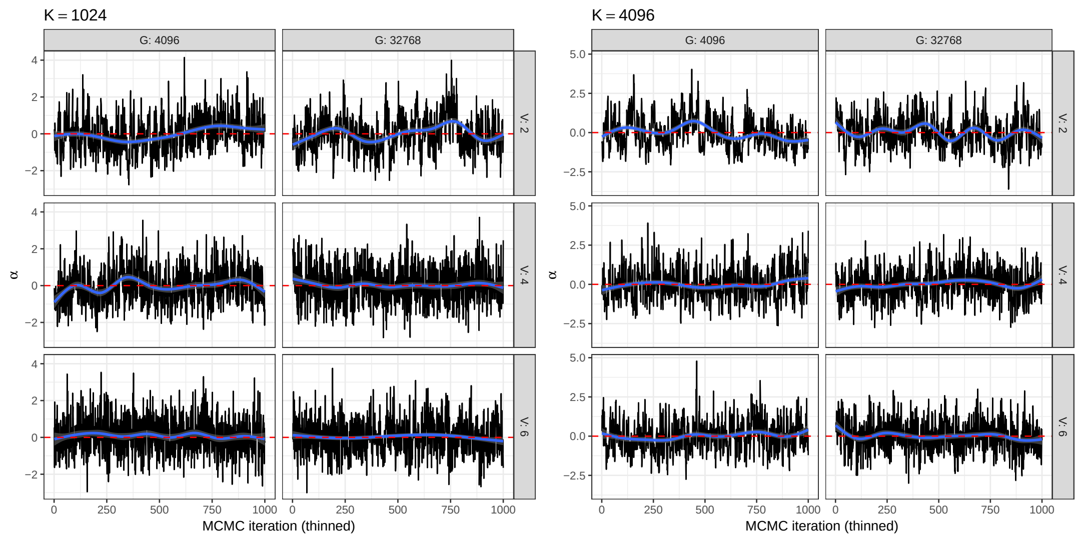


Figure 2.6: Centered and scaled trace plots for α across levels of p for various settings of G and K.

Table 2.2: Estimated exponentiatedcoefficients for the linear regression model: $\log_2$(sec. per eff. sample)$= \beta_0 + \log_2(K)\beta_1 + \log_2(G)\beta_2 + \log_2(p)\beta_3 + \epsilon$. The exponentiated values are a multiplicative effect on the predicted median seconds per effective sample for each doubling in the corresponding predictor (K,G or p).

factor	mult. effect	Est.	lower 95%	upper 95%
K	2^{β_1}	3.3	2.8	3.8
G	2^{β_2}	1.2	1.0	1.4
p	2^{β_3}	0.5	0.4	0.6

2.7 Example: Paschold maize data

Paschold et al. (2012) produced an RNA-seq data set for gene expression of root tissue from 4 samples for each of two recombinant inbred maize genotypes, B73 and Mo17. Each sample was obtained from a combination of 10 primary roots from seedlings 3.5 days after germination. Illumina's Genome Analyzer II was used for sequencing. Two flow cells were utilized, with four replicates of each genotype split across flow cells. Among the researchers' aims was to identify genes which were differentially expressed in these parental lines.

Normalization and transformation For RNA-seq data, column-wise normalization (often referred to as 'between-sample' normalization) is required to adjust for systematic differences arising from the sequencing process. These systematic differences obfuscate, and are irrelevant to, the quantities of interest.

For example, the total of all read counts (‘library size’) for a sample will vary, in part, due to differences in sequencing depth or total read count which lead to some columns having significantly higher total reads than others. A simple fix is to divide, for each sample and gene, by the library size of the sample. Unfortunately, this approach to normalization is problematic because the library size is often driven primarily by a few highly sequenced genes, making this approach too volatile. Fixes to this problem have been proposed; we used the trimmed mean of M-values (TMM) method of Robinson and Oshlack (2010) to obtain an ‘effective library size’. Other approaches have been proposed; for further discussion, see Oshlack et al. (2010).

Because the data are non-negative and variability in expression tends to increase with the overall expression, gene expression data are typically analyzed on a logarithmic scale, with difference in expression being multiplicative. Because of the presence of zeros, we add one to each count before taking logarithms and subtracting the log of the effective library size: $y_{gn} = \log(R_{gn} + 1) - \log(L_n)$, where R_{gn} is the number of reads for sample n, gene g and L_n is the effective library size for the n^{th} sample.

The result after applying these steps to the data in Table 2.1 are shown in Table 2.3.

Table 2.3: Same sample of RNA-seq total read counts as in Table 2.1, after normalization and transformation. Weighted trimmed mean of M-values normalization was performed to estimate effective library size using the `calcNormFactors` function in `edgeR` Robinson et al. (2010).

	$B73_1$	$B73_2$	$B73_3$	$B73_4$	$Mo17_1$	$Mo17_2$	$Mo17_3$	$Mo17_4$
$gene_1$	-9.66	-9.70	-9.55	-9.54	-14.71	-14.62	-15.51	-15.48
$gene_2$	-10.13	-10.03	-10.09	-10.06	-10.13	-10.16	-10.20	-10.31
$gene_3$	-8.83	-8.74	-9.23	-9.37	-8.67	-8.65	-8.95	-8.94
$gene_4$	-13.60	-13.59	-14.25	-14.99	-13.90	-12.97	-13.90	-14.23
$gene_5$	-15.47	-15.38	-16.04	-14.99	-9.24	-9.16	-9.15	-9.41

2.7.1 Model

Let y_{gn} be the normalized log-count for gene g in sample n. We assume the normal model of (2.1) where the design matrix for each gene is

$$\begin{array}{c c}
\textit{Sample} & \begin{array}{ccc} \text{intercept} & \text{Mo17} & \text{flow cell} \end{array} \\
\begin{array}{r} \text{B73}_1 \\ \text{B73}_2 \\ \text{B73}_3 \\ \text{B73}_4 \\ \text{Mo17}_1 \\ \text{Mo17}_2 \\ \text{Mo17}_3 \\ \text{Mo17}_4 \end{array} & \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}
\end{array}.$$

We fitted the BNP model using four independent chains. Random initialization points were generated by setting unique random seeds and following the steps outlined in Section 2.5.4, using 5000 iterations for the initial exploration, and then reordering the cluster indices by π. 50,000 iterations were run for each chain after 10,000 iterations of burn-in. Potential scale reduction factors are shown in figure 2.7. Values close to 1 are consistent with convergence; 1.1 has been suggested as a threshold (Gelman et al., 2014b). While some parameters exceeded 1.1, the vast majority were close to 1.

2.7.2 Comparison of methods

In addition to our method, we also fit the data using two other methods/models: a gene independent model, fitted using least squares, and a hierarchical linear model, fitted by REML, which assumes constant variance and assumes a multivariate normal distribution for the random effects. These models were chosen to show illustrate the nonparametric shrinkage achieved by the BNP model, not because they are recommended or widely used methods for the analysis of these sorts of data.

Gene independent model Estimate β_g in Equation 2.1 by minimizing $\sum_{n=1}^{N} r_{gn}^2 = \sum_{n=1}^{N} \left(y_{gn} - x_n \hat{\beta}_g \right)^2$. This method also provides an estimate of σ_g^2, $\hat{\sigma}_g^2 = \sum_{n=1}^{N} r_{gn}^2/(N-p)$.

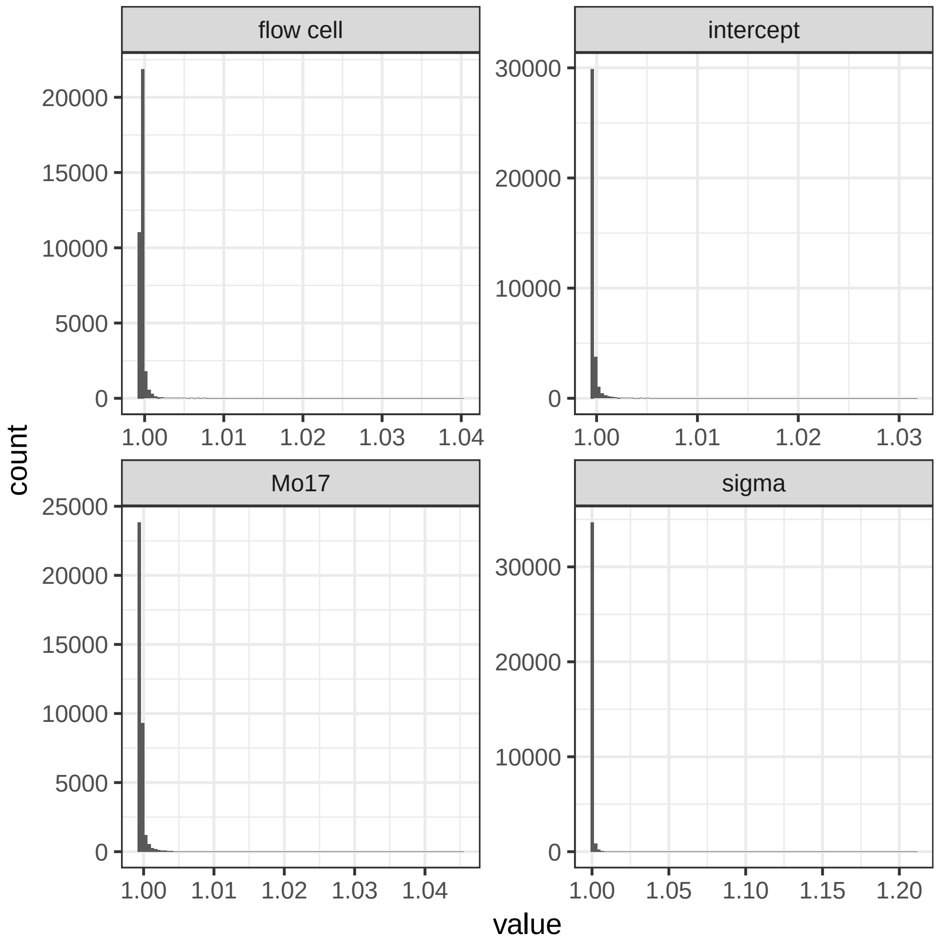


Figure 2.7: Gelman-Rubin potential scale reduction factors for all gene-specific parameters.

Hierarchical/random effects model In addition to Equation 2.1, assume $\sigma_g^2 = \sigma^2$ for all $g = 1, \ldots, G$, and that $\mathcal{P} = \mathrm{N}(\mu, \Sigma)$, for some $\mu \in \mathbb{R}^p$ and Σ a positive definite matrix. This model was fit using the *lmer* function in the *lme*4 package in *R* (Bates et al., 2014; Ihaka and Gentleman, 1996).

In contrast to the BNP model, the gene independent model puts no assumptions on the distribution of gene-specific parameters and borrows no information across genes. The hierarchical linear model assumes that the β_g parameters are realizations of a multivariate normal distribution. This allows for borrowing of information, since μ and Σ are estimated using all of the data and inference for β_g is moderated by them.

Since the hierarchical linear model restricts $\mathcal{P}$ to the family of multivariate normal distributions, the effect of borrowing information via $\mathcal{P}$ is to pull all estimates for β_g toward a common value. Figure 2.8 shows the empirical density of the independent estimates alongside estimates of the underlying distribution for the BNP and normal hierarchical models. These estimates are obtained by binning 10^6 draws of the gene specific parameters – from the posterior distribution of P for BNP and from the estimate of $\mathcal{P}$ for the hierarchical model. To draw β_{new} from $\mathcal{P}|y$, we simply draw $\mathcal{P}^{(s)}$ uniformly from $\{s \in 1, \ldots, S\}$ and draw β_{new} from $\mathcal{P}^{(s)}$. Intuitively, by incorporating local detail in the distribution of the component-specific parameters, the BNP model can direct shrinkage more effectively than the normal hierarchical model. In Figure 2.9, we compare the bivariate histograms of the point estimates obtained by the hierarchical model (middle column) to the gene independent model (left column), and note that there is detectable shrinkage. This shrinkage preserves the shape of the distribution of estimates obtained by the gene independent model while reducing the magnitude of the estimates by shrinking them toward a common value. By comparison, of the point estimates obtained by the BNP model (posterior means), some are shrunk while some are not and the degree of shrinkage in many cases is more dramatic. For example, in the three-way comparison of the bottom panels, the BNP shows aggressive shrinkage, relative to the hierarchical model, of the estimates of Mo17 toward zero for those genes with the most negative flow cell effects. Figure 2.10 shows this phenomena in greater detail. The left-panels show a zoomed in version of the lower-left plot in Figure 2.9, with a bounding box identifying a region of interest. The

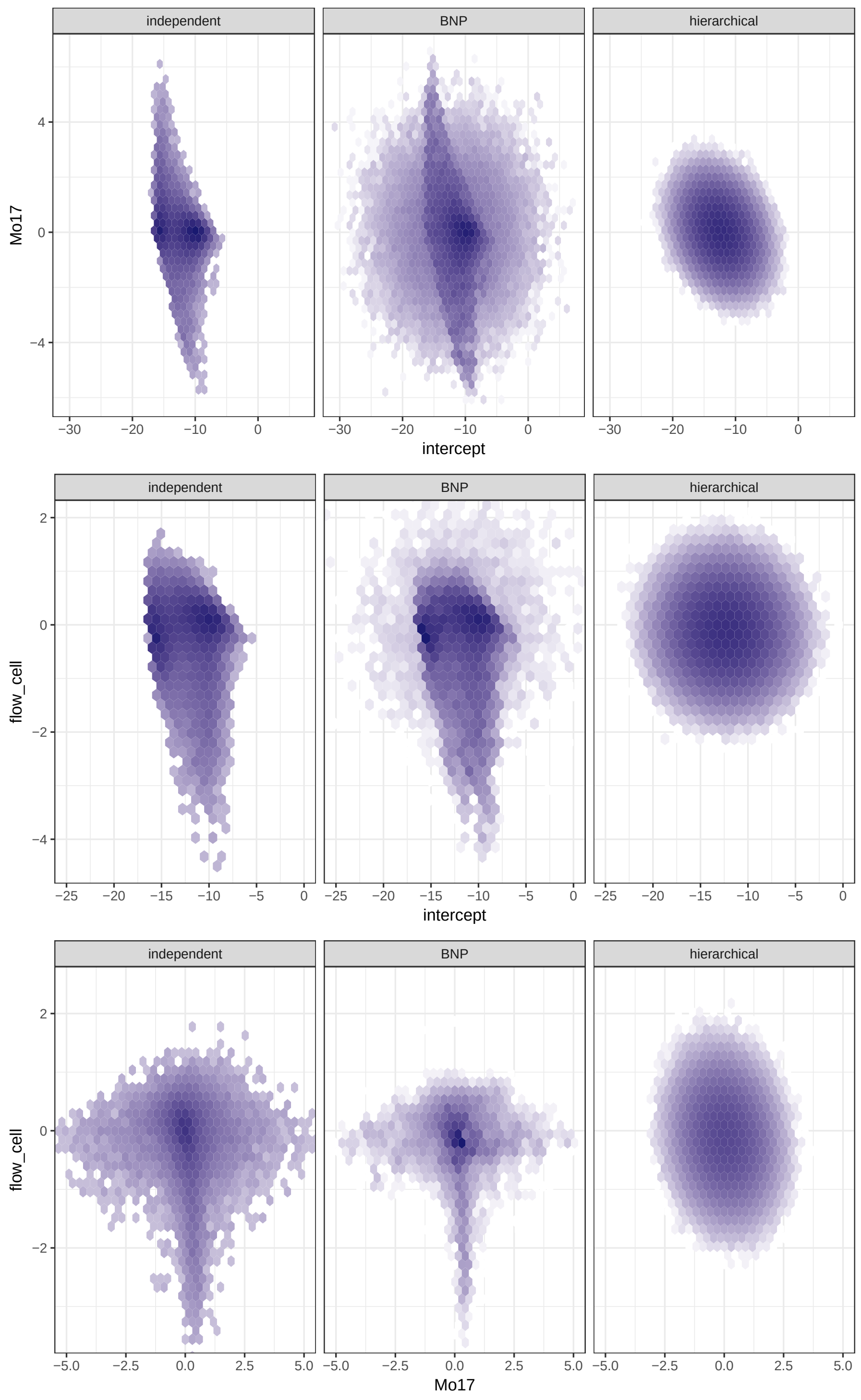


Figure 2.8: Estimates of the underlying distribution of gene-specific parameters, $\mathcal{P}$, based on three models. Because we use a normal base measure, the posterior expection of $\mathcal{P}$ under the BNP model represents a mixture of the the base measure and an empirical measure based on the data. The hierarchical normal model estimates the distribution with a single multivariate normal distribution.

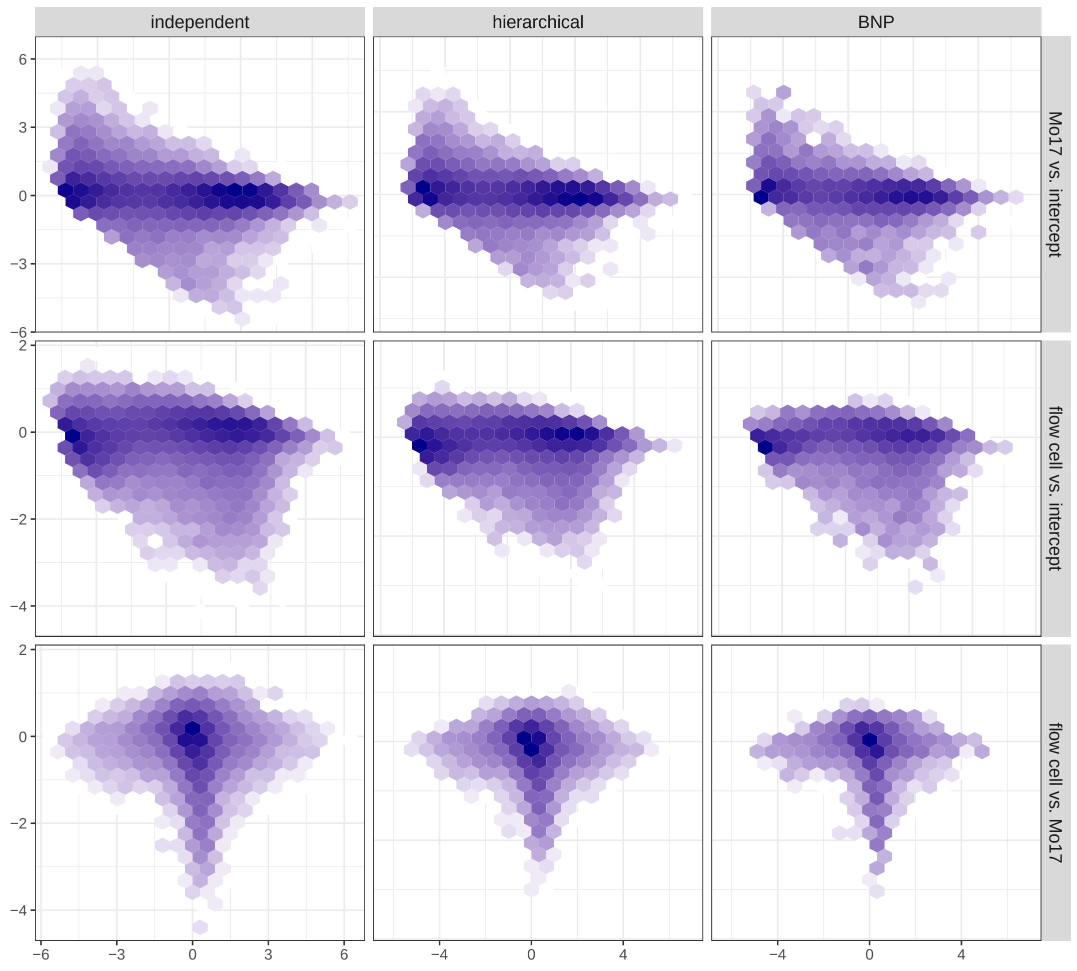


Figure 2.9: Histogram of point estimates across genes, showing pairwise comparisons, using hexagonal binning. Relative to the independent estimates obtained by least-squares, the estimates of the normal hierarchical model shrinks all estimates toward a common mean, while the Bayesian nonparametric model shrinks estimates toward an underlying distribution learned from the data.

next plot to the right shows the gene independent estimates from that region, while the last two plots in the row show the corresponding estimates for the hierarchical and BNP models. The BNP estimates are much less predictable, but they show clear local shrinkage toward regions with high density in the left-most plot.

Figure 2.12 helps to visualize the differences between the shrinkage methods from a wider vantage. Here we see differences in the binned densities of point estimates for β_g for the models we considered taking two elements at a time. The single plot in the fourth row shows the distribution of differences between a histogram of the independent estimates of (intercept, σ)

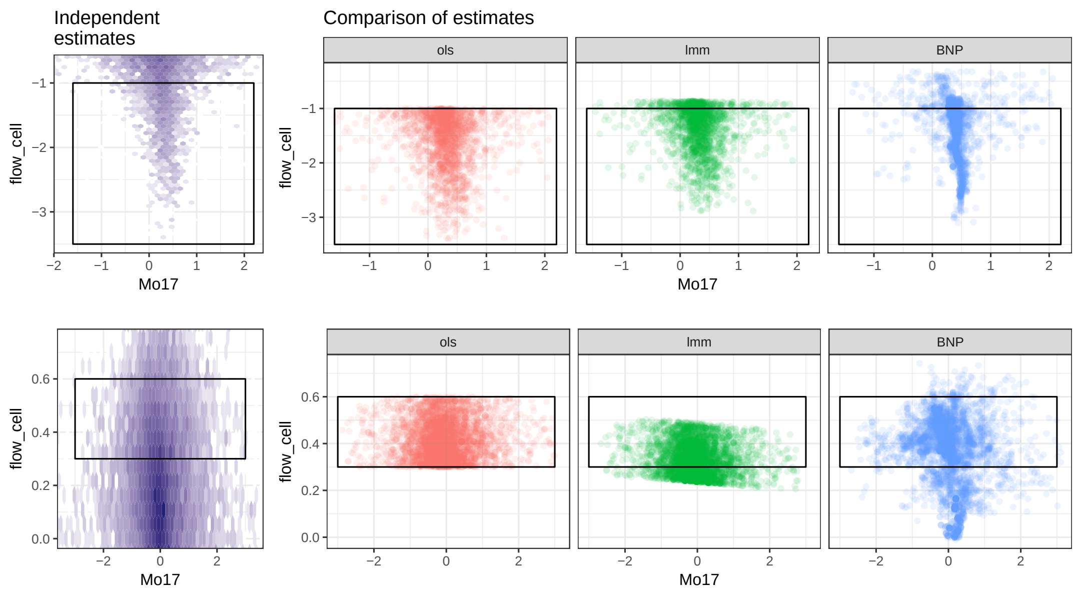


Figure 2.10: Comparing shrinkage in β_{g2} and β_{g3} under the hierarchical and BNP models. The each row shows, with a bounding box, a selected region of the parameter space. The left-most plot shows the empirical density of the gene independent estimates; these are isolated in the next plot to the right. The last plots in each row show estimates for the same genes under the hierarchical model and the BNP model, respectively. The hierarchical model shrinks monotonically and preserves most of the relative positioning of the estimates, whereas the BNP model shrinks toward nearby regions with high density (in the left-most plot).

and the corresponding BNP estimate. Comparisons to the hierarchical model are not shown because that model assumed a constant error variance. The biggest difference between the hierarchical and BNP models is in the third row, which concerns the joint distribution of flow cell and Mo17 effects. The estimates obtained by BNP are shrunken toward two perpendicular line segments which follow the center of the distribution of the OLS estimates. The normality assumption of LMM leads to the points along the vertical axis being all shrunk monotonically toward the origin. The last plot shows that the BNP estimates of σ_g are shrunken toward a curve. This curve represents a global mean-variance trend across genes. This is interesting because compensating for such a trend is a recognized objective in the analysis of RNA-seq data.

2.8 Discussion

Although quite computationally intensive, nonparametric Bayesian methods are becoming computationally feasible in applications, even for fairly sizable datasets, such as gene expression

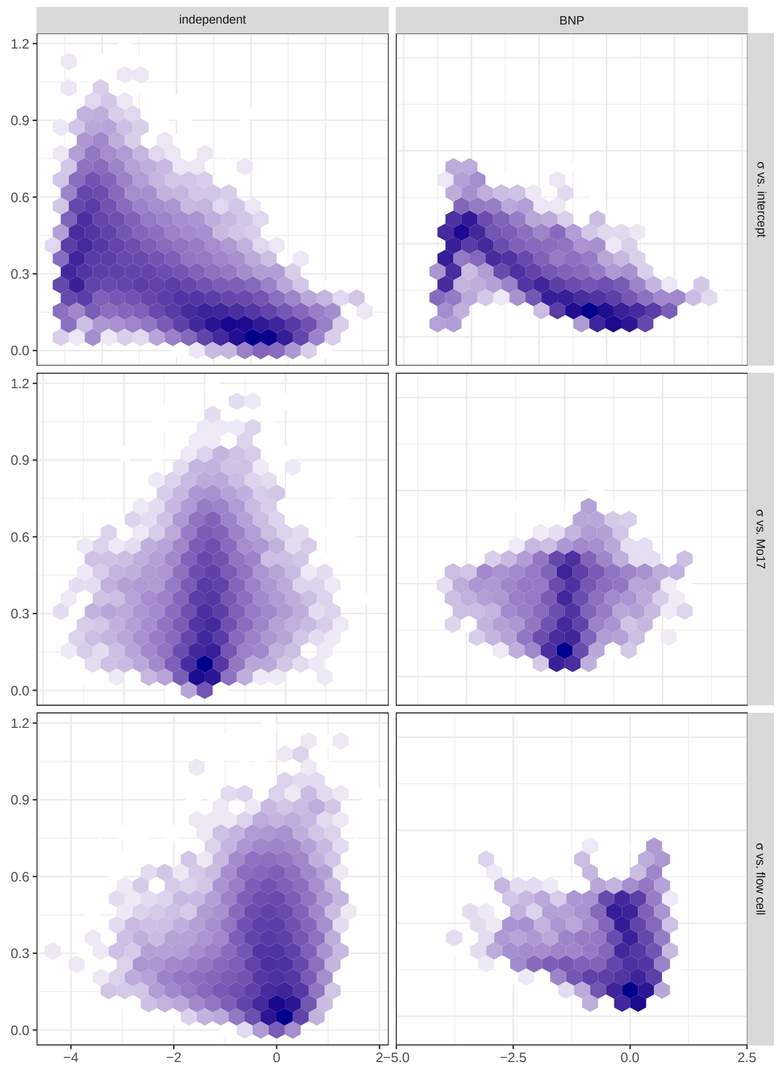


Figure 2.11: Bivariate histogram of point estimates across genes, using hexagonal binning. Relative to the independent estimates obtained by least-squares ("OLS"), the Bayesian nonparametric model shrinks estimates toward an underlying distribution learned from the data.

profiling. One way to achieve computational feasibility is through adapting existing procedures to take advantage of massively parallel systems like the GPU.

The large number of zero counts usually present in RNA-seq data, preclude taking logarithms of the raw data. We have circumvented the problem by adding one to the counts prior to normalization. There is theoretical support for the use of a Poisson model for RNA-seq counts absent any biological variation, a fact which is often used to motivate the use of negative binomial distribution for RNA-seq data, the negative binomial being a gamma mixture of Poissons. We opted to use a normal model for the log-frequency rather to model the counts directly. It would be possible to modify our method to use a negative binomial model for the counts, although there would be additional computational cost because the sufficient statistics are less convenient to work with.

We have found that our proposed procedure scales well and is computationally feasible with data sets containing tens of thousands of components like that which is common currently with gene expression data. In addition, design matrices up to 6 do not present a computational problem for the procedure. For larger sample sizes, we would expect that a larger number of clusters would be required to explain the data. Increasing K is a computational burden; we found the time per effective iteration to be quadratic in K. Also, the potential benefits of borrowing information diminish with sample size, reducing the potential benefit of a hierarchical model.

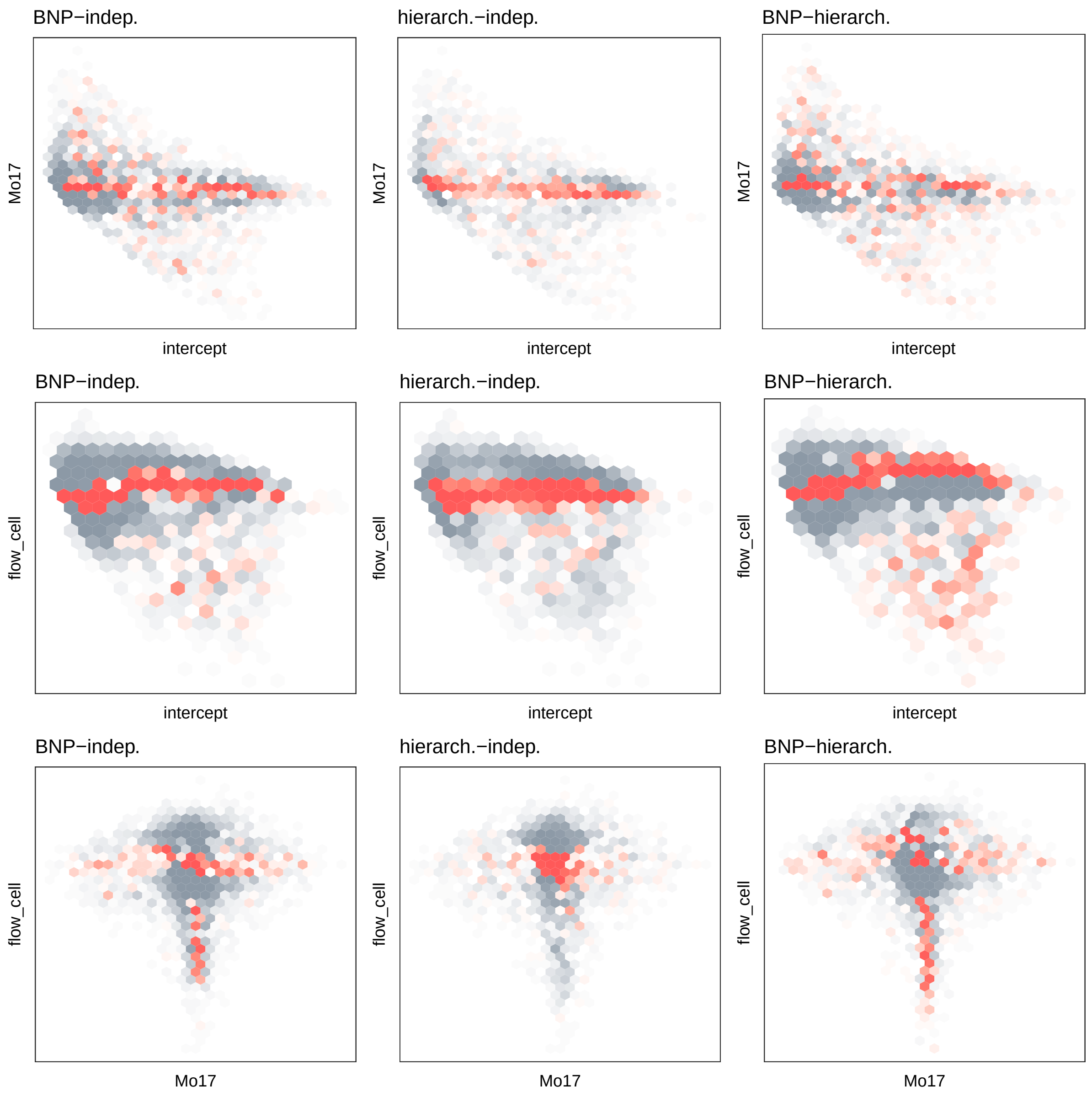


Figure 2.12: Difference of histograms of point estimates for gene-specific parameters under different models. Both BNP and LMM show shrinkage toward zero relative to OLS. The shrinkage with LMM is monotonic toward zero, while the shrinkage with BNP varies across the parameter space.

CHAPTER 3. Detection of gene heterosis in maize using a Bayesian nonparametric model

3.1 Introduction

Heterosis, or hybrid vigor, refers to biological differences seen in offspring of two inbred parents, which give the hybrid qualities superior to either parent. While the phenomenon is well-known and widely exploited, its causes at the molecular level are not well understood (Paschold et al., 2012). New technologies that can simultaneously measure the expression of tens of thousands of genes provide an opportunity to shed light on these mechanisms.

Recently, RNA-seq technology has supplanted microarrays as the primary platform for these studies (Wang et al., 2009). Statistical methods to analyze RNA-seq data have proliferated (Law et al., 2014; Love et al., 2014; McCarthy et al., 2012; Liu et al., 2015; Landau and Niemi, 2016). Because cost limits the number of samples that can be sequenced, the identification of interesting genes in gene expression profiling studies falls into the "$n \ll p$" paradigm. In this paradigm, noise in the data tends to dominate the signal. Various statistical methods have been developed to mitigate this problem. One general approach is to hierarchical modeling.

Hierarchical, or multi-level, models provide a mechanism to borrow strength across groups present in the data by learning the underlying distribution of the first-level model parameters. The case for using hierarchical models for RNA-seq data is compelling because there are a large number of genes, potentially providing substantial information about the distribution of gene-specific parameters, as well as a strong incentive to borrow information since there are often few samples with which to estimate any given gene-specific parameter. Niemi et al. (2015) observed that the choice of parametric family for the hierarchical model impacts both estimation and the ranking of genes. It seems that appropriate modeling of the underlying distribution of the gene-specific parameters will lead to better results. To avoid misspecification of the distribution of gene-specific parameters, one might choose to model them nonparametrically; for example, Liu et al. (2015) proposed a semiparametric model for differential expression. In their model, the parameter representing the mean log-fold-change between the two treatments was modeled with a Dirichlet process (DP).

RNA-seq allows for direct mapping of reads to genetic features, so measurements are counts of total reads. While most RNA-seq methods we have seen use the negative binomial (NB) distribution McCarthy et al. (2012) provided a theoretical motivation for directly modeling the counts via a Poisson mixture, Law et al. (2014) demonstrated a method for estimating precision weights, making a normal data model a viable alternative to count-based methods. We choose to take advantage of the normal model, which allows us to exploit conditional conjugacy work with lower-dimensional sufficient statistics. This eases the computation burden, which can be a considerable benefit when conducting fully Bayesian inference.

The road map for this paper is as follows: In Section 3.2, we describe the heterosis data of Paschold et al. (2012). Next, in Section 3.3.2, we first motivate the use of Bayesian nonparametric models, using an example to illustrate the consequences of misspecification in hierarchical models as well benefits of using a flexible DP prior. We then propose a Bayesian nonparametric model that extends the approach taken in Liu et al. (2015) by modeling all gene-specific parameters with a DP. After specifying the model, we describe our implementation including data steps prior to modeling. Section 3.4.1 describes a simulation study comparing detection of heterosis and point estimates obtained from our model to those obtained using a non-hierarchical approach but that uses the same data normalization and precision weighting. In Section 3.4.2 we present a second simulation study, also focused on detection of heterosis and parameter estimation, that compares performance of our method to two popular negative-binomial based methods, `DESeq2` and `edgeR` (Robinson et al., 2010; Love et al., 2014). Next, in Section 3.5 we present results for the Paschold data set and contrast them with those from the original paper Paschold et al. (2012) and with those obtained by Landau and Niemi (2016) who used an independent parametric model for the gene-specific parameters.

3.2 RNA-seq gene expression data

The starting point for our analyses is an array of RNA-seq total aligned read counts measuring the abundance of a particular messenger RNA transcript in a sample. Such an array consists of G rows and N columns, corresponding to genes and samples, respectively. The abundance of a transcript is often referred to as gene expression; we will follow this convention here. For more details on data collection and preprocessing of RNA-seq data, please see Datta and Nettleton (2014).

A simple analysis of RNA-seq data may be achieved by analyzing each gene independently. The experimental design can generally considered the same for all genes. Select rows for an example data set are shown below in Table 3.1.

Table 3.1: Selected rows from RNA-seq data set showing total aligned read counts for selected genes. Columns are grouped by genotype. Exemplar genes for different types of heterosis are shown.

GeneID	type of heterosis	B73				B73xMo17				Mo17xB73				Mo17			
GRMZM2G306345	HPH	1431	1199	1235	1569	2055	1652	2149	2168	2415	1815	2142	2369	1127	987	1672	1518
GRMZM2G149543	LPH	86	62	131	128	52	43	85	95	60	23	63	74	85	71	178	205
GRMZM2G079613	mixed	122	98	146	150	73	77	159	127	178	122	259	252	108	103	207	171
AC194005.3_FG004	neither	30	31	42	48	17	15	22	26	16	13	16	22	2	2	2	2

3.2.1 Maize data of Paschold et al.

The data come from Paschold et al. (2012). We work with the count of total aligned reads for two parental lines, B73 and Mo17, and the reciprocal hybrid genotypes, B73×Mo17 and Mo17×B73, for 39,656 genes. Of these, we exclude 2835 genes where all counts were exactly zero. Each genotype had 4 biological replicates, each produced from primary roots of 10 kernels 3.5 days after germination. Sequencing of the 16 samples was done using Illumina methodology and equipment in 1 run using 2 flowcells. Reads were mapped to the whole reference genome using the short reads aligner, NOVOALIGN. For more specifics, please see Paschold et al. (2012).

The counts range in value from 0 to 38,006 with a mean value of 255.5 and a median value of 37. Figure 3.1 shows the distribution of the total read counts plus 1 by sample. While the distributions look fairly consistent across samples, there are some differences in the lower

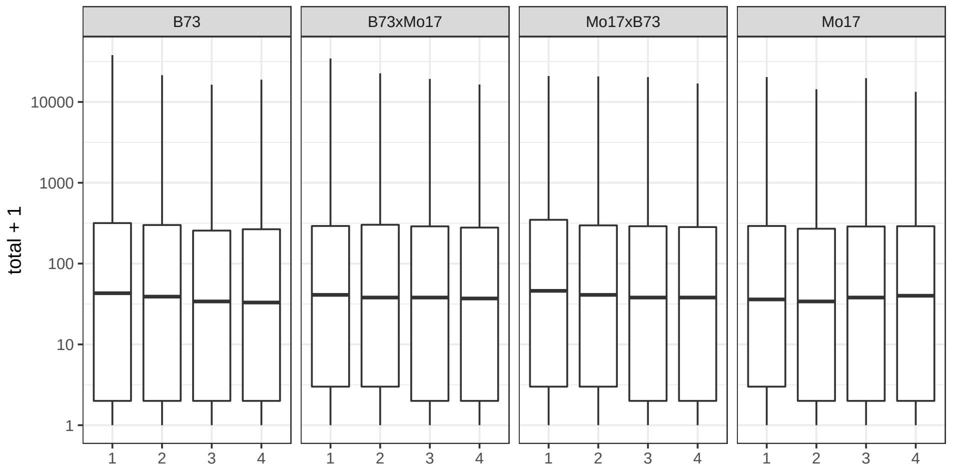


Figure 3.1: Distribution of total read counts plus 1 by sample, grouped by genotype, for all 16 RNA-seq samples in the maize data.

quartiles. The replicate numbers on the x-axis can be used to identify the flow cell; replicates 1 and 2 were sequenced in flow cell 1, while replicates 3 and 4 were sequenced in flow cell 2. The differences between the flow cells shown in the lower quartile of the boxplots for the hybrids suggest that flow cell 1 was more thoroughly sequenced than flowcell 2.

To identify genes which are likely to play a role in heterosis, we look for genes where the ordering of the average expression levels for the 4 genotypes exhibits one of several patterns. We say a gene expresses heterosis, when the expected value of a hybrid genotype differs from the average of the expected values of the two parents. To be precise, we define multiple kinds of heterosis in terms of the mean expression levels of the genotypes. High-parent heterosis (HPH) occurs when the expected value of a hybrid is higher than that of both parents. Similarly, low-parent heterosis (LPH) occurs when the expected value of a hybrid is lower than that of both parents. The first two genes (rows) in Table 3.1 are exemplar genes of HPH and LPH, respectively. The third gene appears that B73$\times$Mo17 displays LPH while Mo17$\times$B73 displays HPH, and in the last gene the hybrid expression appears to lie between that of the parents. Here too, we see the apparent flow cell effects, since within a genotype there appear to be consistent differences between the first and second pairs of replicates.

3.3 Bayesian nonparametric model

The advantage of hierarchical modeling generally is that that by assumming a distribution on some model parameters, we can improve inference by borrowing information, i.e. allowing data which doesn't have a direct dependence on a parameter to inform indirectly about the parameter through the parameter's distribution. Misspecification of the form of that hierarchical distribution can lead to incorrect borrowing of information. In practice, this issue is often overlooked. One reason for this is that in many cases it is difficult to check whether the hierarchical distribution is misspecified. Another reason is that there can be utility to using a hierarchical model, even when the hierarchical distribution is misspecified.

3.3.1 Illustration

To provide an example of the ideas in the preceding paragraph and to motivate the development of our model, we given the following example. We simulate data as follows, $\mu_g \overset{ind.}{\sim} 1/3\,\mathrm{N}(-4,1) + 1/3\,\mathrm{N}(0,1) + 1/3\,\mathrm{N}(4,1)$. Conditional on μ, we simulate $y_{gn}|\mu_g \overset{ind.}{\sim} \mathrm{N}(\mu_g, \sigma^2 = 2^2),\ g = 1, \ldots, 200,\ n = 1, 2, 3$. Now, suppose we observe y and assume the (true) data model, but μ_g and the common error variance, σ^2, are all unknown. A histogram of the sample means, $\hat{\mu}_g$ is shown in the top panel of Figure 3.2. Next, we fit three Bayesian models to the data.

1. Normal model: $\mu_g \overset{ind.}{\sim} \mathrm{N}(\eta, \tau^2)$, with diffuse proper priors on η and τ^2

2. Correct model: $\mu_g \sim \pi_1\,\mathrm{N}(\eta_1, \tau^2) + \pi_2\,\mathrm{N}(\eta_2, \tau^2) + \pi_3\,\mathrm{N}(\eta_3, \tau^2)$, also with diffuse proper priors on η_j and τ^2 and a $\mathrm{Dir}(1, 1, 1)$ prior on π

3. DP model: $\mu_g \overset{ind.}{\sim} \mathcal{P}$ with $\mathcal{P} \sim \mathrm{DP}\left(\alpha\,\mathrm{N}(0, 5^2)\right)$ and an informative prior on α

.

Critically, this last model allows us to be agnostic about the modality of $\mathcal{P}$; further details are given in the next section.

The bottom three panels of Figure 3.2 show density estimates and 90% pointwise credible intervals for the predictive distributions for a new μ_g based on the posteriors the three models. For the Normal model and the Correct model, these were computed by taking quantiles of the

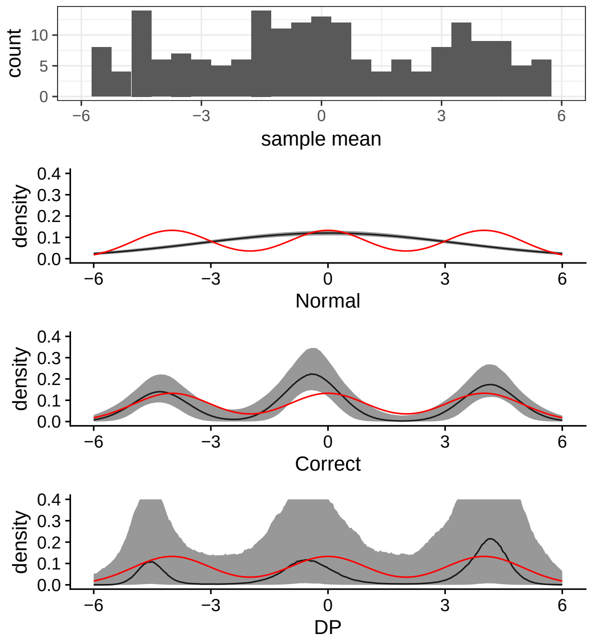


Figure 3.2: Top: Sample averages for the simulated trimodal data. The next three rows correspond to pointwise posterior estimates and 90% credible intervals for the predictive density for a new μ_g. For the DP model, a weighted kernel density estimate employing a bandwidth of 0.1 was used for each posterior draws of $\mathcal{P}$, as the posterior draws do not have a density with respect to Lebesgue measure.

sampled density evaluated on a grid. For the DP model, the sample density was estimated using a weighted kernel with bandwidth of 0.1 because samples of $\mathcal{P}$ do not have a density with respect to Lebesgue measure. The true generating model for the μ_g is shown in red. The posterior for the density of μ_g under the normal model has less uncertainty but is concentrated around an incorrect answer because of its inflexibility. Both the Correct and the DP models contain much of the true density within the pointwise uncertainty intervals. The DP model shows a great deal more posterior uncertainty because it does not assume a parametric model for μ_g. Despite the increased uncertainty in the predictive distribution, the posterior estimation under the DP model is nearly as good as it is under the true model; the left panel of Figure 3.3 shows the lengths of the posterior 95% credible intervals for the μ_g sorted by their length for the three models; the dotted lines represent correspond to a non-hierarchical analysis (independent, uniform priors on μ_g) with σ^2 known $(+/-4/\sqrt{3})$. For this data set, we see that both the nonparametric and true models have less posterior uncertainty than the normal model on average, while all three hierarchical models have substantially less uncertainty than a non-hierarchical model. The right panel shows histograms of the gene specific RMSE, $[\int(\mu_g - \mu_{g0})^2 p(\mu_g|y)d\mu_g]^{1/2}$, computed for each g under the three models, where μ_{g0} is the true value. Again, we see that estimation is substantially improved when the hierarchical distribution is modelled appropriately.

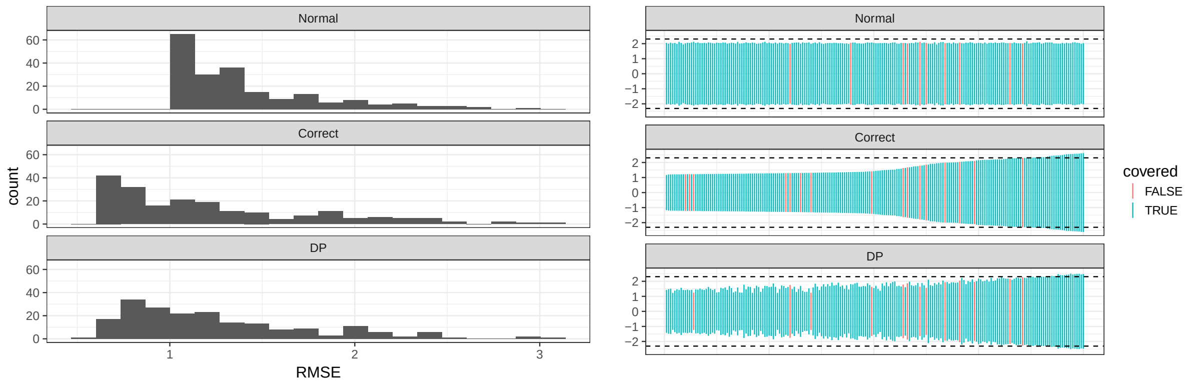


Figure 3.3: Left: Histograms of gene specific RMSE based on posteriors for these models. Right: The widths of the 95% posterior credible intervals for μ_g under the three hierarchical models, arranged by the width for the true model. Intervals which failed to cover the true value are colored red. The dashed lines correspond approximately to credible intervals with independent uniform priors on μ_g.

3.3.2 BNP model for RNA-seq data

Let y_{gn} represent the normalized log-cpm for gene g, sample n and let X be the model matrix. Let $x_n^\top$ be the row of a model matrix corresponding to sample n. Let $g = 1, ..., G$ and $n = 1, ...N$ index genes and samples, respectively. We model

$$y_{gn} \sim \mathrm{N}\left(x_n^\top \beta_g, \frac{\sigma_g^2}{w_{gn}}\right) \tag{3.1}$$

where w_{gn} is a given relative precision for y_{gn}. While not critical to our method, the inclusion of these precisions allows us to avoid dependence on the assumption of constant variance within genes. This is discussed further in Section 3.3.3. As discussed in the previous Section, we intend to model the gene-specific parameters nonparametrically, letting the vector of gene specific parameters follow a Dirichlet process. This is denoted by

$$\left(\beta_g^\top, \sigma_g^2\right) \overset{ind.}{\sim} \mathcal{P}, \quad \mathcal{P} \sim \mathrm{DP}(\alpha \mathrm{Q}), \tag{3.2}$$

where DP(αQ) is a Dirichlet process with concentration parameter, α, and base distribution Q.

The DP satisfies two requirements proposed by Ferguson (1973) for priors over probability distributions, that it have large support on the space of probability distributions and that it be computationally tractable (Ferguson, 1973). The tractibility of the DP is important, because it makes inference computationally feasible. The large support allows us to discard unwarranted parametric assumptions about $\mathcal{P}$.

The stick-breaking characterization of the DP, due to Sethuraman (1994), says that

$$\mathcal{P} = \sum_{k=1}^{\infty} \pi_k \delta_{(\tilde{\beta}_k^\top, \tilde{\sigma}_k^2)}, \quad \pi_k = \nu_k \prod_{l<k} (1-\nu_l), \quad \nu_k \overset{ind.}{\sim} \text{Beta}(1,\alpha),\ \alpha > 0, \tag{3.3}$$

where $\left(\tilde{\beta}_k^\top, \tilde{\sigma}_k^2\right) \overset{ind.}{\sim} Q$, and where δ_x represents the Dirac delta function. It is clear from Sethuraman's representation of the DP that $\mathcal{P}$ is almost surely discrete. The name, "stick-breaking", given to this characterization, is illuminated by noting the specification of the cluster weights, π_k, in terms of the independent ν_k can be rewritten as

$$\pi_k = \nu_k \left(1 - \sum_{l<k} \pi_l\right), \tag{3.4}$$

so ν_k has the interpretation of the relative size of π_k as compared to the remaining "stick", $1 - \sum_{l<k}$.

A consequence of (3.3) is that $\text{E}(\pi_k) < \text{E}(\pi_l),\ k > l$ and that $E(\pi_k) = [1/(\alpha+1]\,[\alpha/(\alpha+1)]^{k-1} = \alpha\,[\alpha/(\alpha+1)]^k$, which tell us that π_k decreases in expectation in k, is decreasing quickly for small values of α and slowly for large values. It is important to understand that this model gives positive prior probability that $(\beta_g^\top, \sigma_g^2) = (\beta_{g'}^\top, \sigma_{g'}^2)$ for any g and g'. Another related fact involving α, pointed out by Antoniak (1974), is that the expected number of unique values of $(\beta_g^\top, \sigma_g^2)$ is

$$\sum_{i=1}^{G} \frac{\alpha}{\alpha + i - 1}. \tag{3.5}$$

The bet made on sparsity by assuming this latent clustering allows for such flexibility in learning $\mathcal{P}$, without overfitting the model to the data.

Equation (3.4) suggests that for large K, π_ℓ are negligible for $\ell > K$. We use this fact to justify our use of a truncated approximation to (3.3),

$$\mathcal{P} = \sum_{k=1}^{K} \pi_k \delta_{\left(\tilde{\beta}_k^\top, \tilde{\sigma}_k^2\right)}, \quad \pi_k = \nu_k \prod_{l<k}(1-\nu_l), \quad \nu_k \overset{ind.}{\sim} \text{Beta}(1,\alpha),\ \alpha > 0,\ k < K,\ \nu_K := 1. \quad (3.6)$$

This is the truncation approximation to the DP presented by Ishwaran and Zarepour (2000).

3.3.3 Normalization and precision weights

Between sample normalization Artifacts of the sequencing procedure can lead to some samples having larger or smaller read counts on average relative to other sample. This introduces biases that require adjustment. These corrections are typically made at the time of analysis by introducing a sample specific offset. Robinson and Oshlack (2010) proposed a method called TMM, which uses a trimmed mean of log-fold change between two samples to robustly estimate the systemic bias. For details, see Robinson and Oshlack (2010).

Precision weights adjust for non-constant variance An argument in favor of the use of overdispersed Poisson models in general, and negative binomial models in particular, for RNA-seq data is that the quadratic mean-variance relationship implied by these models generally agrees with patterns observed in RNA-seq counts. By an argument of McCarthy et al. (2012), the variance of a count, r_{gn}, is $\text{Var}(r_{gn}) = \mu_{gn} + \phi_g \mu_{gn}^2$ under the assumption that r_{gn} is a mixture of Poisson with mean μ_{gn} and that $\phi_g > 0$ is the coefficient of variation for the mixing distribution. In contrast, the log-normal model has a mean-variance relationship given by $\text{Var}(r_{gn}) = \phi_g \mu_{gn}^2$ (using a non-standard mean parameterization with $\phi_g = \exp(\sigma^2) - 1$ and ignoring problems related to discreteness and zero counts).

Because treatment effects are usually modeled on the log scale for gene expression, consider the delta method approximation for a overdispersed Poisson count in terms of the log-count: $\text{Var}(\log r_{gn}) \approx (1/\mu_{gn})^2(\mu_{gn} + \phi_g \mu_{gn}^2) = 1/\mu_{gn} + \phi_g$. For the normal model, the corresponding expression is $\text{Var}(\log r_{gn}) = \sigma_g^2 \approx \log \phi_g$ for large μ_{gn}.

Law et al. (2014) proposed a procedure called *voom*, whose name comes from "variance modeling at the observational level", as a way to make RNA-seq data compatible with normal linear models. The voom procedure makes a normal model for normalized log-counts a viable alternative to overdispersed Poisson models by assigning to each count a precision weight. The computation of the weights involves a nonparametric regression fit to the square root of gene-wise standard deviations, using the overall log-mean expression for each gene as a predictor. Count specific weights are selected as predicted values based on the nonparametric model fit.

In addition to making viable methods based on normal distributions, Law et al. (2014) suggests additional reasons for preferring to use voom:

- RNA-seq data display a non-ignorable trend in the negative binomial overdispersion parameter with gene abundance, so a nonparametric correction is also part of negative binomial methods
- by adjusting for the global mean variance trend at the count level, voom should provide better precision for genes which large variation in expression versus models which only provide correction at the gene level (via ϕ_g)

To be consistent with *voom* we use a base 2 logarithm to compute log-cpm. Explicitly, log-cpm is computed by

$$y_{gn} = \log_2 \left(\frac{r_{gn} + 0.5}{R_n + 1} \cdot 10^6 \right),$$

where R_n is the effective library size for sample n.

3.3.4 Priors

Q serves as a prior guess at $\mathcal{P}$, with α being the prior "sample size" attributed to that guess. For large values of α, π_k will be small and decrease slowly, so that the expression given for $\mathcal{P}$ in (3.3) will allocate its mass more or less in a way that approximates Q, albeit discretely. If α is small, then $\mathcal{P}$ will allocate most of its mass to a small number of the atoms. For conjugacy, we choose Q to be a product measure of the form

$$\tilde{\beta}_k \overset{ind.}{\sim} \mathrm{N}(m_\beta, C_\beta), \quad \tilde{\sigma}_k^2 \overset{ind.}{\sim} \mathrm{IG}(a_{\sigma^2}, b_{\sigma^2}), \tag{3.7}$$

where $\mathrm{IG}(a,b)$ is an inverse gamma distribution parameterized by shape and scale parameters, a and b, and with density given by

$$p(x|a,b) = \frac{b^a}{\Gamma(a)} x^{a+1} e^{-b/x}.$$

The use of diffuse base measures is not recommended as it can result too few practically significant weights, π_k (Gelman et al., 2014a, p.554). For purposes of comparability, we follow the approach used in Love et al. (2014) for setting the empirical Bayes prior for β_g to set Q. In their method, variances are selected to match the quantiles of the empirical distribution of the independently obtained estimates of β_g, σ_g^2. For details, see Love et al. (2014).

The prior distribution for α is set indirectly by considering the implied expected number of clusters, given in (3.5). Although it is impossible to guess the number of clusters *a priori*, we believe that most genes will have a pattern of expression that is similar to some other gene, so that after grouping the genes into clusters, the number of clusters should be much smaller than the number of genes. Also, due to computational constraints, there is a upper bound on the number of clusters we can handle. Since the number of clusters is expected to scale with G, it is sensible that our prior also takes account of G, a concept also noted by Escobar (1994). We have found that the prior, $\alpha \sim \mathrm{Ga}(a_\alpha = 3, b_\alpha = 3/G^{0.5})$, scales with G and concentrates on a reasonable range of values for the number of expected clusters, between $G^{.25}$ and $G^{.75}$. To give some idea, for data where $G = 30000$, this prior implies a 99% probability on the average number of clusters being between 13 and 2280.

3.3.5 Gibbs Sampler

Our Gibbs sampler for performing MCMC sampling from the posterior distribution for the BNP model is adapted from the blocked Gibbs sampler of Ishwaran and Zarepour (2000). The steps are given below; derivations of the full conditional distributions are provided in Appendix B.

Sample ζ_g, α: α and all ζ_g are conditionally independent. α is conditionally conjugate with full conditional

$$\alpha \sim \mathrm{Gamma}(K + a_\alpha - 1, -\log \pi_K + b_\alpha).$$

$$\zeta_g \overset{ind.}{\sim} \text{Categorical}\left(\hat{\pi}_1, \ldots, \hat{\pi}_K\right), \text{ with}$$

$$\hat{\pi}_k \propto \pi_k \, \text{N}(y_g; X\tilde{\beta}_k, \tilde{\sigma}_k^2 W_g^{-1}).$$

Here W_g is a diagonal matrix consisting of the log-count specific precisions. The ζ_g parameters are categorical random variables with weights proportional to the likelihood of y_g given $\tilde{\beta}_k$ and $\tilde{\sigma}_k$, weighted by π_k. This step is the most computationally intensive — just computing the weights requires the calculation of $G \times K$ such likelihoods. However, using fine-grained GPU parallelism, doing so is computationally feasible.

Sample $\tilde{\beta}_k$:

$$\tilde{\beta}_k \overset{ind.}{\sim} \text{N}(\hat{m}_k, \hat{C}_k),$$

$$\text{where } \hat{C}_k = \left(\tilde{\sigma}_g^{-2} \sum_{g:\zeta_g=k} X^\top W_g X + C^{-1}\right)^{-1}, \text{ and } \hat{m}_k = \hat{C}_k \left(\sum_{g:\zeta_g=k} X^\top W_g y_g + C^{-1} m\right).$$

The $\tilde{\beta}_k$ are conditionally independent. Note that if $\zeta_g \neq k$ for all g, then $\tilde{\beta}_k$ is drawn from the base measure.

Sample $\tilde{\sigma}_k^2$

$$\tilde{\sigma}_k^2 \overset{ind.}{\sim} \text{IG}(\hat{a}_k, \hat{b}_k),$$

$$\text{where } \hat{a}_k = a_{\sigma^2} + \frac{1}{2} N M_k, \text{ and } \hat{b}_k = b_{\sigma^2} + \frac{1}{2} \sum_{g:\zeta_g=k} y_g^\top W_g y_g - 2\beta_g^\top X^\top W_g y_g + \beta_g^\top X^\top W_g X \beta_g$$

Here M_k is the number of g for which $\zeta_g = k$. Again, the $\tilde{\sigma}_k$ are conditionally independent.

Sample ν_k

$$\nu_k \overset{ind.}{\sim} \text{Be}(M_k + 1, \sum_{\ell > k} M_\ell + \alpha).$$

Given α and ζ, the ν_g are conditionally independent.

3.3.6 Assessing convergence

The validity of our inference depends on adequate mixing of our MCMC chains. To assess this convergence where only a single chain was run, Geweke diagnostic were calculated for the gene-specific parameters, which, upon convergence are normally distributed with a standard deviation of 1.

When multiple chains were run for the same target we calculated Gelman-Rubin potential scale reduction factors (Rhat) instead of Geweke diagnostics for the gene-specific parameters. This is computationally convenient because it can be performed using sample moments and vectorized across genes. Upon convergence, Rhats converge to 1.

3.4 Simulation studies

We conducted two simulation studies to gain test the performance of the BNP model relative to another log linear method using voom as well as some popular count-based methods. We consider two criteria: 1) the average precision, under squared error loss, of the estimated effects, and 2) the accuracy of gene classification for each method of interest.

We overall precision of estimates as mean squared prediction error (MSPE), averaging across genes for each component of β_g, i.e. $\frac{1}{G}\sum_{g=1}^{G}(\hat{\beta}_{g\ell} - \beta_{g\ell})^2$. For the BNP method, we take the posterior mean as the point estimate.

The geometry of the region of the parameter space where gene expression heterosis holds makes it difficult to construct hypothesis tests for heterosis, and such tests are not available in the methods we wish to compare against. Instead, we chose to use the classification of genes whose effects are larger than a threshold to be our criteria for assessing the utility of the different methods.

McCarthy and Smyth (2009) described a statistical test, coined TREAT, to evaluate whether the log-fold-change due to a model term is larger than a specified threshold in absolute value, i.e. it allows the ranking of genes based on the evidence of the hypothesis, $H_{g\ell} : |\beta_{g\ell}| > t$. The authors note that, "The biological significance of a given fold-change is likely to depend on the gene and on the experimental context. On the other hand, it is reasonable to assume that there is a minimum fold-change threshold below which differential expression is unlikely to be of interest for any gene (McCarthy and Smyth, 2009, pp. 765-755)." Their methodology, based on a non-hierarchical log-linear model, implemented in the R-package, `limma`, estimates all of the β_g independently without any pooling. Testing with TREAT consists of a moderated t-test that involves an shrunken estimates of σ_g^2 (McCarthy and Smyth, 2009). For more details, see McCarthy and Smyth (2009). All three competing methodologies considered in this section can be adapted to test log-fold-change greater than a threshold and have functionality provided in their respective software for doing so.

A model matrix for the Paschold et al. (2012) data described in Section 3.2, which is also assumed for our simulations is

$$X = \begin{array}{r} \text{Sample} \\ \text{B73}_1 \\ \text{B73}_2 \\ \text{B73}_3 \\ \text{B73}_4 \\ \text{B73} \times \text{Mo17}_1 \\ \text{B73} \times \text{Mo17}_2 \\ \text{B73} \times \text{Mo17}_3 \\ \text{B73} \times \text{Mo17}_4 \\ \text{Mo17} \times \text{B73}_1 \\ \text{Mo17} \times \text{B73}_2 \\ \text{Mo17} \times \text{B73}_3 \\ \text{Mo17} \times \text{B73}_4 \\ \text{Mo17}_1 \\ \text{Mo17}_2 \\ \text{Mo17}_3 \\ \text{Mo17}_4 \end{array} \begin{array}{c} \begin{array}{ccccc} \text{intercept} & \text{parental HD} & \text{hybrid} & \text{hybrid HD} & \text{flow cell} \end{array} \\ \left(\begin{array}{rrrrr} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & -1 & 1 \\ 1 & 0 & 1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 1 \end{array}\right) \end{array}.$$

3.4.1 Comparison to a non-hierarchical log-linear method

The voom procedure can be used as part of any analysis pipeline that uses precision weights. To assess the performance of our method apart free of consideration of the weights, we simulated log-cpms and performed an unweighted analysis (all $w_{gn} = 1$) using our BNP model as well as the tool suggested by the authors of voom, limma (Smyth, 2005). This second type of analysis was carried out using the package `limma` in `R` (Smyth, 2005). The limma method fits the same linear model to the data, but fits each gene independently. Thus, a comparison of the estimates obtained under BNP model to the limma estimates will show the result of borrowing information across genes via the Dirichlet process prior on $\mathcal{P}$.

For this set of simulations, we determined the true values of the gene-specific parameters by taking a posterior draw of $\mathcal{P}$ from fitting the Paschold et al. (2012) data and sampling 1000 gene-specific parameters from that $\mathcal{P}$. Normal data were then generated for each gene, conditional on the parameters and X. These steps are outlined in list form below. For each simulated data set, a single MCMC chain was run for 200,000 iterations after 10,000 burn-in iterations. These were thinned to yield 4,000 samples for each gene-specific parameter and α. Geweke diagnostics for the gene-specific parameters showed slight departures from normality with a few genes having z-scores greater than 5 in absolute value. The vast majority of genes reported an ESS greater than 1000. The smallest ESS across all genes and all simulations was 235.

1. Select random draw of $\mathcal{P}$, $\mathcal{P}^{(s)}$, obtained from the posterior distribution given the entire Paschold data set

2. Sample (β_g, σ_g^2) independently from $\mathcal{P}^{(s)}$, $g = 1, \ldots, 10000$

3. Sample $y_g \sim N(X\beta_g, \sigma_g^2 I)$

3.4.1.1 Classification accuracy

We can compare the ranking given by these tests to ones based on posterior probabilites that $H_{g\ell}$ is true. We use receiver operating characteristic (ROC) curves to do this. ROC curves provide a graphical summary of the performance of a classifier by showing, as a function of the top H genes as ranked by the classifier, the proportion of cases correctly identified (true positive rate, TPR) along with the proportion of non-cases that are improperly identified (false positive rate, FPR). The top panel of Figure 3.4 shows a typical ROC curve among the ten we observed. The bottom panel shows the area under the ROC curve (AUC) up to a FPR of 0.1. For both plots we show results for each component of β (excluding the intercept) and a range of thresholds. In our simulations the BNP method was more accurate than limma on average in identifying genes with effects exceeding a threshold in absolute value.

3.4.1.2 Precision of estimated effects

We also compared the precision of point estimates obtained under the two models. For the BNP model, we take the posterior mean as our point estimate for $\beta_{g\ell}$. Here we expect BNP to do much better than limma because limma does not use a hierarchical model for shrinkage, whereas the BNP provides shrinkage toward the mass of the underlying distribution of the β_gs. Figure 3.5 shows that this is indeed the case; the mean squared prediction errors averaging across genes are uniformly smaller that those for limma.

3.4.2 Comparisons to count-based methods

Much of the recent methods proposed for the analysis of RNA-seq data are based on negative binomial generalized linear models. The goal of the following simulation study was to compare performance of the BNP method using the voom weights to popular negative binomial-based methods. To generate the count data, we first analyzed the Paschold et al. (2012) data with the voom-limma pipeline. Each simulated data set was generated by first sampling a random subset of 10,000 genes without replacement. Next, we simulated normal data, conditioning on the estimates obtained by limma for β_g, σ_g^2, as well as the precision weights, computed by the voom, corresponding to the selected genes. These simulated log-cpm values were then converted to log-counts, centering by $\log_2(\bar{R}.) - \log_2(10^6)$, where $\bar{R}.$ is the geometric mean of the estimated effective library sizes from the original data. Finally, these log-counts ($\log_2 r_{gn}$) were exponentiated and rounded to obtain the simulated counts, r_{gn}. For each simulated data set, we analyzed the data using our BNP method and also with two popular count-based methods, `edgeR` and `DESeq2`, the latter both with and without independent normal priors on the β_g parameters. (Robinson et al., 2010; Love et al., 2014). Just as in the previous study, we considered precision of point estimates and accuracy of gene classification. The steps described above are outlined below. For each simulated data set, a single MCMC chain was run for 50,000 iterations after 10,000 burn-in iterations. These were thinned to yield 2,000 samples for each gene-specific parameter and α. Geweke diagnostics for the gene-specific parameters showed slight departures from normality with a handful of genes having z-scores greater than 5 in absolute value. For every simulation 99% of genes for every component had an ESS greater than 400, but every simulation had at least one gene with an ESS less than 100 and one simulation had a parameter with an ESS less than 10.

1. Sample 10,000 indices, g, from $\{1, 2, \ldots, 36,000\}$ without replacement.
2. Set $\beta_g := \hat{\beta}_{voom,\ g}$, $\sigma_g^2 := \hat{\sigma}^2_{voom,\ g}$
3. Sample $y_{gn} \sim \mathrm{N}(x_n^\top \beta_g, \sigma_g^2 / w_{gn})$
4. Set $r_{gn} = \text{Round}\left\{2^{y_{gn} + \log_2(10^6) - \log_2(\bar{R}.)}\right\}$

3.4.2.1 Classification accuracy

The methodologies of Robinson et al. (2010) and Love et al. (2014) implemented in the `R` packages,`edgeR` and `DESeq2`, both provide threshold tests analogous to TREAT in `limma`. Love et al. (2014) provide the option of employing empirical Bayes priors to β_g; we show results both with (shrunk) and without (unshrunk). We again used ROC to compare the accuracy in gene ranking between these three methods. For this simulation, unlike the first, the true parameters for each gene are unique.

Comparing AUC across the ten simulations (Figure 3.6), we see that, in most cases, `DESeq2` provides better rankings without the empirical Bayes prior. The AUCs for the other three methods are comparable, with small differences depending on the effect type and the threshold. Overall, it appears that BNP and `DESeq2` (unshrunk) have an edge over edgeR and that BNP may have a slight advantage over the other methods when the threshold is set fairly high.

3.4.2.2 Precision of estimated effects

Plots of MSPE for the 10 simulations, shown in Figure 3.7, show that BNP compares favorably to the other methods, producing estimates that are closer to the truth, on average, that the count-based methods we tried. We expected that the empirical Bayes version of `DESeq2` would do better than the unshrunk version, since regularization of estimates typically reduces MSE. However, that was true only for the hybrid half-difference and the flow cell effect; for

Table 3.2: Definition of types of gene heterosis in terms of mean expression by genotype as well as using parameterization implied by the design matrix.

Type of heterosis	Relation	β-parameterization
high-parent B73xMo17	max{B73,Mo17} < B73xMo17	$\lvert\beta_{g2}\rvert < \beta_{g3} + \beta_{g4}$
low-parent B73xMo17	min{B73,Mo17} > B73xMo17	$-\lvert\beta_{g2}\rvert > \beta_{g3} + \beta_{g4}$
high-parent Mo17xB73	max{B73,Mo17} < Mo17xB73	$\lvert\beta_{g2}\rvert < \beta_{g3} - \beta_{g4}$
low-parent Mo17xB73	min{B73,Mo17} > Mo17xB73	$-\lvert\beta_{g2}\rvert > \beta_{g3} - \beta_{g4}$

the parental half-difference and the hybrid effect, the unshrunk `DESeq2` estimates were more accurate.

3.4.3 Calibration of posterior probabilities

Due to being fully Bayesian, the BNP method enjoys a further advantage over these other methods in that, by using posterior samples, we can evaluate probabilites of hypotheses with composite null parameter spaces which cannot be done using any of the other methods we tried. The ROC plots do suggest that the posterior probabilites do a good job with ranking the genes. However, because they are interpretable as probabilites, we can still question their accuracy relative to the frequency of true positives.

We now turn to four types of heterosis mentioned earlier in Section 3.2.1. These are defined in terms of our parameterization of the model matrix in Table 3.2.

Figure 3.8 shows estimated calibration curves for the 10 simulation studies for low- and high-parent heterosis for both of the two hybrid genotypes. Each curve was produced by binning genes by binning genes with similar posterior probabilites and plotting relative frequency of "true-positives" vs. the midpoint probability of the bin. Ideally, these curves would follow the identity line. We don't view the deviations that we see as seriously compromising our approach, but we note that there do appear to be some biases. In particular, the sideways "S" shape in the plot for low-parent B73×Mo17, suggets that the posterior probabilities tend to be pulled away from moderate values for this particular hypothesis.

3.5 Analysis of Paschold data

To fit the Paschold data, we ran 4 chains, each for 40,000 iterations after 30,000 iterations of warmup. The samples were thinned by a factor of 40, for a total of 160,000 post burn-in draws (4,000 thinned post burn-in draws) for each gene-specific parameter and α. Posterior probabilities of the four heterosis hypotheses as well as posterior first- and second-moments for the gene-specific parameters, β_g and σ_g, were computed using all the draws. Potential scale reduction factors were calculated for all of the gene-specific parameters and α. Rhats for all parameters were less than 1.1.

3.5.1 Comparison to original results

In Paschold et al. (2012), the results of a series of hypothesis tests were combined to classify genes into several categories characterizing an ordering or partial ordering of the low-parent, high-parent and hybrid expression levels (Paschold et al., 2012, p.2448). LPH and HPH each correspond to a union of 2 of these categories ($\{7, 8\}$ and $\{5, 6\}$, respectively.) In Figure 3.9, we show histograms of posterior probabilities pertaining to high- and ow-parent gene heterosis for the genes subsetted by whether or not they were classified as "discoveries" in the original paper. High-parent heterosis corresponds to genes for which the mean expression of the hybrid is higher than the more highly expressed parent and low-parent heterosis where it is lower than the less expressed parent. Generally speaking, the posterior probabilities tend to agree with the categorization — particularly for non-discoveries — however, a number of discoveries have low posterior probabilities in our analysis. This is likely due to borrowing of information across genes leading to shrinkage toward general patterns of expression, where both low- and high-parental expression were uncommon (Paschold et al. (2012) found evidence of these extreme patterns in less than 1% of genes.) We also note that a small number of non-discoveries are found to have high-posterior heterosis probabilities. Due to the multidimensional shrinkage featured by the model, it can be difficult to sort out what accounts for the difference in some cases.

Figure 3.14 shows how the nonparametric model shrinks estimates. Genes categorized as showing evidence of extreme gene heterosis in the original paper are represented as black points. The left panel shows independently obtained estimates for the parental half-difference and the mean hybrid effect overlayed on a bivariate histogram of the independent estimates for all genes. The right panel shows the corresponding estimates for our analysis overlayed on a posterior pointwise estimate of $\mathcal{P}$. In this second plot we can see significant shrinkage toward the regions where $\mathcal{P}$ has higher posterior density. This helps to explain why a large number of discovered genes have low posterior probabilities for the heterosis hypotheses: when weighed against the overall patterns within the data, in many cases it is likely that the magnitude of the independent estimates is overstated and that detection was a result of random variation.

A result of this shrinkage is increased posterior precision. Figure 3.10 shows 90% posterior credible intervals for the exemplar genes from Table 3.1. The credible intervals are fairly tight, and the posterior means fall outside the range of the data in some cases. This is a result of shrinkage due to the borrowing of information.

3.5.2 Comparison of heterosis results to parametric analysis

Landau (2016) undertook an analysis of the same data. They modeled the counts directly with a Poisson distributions, using a generalized linear model with log link and gene-specific linear predictors as well as a gene-specific variance parameter for a lognormal mixture on the means. They also assumed a parametric distribution for the gene-specific parameters. Their hierarchical model assumed independent normal priors on the coefficient components, $\beta_{g\ell}$. Table 3.3 shows a comparison of the posterior probabilities found by our method to Landau's binned into five intervals for each of the four types of extreme heterosis. While there is a fair amount of agreement between these two methods on the genes with highest and lowest probabilities of heterosis, the differences between the inferences are considerable.

From the marginal distributions of probabilities, we see that BNP finds lower chances of LPH across genes than Landau. For example, Landau found 1.6% of genes to have greater than 95% probability of low-parent heterosis for B73$\times$Mo17, but the majority of these were given less than 85% probability by BNP. However, BNP also found higher probabilities for some genes: More than 1/3 of the 0.7% genes which BNP found to be high probablity genes

Table 3.3: Two-way tables comparing the posterior probabilities of low-parent gene heterosis for all genes in the Paschold et al. (2012) data set found by BNP and Landau and Niemi (2016).

		Landau and Niemi (2016)					
		[0,0.05]	(0.05,0.15]	(0.15,0.85]	(0.85,0.95]	(0.95,1]	total
BNP	[0,0.05]	0.283	0.083	0.116	0.001	0.000	0.483
	(0.05,0.15]	0.017	0.037	0.129	0.003	0.001	0.186
	(0.15,0.85]	0.009	0.025	0.254	0.017	0.009	0.313
	(0.85,0.95]	0.000	0.000	0.007	0.002	0.002	0.011
	(0.95,1]	0.000	0.000	0.003	0.001	0.003	0.007
	total	0.308	0.145	0.508	0.022	0.016	1.000

Low-parent heterosis B73×Mo17.

		Landau and Niemi (2016)					
		[0,0.05]	(0.05,0.15]	(0.15,0.85]	(0.85,0.95]	(0.95,1]	total
BNP	[0,0.05]	0.296	0.090	0.135	0.003	0.001	0.523
	(0.05,0.15]	0.021	0.036	0.093	0.002	0.001	0.152
	(0.15,0.85]	0.010	0.031	0.237	0.011	0.005	0.295
	(0.85,0.95]	0.000	0.000	0.015	0.002	0.003	0.021
	(0.95,1]	0.000	0.000	0.003	0.002	0.005	0.010
	total	0.327	0.157	0.483	0.020	0.014	1.000

Low-parent heterosis Mo17×B73.

(>95%) were found to only have moderate probabilities (15%-85%). The story is different for HPH. Here, BNP has greater confidence, finding both more high- and low-probability genes than Landau.

While we cannot say from these comparisons whether one model is better than another, they serve to highlight the impact that distributional assumptions can have in detecting patterns of gene expression. Given our belief that models like that of Landau (2016) and Love et al. (2014) make unreasonable assumptions about the distribution of the gene-specific parameters, we expect that the BNP probabilites incorporate more information about the broader patterns of expression exhibited in the data.

3.5.3 Some practical considerations

On a different note, Landau (2016) found that when an independent normal model was assumed for β_{gl}, their posterior distributions were well approximated by a normal distribution

Table 3.4: Two-way tables comparing the posterior probabilities of high-parent heterosis for all genes in the Paschold et al. (2012) data set found by BNP and Landau and Niemi (2016).

		Landau and Niemi (2016)					
		[0,0.05]	(0.05,0.15]	(0.15,0.85]	(0.85,0.95]	(0.95,1]	total
BNP	[0,0.05]	0.395	0.126	0.155	0.000	0.000	0.676
	(0.05,0.15]	0.037	0.037	0.085	0.000	0.000	0.159
	(0.15,0.85]	0.020	0.025	0.100	0.003	0.001	0.149
	(0.85,0.95]	0.000	0.000	0.011	0.001	0.001	0.012
	(0.95,1]	0.000	0.000	0.003	0.001	0.001	0.005
	total	0.452	0.187	0.353	0.005	0.002	1.000

High-parent heterosis B73×Mo17.

		Landau and Niemi (2016)					
		[0,0.05]	(0.05,0.15]	(0.15,0.85]	(0.85,0.95]	(0.95,1]	total
BNP	[0,0.05]	0.352	0.115	0.165	0.000	0.000	0.633
	(0.05,0.15]	0.035	0.032	0.087	0.000	0.000	0.155
	(0.15,0.85]	0.033	0.035	0.116	0.005	0.002	0.190
	(0.85,0.95]	0.001	0.001	0.014	0.001	0.001	0.017
	(0.95,1]	0.000	0.000	0.003	0.001	0.001	0.005
	total	0.421	0.183	0.386	0.007	0.003	1.000

High-parent heterosis Mo17×B73.

matching the first two posterior moments. This is convenient because computation of these moments requires minimal storage, as running means can be used to update these after each iteration of the MCMC chain.

Figure 3.11 shows full posterior distributions and normal approximations for each component of β_g for four randomly selected genes. The quality of the moment-based approximation is satisfactory in some cases but not in others. Due to the discrepencies displayed here, we would recommend basing inference on the full posterior, rather than a normal approximation. The traceplots for these parameters (Figure 3.12) show no indication of a lack of convergence. For reference, the data and estimated mean expression levels are displayed in Figure 3.13.

3.6 Discussion

Hierarchical models have an important role in problems where, as is true in gene expression, the number of model parameters is much larger than the number of subjects. By modeling the parameters, we use the "between-gene" information in the data to moderate our inferences about individual genes . Where some methodolgies employ empirical Bayes concepts for certain procedures, hierarchical models are not usually an explicit part of the model. Empirical evidence suggests that, with regard to the gene-specific parameters, default model assumptions such as independent normals, are not realistic — the empirical distributions exhibit irregular shapes, multimodality and correlations. Since it seems difficult to suggest a parametric alternative to the common default assumptions (independence, normality), a one-size-fits-all nonparametric model seemed worth investigating. We have demonstrated a computationally demanding, yet feasible, solution in our BNP method.

Our simulation studies show that the BNP methodology provides a viable alternative to more standard methods. Intuitively, we expect that it should do better than the alternative methods we considered in cases when the distribution of gene-effects has an unusual shape that contradicts independence. We can imagine situations where the underlying distribution of the gene specific parameters is itself an object of importance. For example, the v-shape displayed in Figure 3.14 describes a pattern whereby the average hybrid mean expression tends toward but generally remains less than the mean expression of the higher parent. We might interpret this to be a result of dominance, where a dominant allele masks the effect of the recessesive allele in the heterozygous maize plant.

When there is only interest in expression at the level of genes, it should still be valuable to have a model flexible enough to handle detectable patterns in the data. Figure 3.15 shows side-by-side plots of the distribution of the estimated components by BNP and `DESeq2`, which assumes independent priors on the gene-specific effects (excluding the intercept). The BNP estimates are shrunk toward the mass of the distribution, whereas `DESeq2` estimates are shrunk toward the origin.

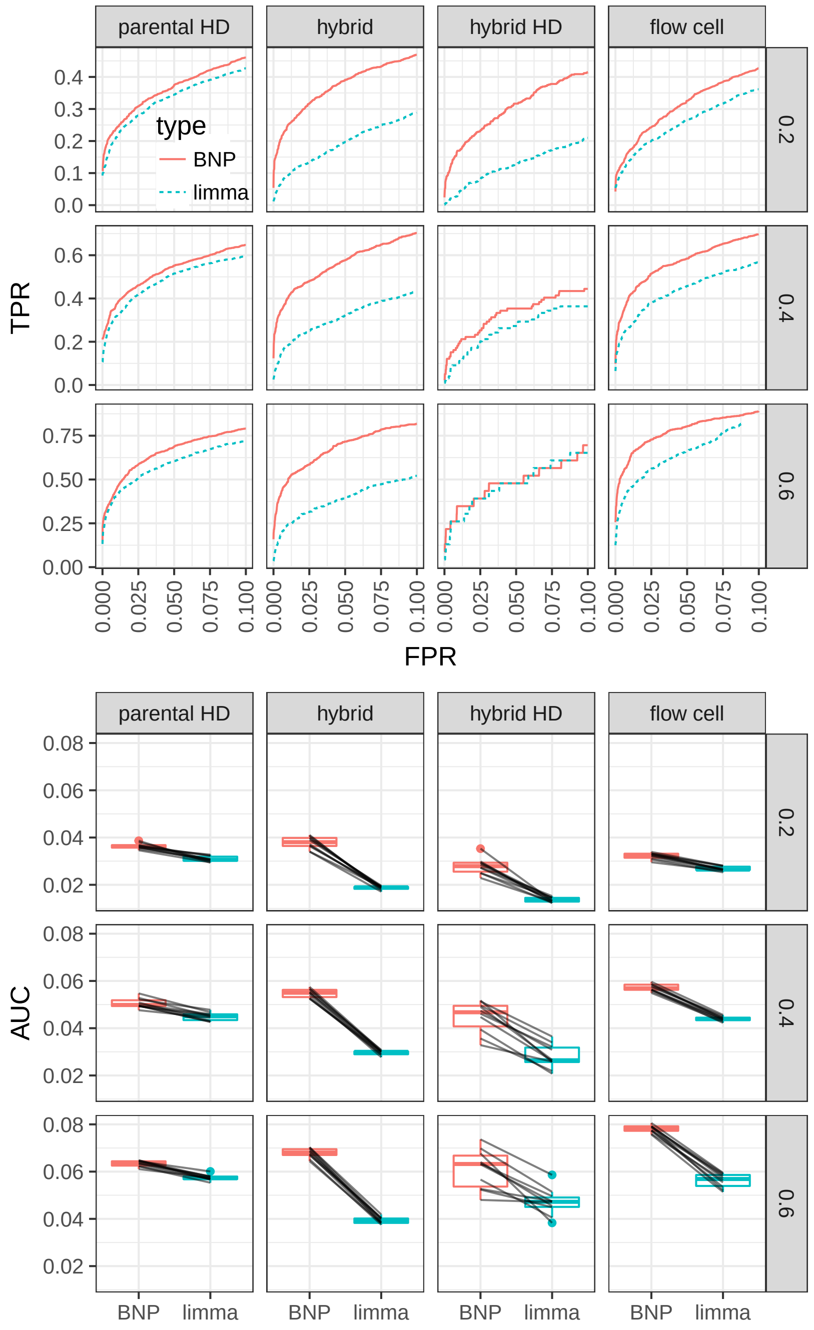


Figure 3.4: Comparison of gene classification accuracy for BNP and non-hierarchical log-linear model. Top: ROC curves for a typical simulated data set for classification of genes by whether or not a particular effect has a log-fold-change greater than a threshold. Bottom: Boxplots of AUC for the same tests for all ten simulations, up to a false discovery rate of 0.1. The black lines identify particular simulated data sets.

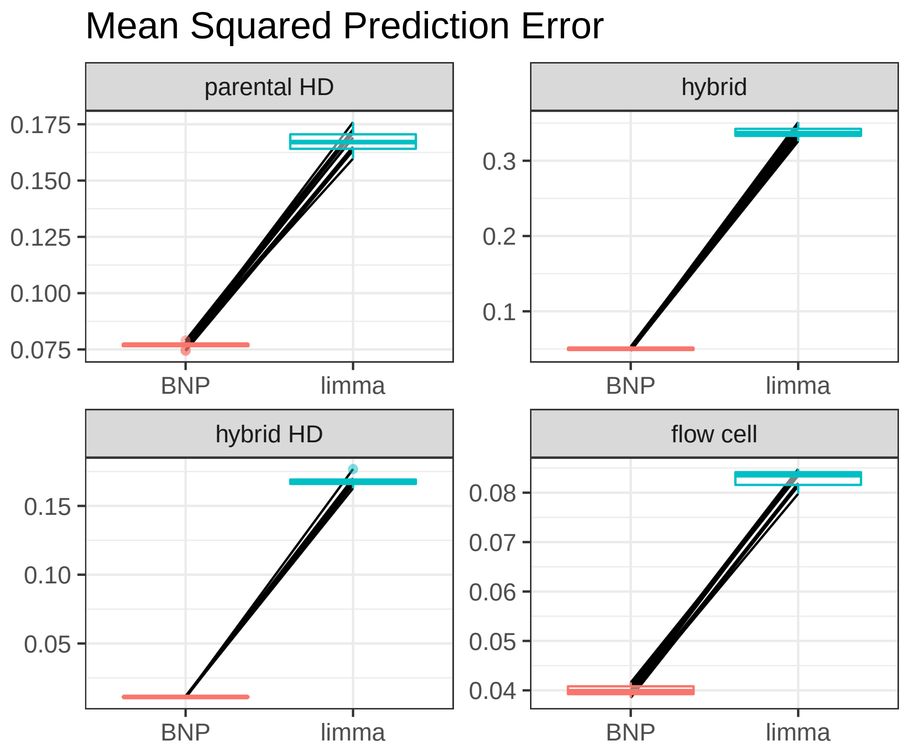


Figure 3.5: Comparison of precision of coefficient estimation for BNP and non-hierarchical log-linear model. Mean squared prediction error for $\beta_{g\ell}$ averaging over g.

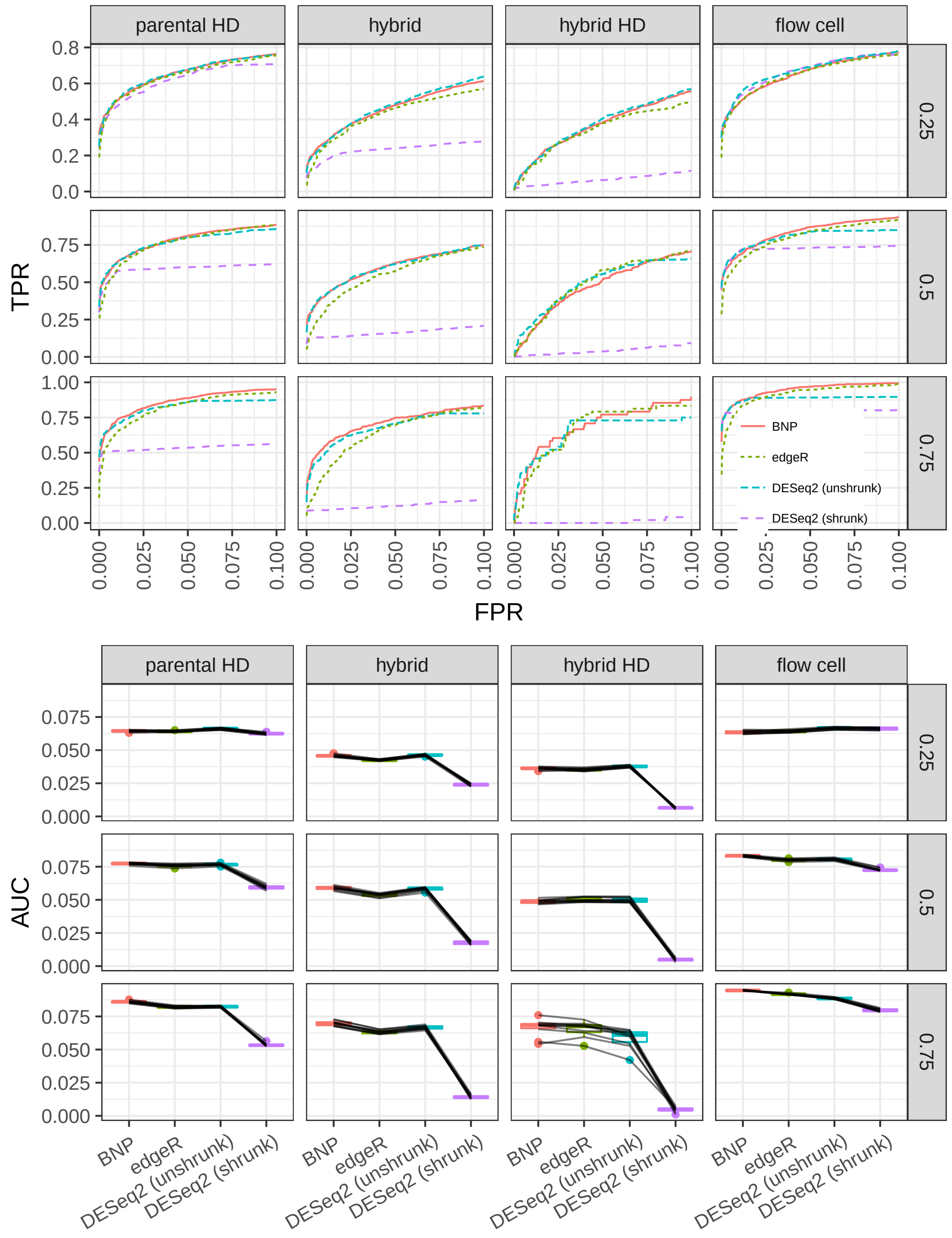


Figure 3.6: Comparison of gene classification accuracy for BNP using voom weights and some popular count-based methods. Top: ROC curves for a typical simulated data set for classification of genes by whether or not a particular effect has a log-fold-change greater than a threshold. Bottom: Boxplots of AUC for the same tests for all ten simulations, up to a false discovery rate of 0.1. The black lines identify particular simulated data sets.

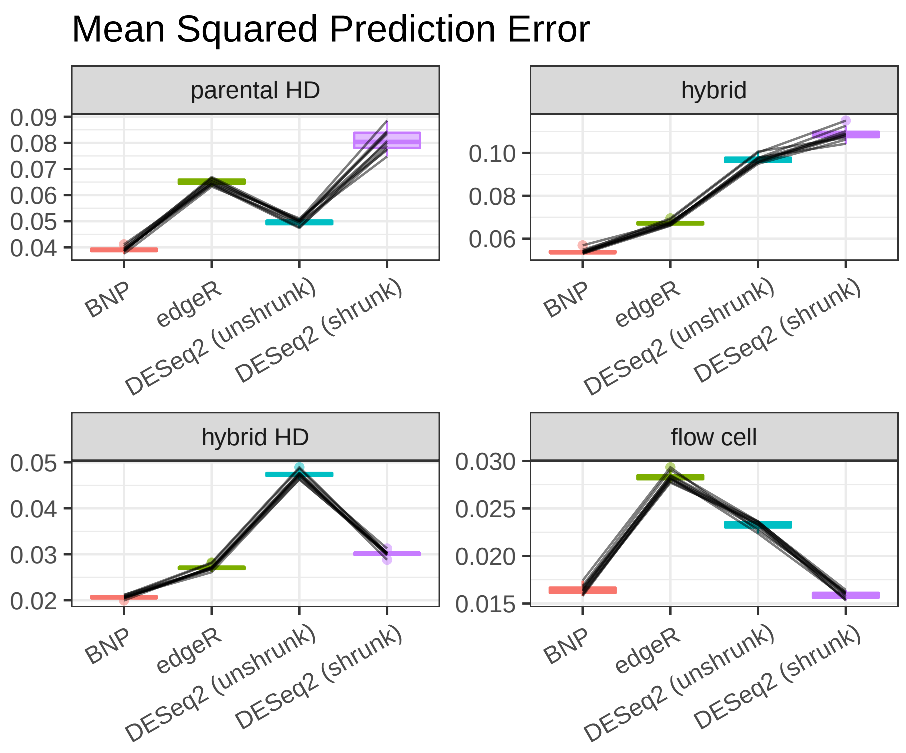


Figure 3.7: Comparison of precision of coefficient estimation for BNP and non-hierarchical log-linear model. Mean squared prediction error for $\beta_{g\ell}$ averaging over g.

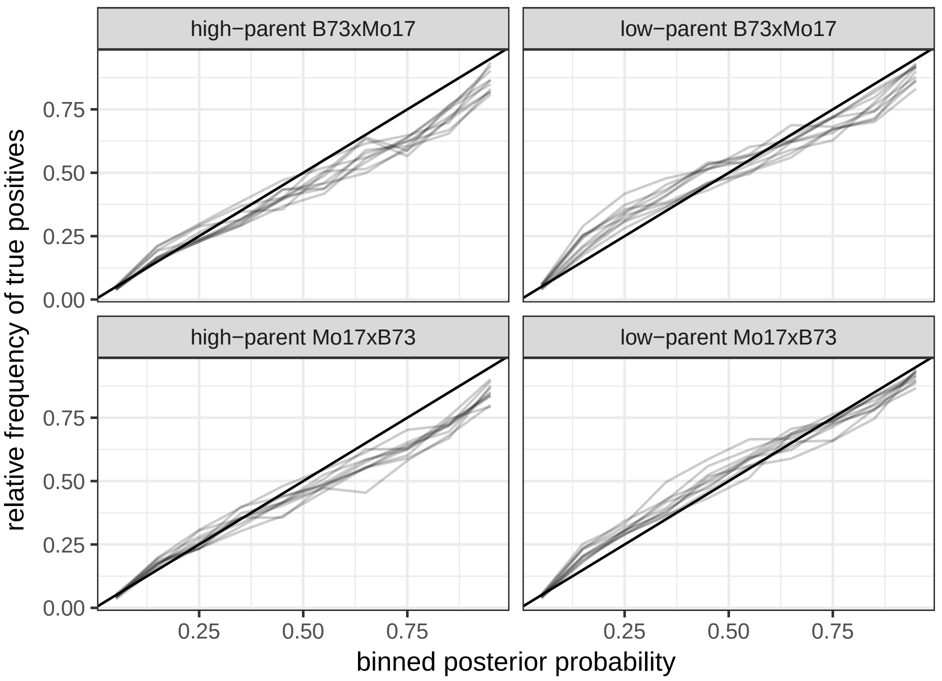


Figure 3.8: Calibration curves for BNP posterior probabilities calculated using posteriors and truth from the count-based simulations. Curves show true relative frequencies for the hypothesis indicated with bins determined by the posterior probability for each gene. The midpoint of the interval was used for plotting.

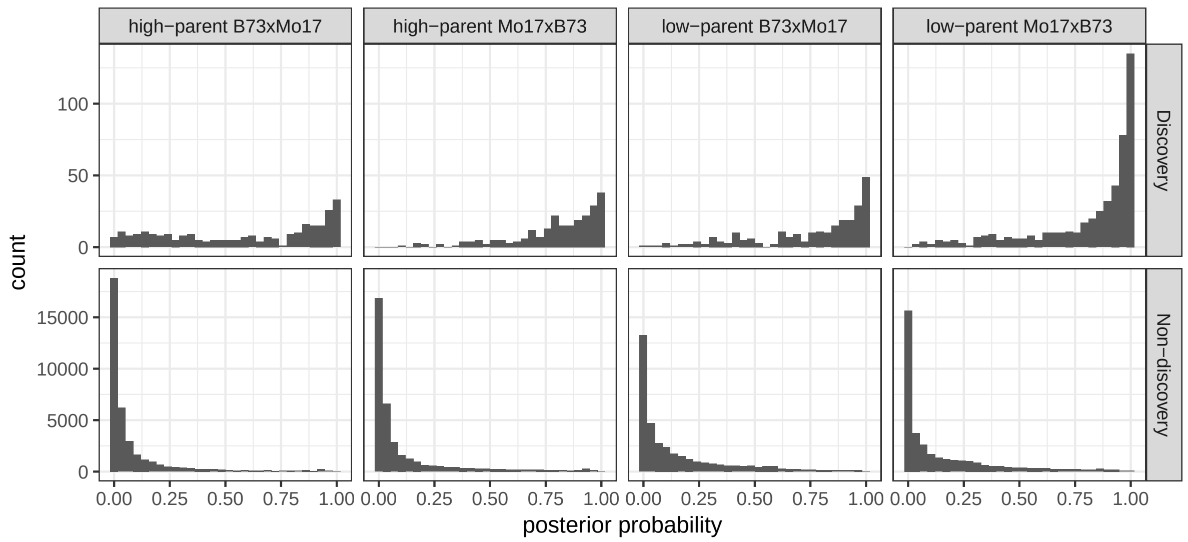


Figure 3.9: Posterior probabilities for low- and high-parent gene heterosis hypotheses for subsets of genes classified as discoveries and non-discoveries in the original paper (Paschold et al. 2012).

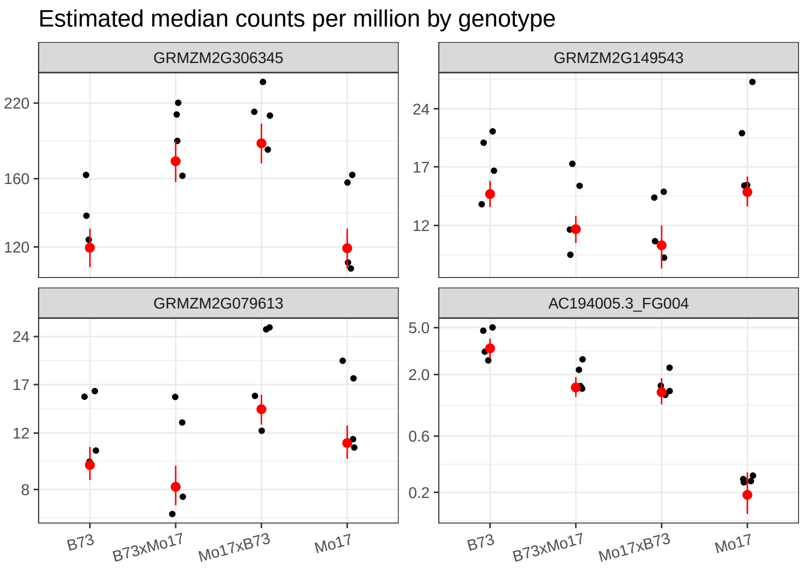


Figure 3.10: Plots of the exemplar genes from Table 3.1, illustrating presence and absence of patterns of heterosis. The points are jittered horizontally. The uncertainty intervals shown are 90% credible intervals for the mean log-cpm, scaled to the counts per million scale.

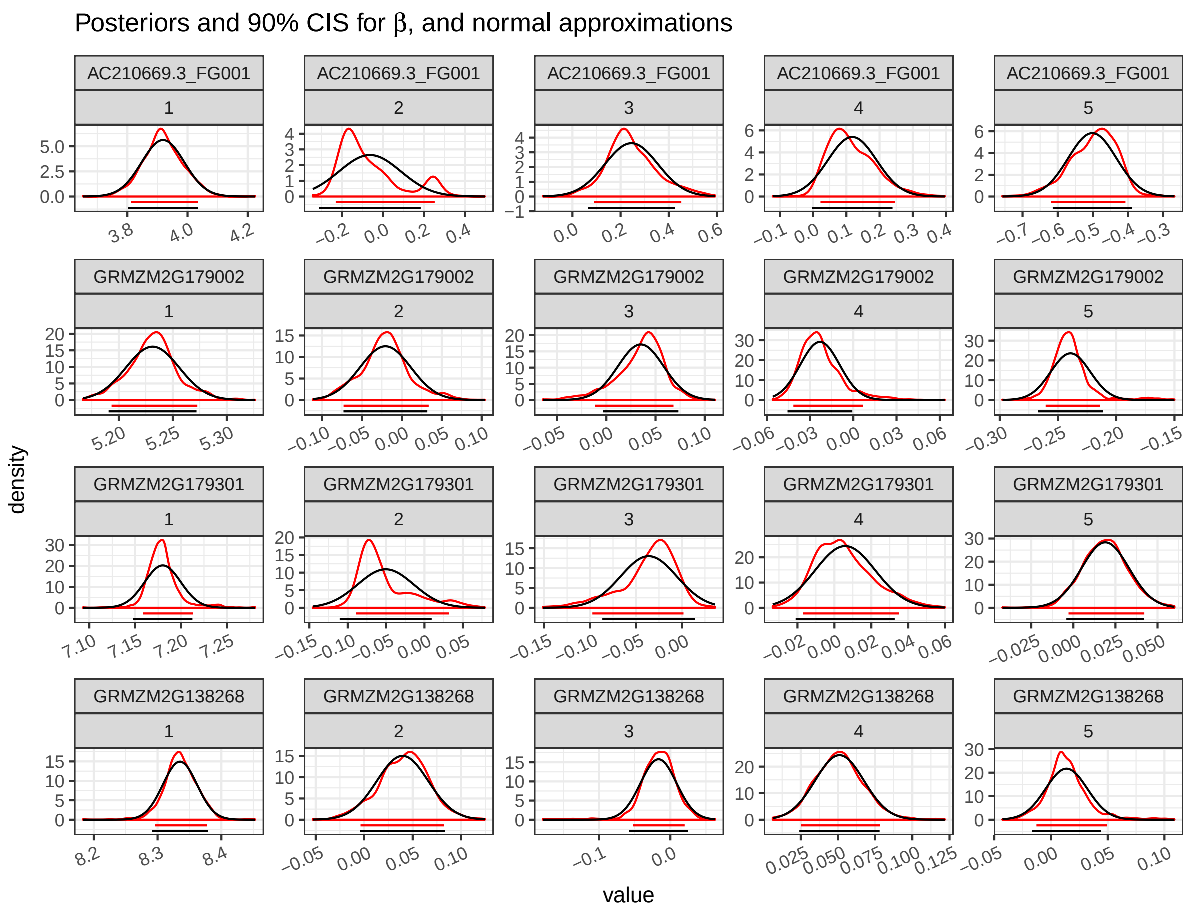


Figure 3.11: Posterior distributions of β_g for 4 randomly selected genes (red). Normal approximations based on the first two posterior moments are also shown (black). Line segments representing 90% credible intervals are drawn in the lower part of each plot.

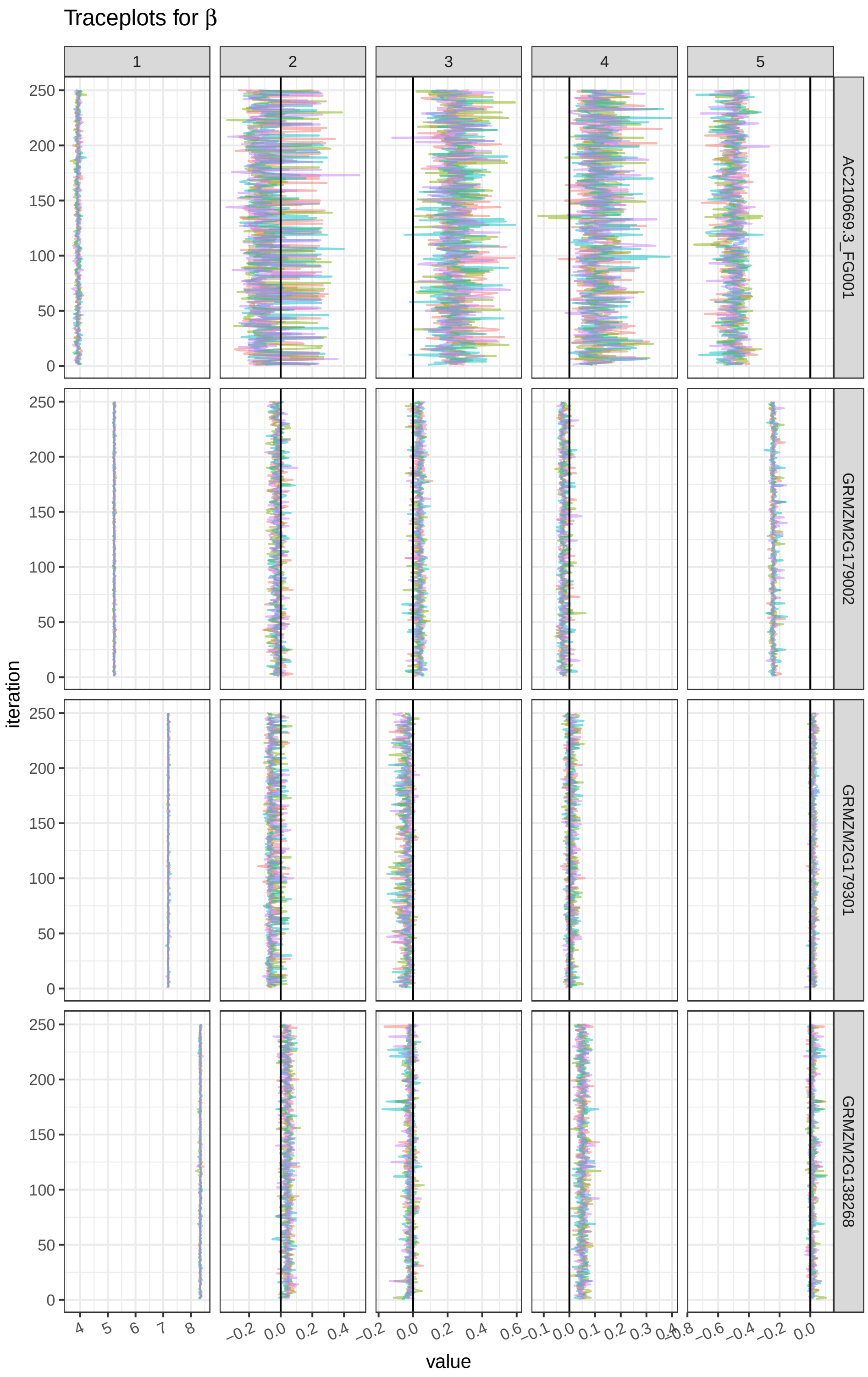


Figure 3.12: Traceplots of β for randomly selected genes. 250 thinned samples for each independent Markov chain, the chain identified by color, are shown.

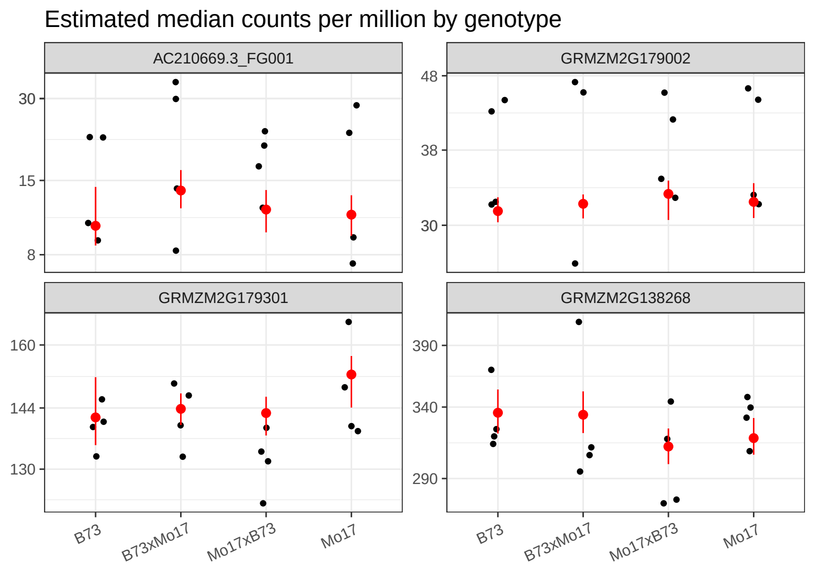


Figure 3.13: Adjusted log-cpm for randomly selected genes by genotype (black) and corresponding posterior means and 90% confidence intervals for the mean log-cpm expression, transformed to the counts per million scale (red).

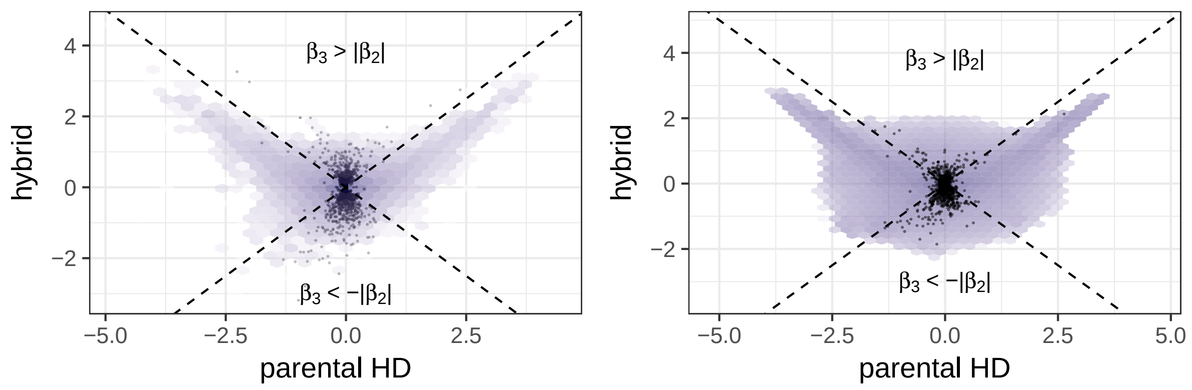


Figure 3.14: The left panel shows independently obtained estimates for the parental half-difference and the mean hybrid effect for genes classified as heterotic in the maize study. These are overlayed on a bivariate histogram of the independent estimates for all genes. The right panel shows the corresponding estimates for our analysis overlayed on a posterior pointwise estimate of $\mathcal{P}$. In this second plot we can see significant shrinkage toward the regions where $\mathcal{P}$ puts more probability in the posterior.

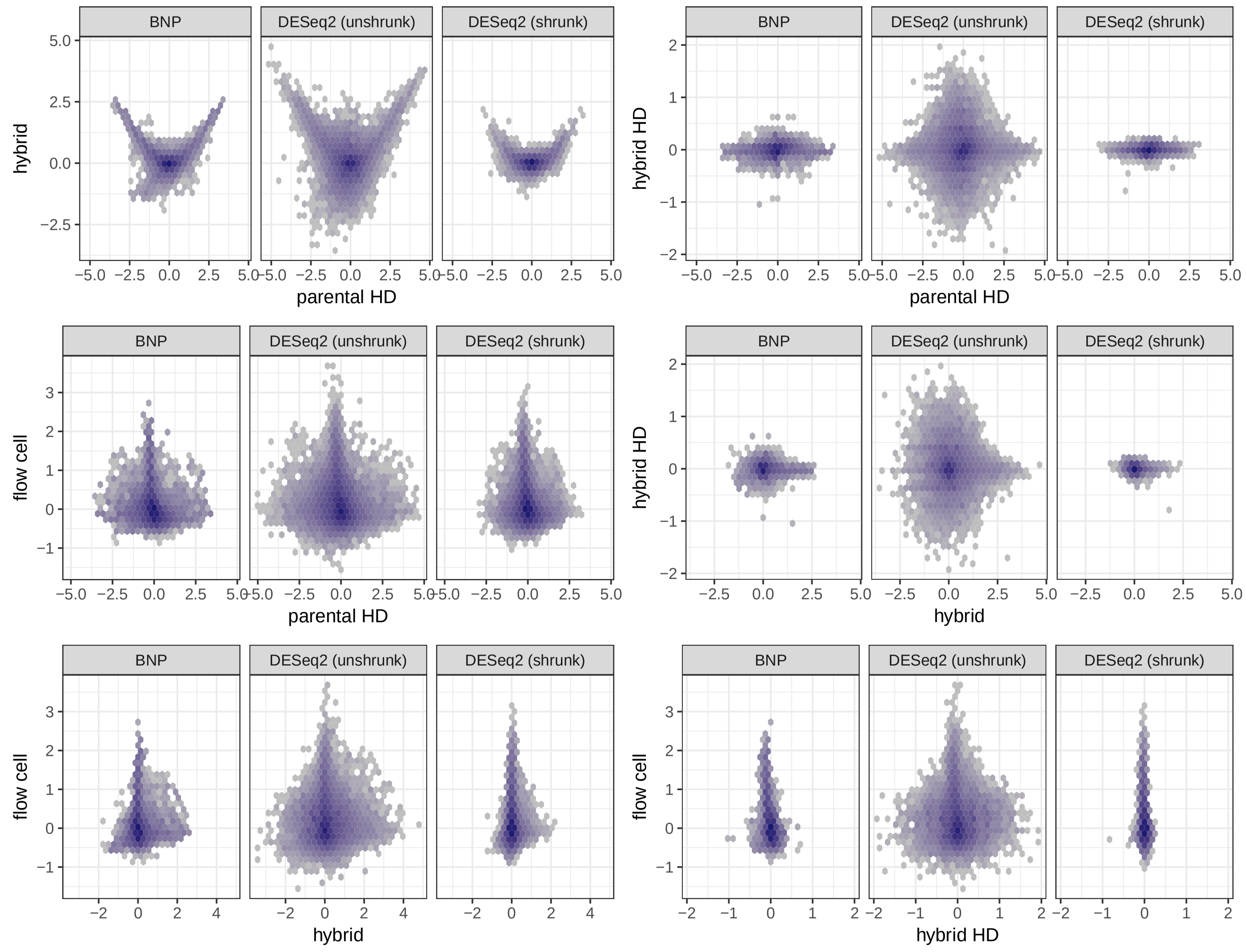


Figure 3.15: Bivariate histograms of estimates of gene-specific effects, excluding intercepts. The estimates shown are posterior means obtained with the BNP method as well as estimates obtained with DESeq2 with and without independent normal shrinkage priors (MAP and MLE, respectively) are shown.

Additionally, while posterior sampling is cumbersome in the amount of memory involved (approximately 6 GB for 4,000 samples of β), the interpretability of posterior quantities of interest and the inferential flexibility allowed make a strong case for the use of the Bayesian approach.

There are various modifications that might be considered to our model, within the same graphical structure. In the BNP method, precision weighting was used to correct for inaccuracies in mean-variance relationship implied by a normal assumption on the normalized log-counts. We would prefer not to have our inference conditional on these pre-computed estimates. An alternative approach would be to adopt an over-dispersed count model, such as the negative binomial and model the counts directly. Our reason for not doing this in the first place is that by proper choice of base measure, we get conditionally conjugate draws for $\tilde{\beta}_k$ and $\tilde{\sigma}_k^2$ that depend only on simple linear combinations of low-dimensional summaries of the counts. In contrast, the log-likelihood for the negative binomial involves quantities such as $\sum_n \log \Gamma(y_{gn} + \phi)$, for real-valued overdispersion parameter ϕ, which is not linear in the parameter. Despite such nuisances, the overall structure of our Gibbs sampler would remain unchanged.

Another modification that we might consider is the choice of prior for the stick-breaking weights. The implication of the Beta$(1, \alpha)$ priors is that the weights decay exponentially on average. A generalization described in Ishwaran and James (2001) is $\nu \sim \text{Beta}(1 - a, b + ak)$, which the authors refer to as a Pitman-Yor process, contains the Dirichlet process as a special case ($a = 0, b = \alpha$). Other restrictions can selected to represent alternative prior assumptions. For example, one can select $a = \alpha, b = 0$ which implies that the weights follow a power law. We might expect that this would lead to more significant weights and thus less aggressive shrinkage that the Dirichlet process.

CHAPTER 4. A hierarchical failure-time model for observational data exhibiting infant-mortality and wearout failure modes

4.1 Introduction

Consumers have an interest in accurate assessment of product reliability. Toward this end, there is a need for models that can accomodate common failure patterns and methods which can make the most of available data and properly account for uncertainty. Doing these things well enables good decision-making in matters related to product life, including purchase decisions and contingency planning.

The reliability of many engineered products follows a similar pattern. Relatively high rates of failure occurring early ("infant mortality") is due to manufacturing defects. After this "burn-in" period, failure rates stabilize after the majority of defective units have failed. Finally, after prolonged use, rates of failure increase due to wearout. Ignoring this pattern in failure rates can lead to spurious inferences about a product's reliability and suboptimal decisions. This highlights the importance of choosing a sufficiently flexible model.

This paper presents a general framework for statistically modeling reliability field data from a heterogeneous population of components or subsystems operating within a larger population or fleet of systems. Such a model would be useful for applications such as

- Making purchase decisions from among different suppliers of the components or subsystems
- Making predictions for the number of needed replacement components or subsystems for spare part provisioning

Nonparametric methods require relatively few assumptions, but may not be suitable for prediction when extrapolation is required. Moreover, nonparametric estimates made using small samples, or data which are heavily censored may be too noisy to be useful. Appropriate parametric models can make better use of available data. For inference, likelihood-based methods are able to deal with truncated and/or censored data. Furthermore, by assuming a parsimonious representation of the data-generating process, parametric models admit simpler, more intuitive ways to compare populations of interest, borrow strength across groups, and incorporate prior information.

Often no information is available to identify why a unit failed, even when the failed units are available. Physical failure mode analysis (sometimes referred to as "autopsy") can be extremely time consuming and expensive. This presents a dilemma for analysts: the data may indicate multiple failure modes, contraindicating the use of a unimodal parametric distribution. However, without any knowledge of cause of failure, traditional competing risk models are not identifiable. Chan and Meeker (1999) proposed the Generalized Limited Failure Population (GLFP) model. This model can provide a solution to the problem of unknown cause of failure in the special case where there is evidence of two modes of failure which impact different stages of product life. It avoids non-identifiability by introducing a parameter representing the population proportion defective. When this parameter is zero, the GLFP model reduces to a unimodal distribution.

The GLFP model has a meaningful parameterization and accommodates lifetime data with both infant mortality and wearout. Unfortunately, it can require a lot of data to fit due to the model's complexity. When multiple sources of information are available, partial pooling, accomplished via hierarchical modeling, can reduce the amount of data required to produce stable parameter estimates. When multiple populations are of interest, comparisons based on separate, unrestricted GLFP model fits will be limited to those products with sufficient data. As we will show, hierarchical modeling of the GLFP parameters allows for borrowing of information across populations which imposes "soft" constraints on model parameters. This enables estimation of the lifetime distribution for all populations via shrinkage toward a "pooled" model, with the degree of shrinkage inversely related to the amount of population specific information.

We demonstrate computation of various quantities of interest while comparing reliability across populations using numerical integration over the posterior distribution with samples obtained via Markov Chain Monte Carlo (MCMC).

The structure of the paper is as follows. Section 4.2 introduces the proposed method and potential applications and examines previous related work. Section 4.3 introduces the GLFP model and illustrates its use with a subset of hard drive failure data from the company, Backblaze. In Section 4.4, we describe the complete Backblaze Hard Drive data set, which is used throughout as a motivating example. In Section 4.5 we discuss using hierarchical modeling of the GLFP model parameters to extend the GLFP model to multiple populations. Section 4.6 applies the hierarchical GLFP model to the hard drive data. First, we do model selection among four models of increasing complexity, using approximate leave-one-out cross-validation. Next, for the selected model, we assess the model fit using a graphical comparison of replicated data sets generated from the posterior distribution to the observed data. Finally, we present an evaluative comparison of populations taking into account practical considerations related to product lifetime. Section 4.7 describes potential applications of our model and limitations for inference implied by the model assumptions.

4.2 Background

In engineering applications, a product can often fail from one out of a set of possible malfunctioning components. For example, a computer system can break if the mother board, disc drive or power supply stop working. Circuit boards (CB) can fail due to a manufacturing defect or later as a result of wearout of certain components. The general name for such products is a series system where the lifetime of the product is the minimum failure time across different components or risks (Nelson, 1982, Chapter 5). A common assumption in series systems is the time to failure for each risk is statistically independent. Thus, the overall reliability of a unit is the product of all the risks. Parameter estimation is straightforward if the cause of failure is known for each observation. However, in many situations the cause of failure is unknown to the analyst. This is referred to in the literature as masking.

Previous papers have employed various assumptions and methods to model masked lifetime failure data. When modeling computer system failures Reiser et al. (1995) assumed each observed failure came from a known subset of failure modes, and estimation was performed using a Bayesian approach. Chan and Meeker (1999) labeled the cause of circuit board failures as infant mortality, unknown, or wearout based on the time of observed failures. This helped identify parameters when using maximum likelihood (ML) estimation. Extending Chan and Meeker's analysis, Basu et al. (2003) performed a Bayesian analysis with informative priors to better identify early versus late failure modes without making any assumptions about the cause of failure. Berger and Sun (1993) introduced the Poly-Weibull distribution where the failure time is the minimum of a several Weibull distributions, corresponding to different failure modes. Ranjan et al. (2015) considered a competing risk model for infant mortality and wearout as a mixture of Weibull and exponential failure distributions. Treating the unknown failure modes as incomplete data, an expectation maximization (EM) algorithm providing ML estimates was used, in addition to Bayesian estimation.

4.3 Generalized Limited Failure Population Model

4.3.1 Weibull parameterization

The Weibull cumulative distribution function (cdf) is

$$\Pr(T \le t|\alpha, \beta) = F(t|\alpha, \beta) = 1 - \exp\left[-\left(\frac{t}{\alpha}\right)^{\beta}\right], \quad t > 0, \tag{4.1}$$

where $\beta > 0$ is the Weibull shape parameter and $\alpha > 0$ is a scale parameter. Because $\log(T)$ has a smallest extreme value distribution (a member of the location-scale family of distributions), the Weibull cdf can also be written as

$$\Pr(T \le t|\mu, \sigma) = F(t|\mu, \sigma) = \Phi_{SEV}\left[\frac{\log(t) - \mu}{\sigma}\right], \quad t > 0,$$

where $\Phi_{SEV}(z) = 1 - \exp[-\exp(z)]$ is the standard smallest extreme value distribution cdf and $\mu = \log(\alpha)$ and $\sigma = 1/\beta$ are, respectively, location and scale parameters for the distribution of $\log(T)$. Therefore, the Weibull distribution is a member of the log-location-scale family of distributions.

We will use an alternative parameterization where α is replaced by the p quantile $t_p = \alpha\,[-\log(1-p)]^{\sigma}$. Replacing α in (1) with the $\alpha = t_p/[-\log(1-p)]^{\sigma}$ gives

$$\begin{aligned}\Pr(T \le t|t_p, \sigma) = F(t|t_p, \sigma) &= 1 - \exp\left[-\left(\frac{t}{t_p/[-\log(1-p)]^{\sigma}}\right)^{1/\sigma}\right] \\ &= 1 - \exp\left[\log(1-p)\left(\frac{t}{t_p}\right)^{1/\sigma}\right], \quad t > 0.\end{aligned}$$

There are two important reasons for using this parameterization.

- Especially with a high-reliability product, it will be easier to elicit prior information about a quantile (t_p) in the lower tail of the distribution than it will be to elicit prior information about α. In addition, there is generally available information about the shape parameter, σ, for a given failure mechanism (e.g., if the failure is due to a wearout mechanism, then it is known that $\sigma < 1$).

- Because of heavy censoring in reliability field data, the parameter estimates of the μ and σ parameters will generally be highly correlated and thus specification of independent marginal prior distributions would be be inappropriate. On the other hand, estimates of t_p and σ, for some appropriately chosen value of p, will be approximately independent, allowing the easier elicitation and specification of independent marginal prior distributions. For example, if a typical field data set has 10% of units failing, then choosing $t_{0.05}$ would work well.

4.3.2 GLFP model

Let F_1, F_2 be Weibull distributions with parameters (t_{p1}, σ_1) and (t_{p2}, σ_2), respectively. The Generalized Limited Failure Population model (GLFP) of Chan and Meeker (1999) is defined as follows: Let $T \sim \text{GLFP}(\pi, t_{p1}, \sigma_1, t_{p2}, \sigma_2)$. Then

$$\Pr(T \le t) = H(t) = 1 - (1 - \pi\, F_1(t))(1 - F_2(t)),\ t > 0,\ 0 < \pi < 1.$$

The GLFP model can be understood as a mixture model with a binary latent variable, $\delta_i \overset{ind.}{\sim}$ Bernoulli(π). δ_i is an indicator for whether or not unit i is defective. Expressed conditional on δ_i,

$$\begin{aligned} P(T \le t|\delta_i = 1) =& 1 - (1 - F_1(t))(1 - F_2(t)) \\ P(T \le t|\delta_i = 0) =& F_2(t). \end{aligned}$$

The parameter π represents the proportion of units susceptible to early failure, hence susceptible to both failure modes. Here the cause of failure is not assumed to be known, thus units from the same population are exchangeable.

Taking the derivative of the (marginal) cdf, the density for the GLFP is

$$\begin{aligned} h(t|\pi, t_{p1}, \sigma_1, t_{p2}, \sigma_2) = \pi f_1(t|t_{p1}, \sigma_1)\left(1 - F_2(t|t_{p2}, \sigma_2)\right) + \\ f_2(t|t_{p2}, \sigma_2)\left(1 - \pi F_1(t|t_{p1}, \sigma_1)\right). \end{aligned}$$

4.3.3 Censoring and truncation

A common feature of lifetime data is right-censoring. In the analysis of reliability field data, it is rare all units are observed until failure. If a unit has not yet failed when the data are analyzed it is considered right-censored. In other words, right-censoring puts a lower bound on the failure time.

When an observation is left-truncated, it would not have been observed if it had failed prior to a particular time, which we refer to as the left-truncation time. Left-truncation is a common feature of observational lifetime data, where the factors leading to inclusion in the data set are uncontrolled and/or the population of interest has a history prior to any data collection. Ignoring left truncation can lead to biased estimates. However, dropping left truncated observations should be avoided because it could substantially reduce the total available information. Therefore, we incorporate both right censoring and left truncation into the likelihood.

4.3.3.1 Likelihood

We now give the general form for the likelihood function, taking into account left truncation and right censoring. Let t_i denote the end of the observed lifetime of unit i, in hours. Let t_i^L be the left truncation time, the age of unit i when data reporting commenced. Additionally, let c_i be an indicator for censoring; $c_i = 1$ if the failure time is right-censored, $c_1 = 0$ if the unit failed (at time t_i). The likelihood for the GLFP is a function of the parameters $\boldsymbol{\theta} = (\pi, t_{p1}, \sigma_1, t_{p2}, \sigma_2)$. Assuming the lifetimes of all units are independent, the likelihood for the data, $t_1, \ldots, t_n$ is

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} \left[\frac{h(t_i; \boldsymbol{\theta})}{1 - H(t_i^L; \boldsymbol{\theta})} \right]^{1-c_i} \left[\frac{1 - H(t_i; \boldsymbol{\theta})}{1 - H(t_i^L; \boldsymbol{\theta})} \right]^{c_i}$$

4.4 Motivating Example

4.4.1 Backblaze hard drive data

Backblaze is a company that offers cloud backup storage to protect against data loss. Since 2013 it has been collecting data on hard drives operating at its facility. The purpose is to provide consumers and businesses with reliability information on different hard drive-models. The hard drives continuously spin in controlled storage pods. Drives are run until failure. When a hard drive fails it is removed and replaced. In addition, the number of storage pods is increasing as Backblaze adds drives to its storage capacity. Every quarter Backblaze makes its hard drive data publicly available through their website (`https://www.backblaze.com/b2/hard-drive-test-data.html`, Accessed April 1, 2016). In addition, Backblaze publishes summary statistics of the different drive-models currently operating. No other analysis or modeling of the failure data is provided. Backblaze does, however, encourage others to further analyze their data.

As of the first quarter of 2016, Backblaze was collecting and reporting data on 63 different drive-models. Some drive-models have been running since 2013 or before, while others were added at a later date. Data have been reported on 75,297 different hard drives that are or were in operation. The number of drives varies by drive-model; some drive-models only have a service record for a single drive whereas the maximum number of daily service records for a single drive-model is 35,860. Figure 1 shows a scatterplot of the total observed running time in hundred of thousands hours versus the total number of failures for drive-models with at least 3 failures. For model identification, a minimum of three failures was the criterion for hard drive brands to be included in our analysis.

4.4.2 Analysis of a single drive-model

To illustrate the GLFP model, we will present an analysis of the drive-model with the most observed failures, Drive-model 14. The Kaplan-Meier (K-M) estimate plotted below suggests at least two failure modes. The curve has a slow ramp up in failures till about 10,000 hours (early failures), and increases more rapidly from about 10,000 to 20,000 hours (presumably wearout failures). In addition, computer hard drives are known to have a mixture of early and late failures (Chan and Meeker, 1999). Therefore, there is both an empirical and a theoretical justification to apply the GLFP model.

To estimate the parameters of the GLFP model we use a Bayesian approach, selecting proper prior distributions to improve identification of the model parameters, which also ensures a proper posterior distribution. We reparameterize the component Weibull distributions using the 0.50 quantile for the infant mortality failure mode ($t_{0.5,1}$) and the 0.20 quantile for the wearout failure mode ($t_{0.2,2}$).

When eliciting prior distribution information it is much easier to ask about recognizable characteristics of a distribution instead of the parameters of the distribution. Following the approach used in (Meeker et al., 2017, Section 15.2.2) we will use diagonal braces ($\langle ., . \rangle$) to refer to 95 percent central probability intervals, rather than the standard model parameters when specifying prior distributions. Using this convention the prior distributions used in our analyses are

$$t_{0.5,1} \sim \text{Log-normal}\langle 1.7, 7.6 \times 10^6 \rangle, \quad t_{0.2,2} \sim \text{Log-normal}\langle 8.6, 5.6 \times 10^7 \rangle$$

$$\pi \sim \text{Logit-normal}\langle 1.0 \times 10^{-3}, 7.1 \times 10^{-1} \rangle, \quad \sigma_1, \sigma_2 \overset{ind.}{\sim} \text{Log-normal}\langle 7.4 \times 10^{-3}, 1.3 \times 10^2 \rangle.$$

These prior distributions put probability mass on a wide range of values for all model parameters — much larger than we would expect for a typical Weibull distribution. Thus, we consider these prior distributions to be relatively uninformative.

Table 4.1 gives the posterior median and 95% credible intervals for the 5 GLFP parameters for Drive-model 14. The parameter π is an estimate of the proportion of drives susceptible to infant mortality. As expected, this proportion is small with a median value of 0.05. The shape parameter estimates for the two Weibull distributions also match intuition. The infant mortality failure mode puts posterior probability on a values of β_1 less than 1, which corresponds to a decreasing hazard. Conversely, the credible interval for β_2, the wearout mode, is strictly above 1, implying an increasing hazard function. The two quantiles are also well identified with the infant mortality failure mode having an much earlier time to reach the 0.50 quantile compared to the time to reach the 0.20 quantile for the wearout mode.

Probability plotting is a simple method to assess and compare the adequacy of members of the log-location-scale family of distributions. After properly transforming the axes of a plot, graphing an empirical estimate of fraction failing as a function of time along with pointwise confidence bands provides a visual check for distributional goodness of fit. We applied this method using the Kaplan-Meier nonparametric estimate of the empirical cdf. With left truncation, however, the standard Kaplan-Meier estimator is biased, so we used an adjusted version due to **?**)Chapter 11]meeker (see Appendix for a detailed description).

In Figure 4.3, we overlay the posterior median of the fitted GLFP model onto the adjusted K-M estimate with axes on the Weibull probability scale. The plot also contains 90% pointwise uncertainty bands associated with each estimate. While the Weibull model is inadequate for these data, the GLFP model fits quite well, as it is able to adequately describe the rapid increase in the empirical cdf between 8000 and 20,000 hours.

4.5 Hierarchical GLFP model

In order to describe the entire population, consisting of different but similar sub-populations, and because of the limited amount of data from many of the sub-populations, we model the sub-population-specific parameters hierarchically, borrowing strength across the sub-populations. Let

$$y_{ig} \overset{ind.}{\sim} \text{GLFP}\left(\pi_g, t_{p_1 g}, \sigma_{1g}, t_{p_2 g}, \sigma_{2g}\right), \tag{4.2}$$

where $g = 1, \ldots, G$ indexes the sub-populations. The likelihood for the hierarchical GLFP is a function of the sets of parameters $\boldsymbol{\theta_g} = (\pi_g, t_{p_1 g}, \sigma_{1g}, t_{p_2 g}, \sigma_{2g})$, one set for each group, g. Assuming the lifetimes of all units are independent within and across groups, the likelihood for the data, $i = 1, \ldots, n_g$, is given by

$$L(\boldsymbol{\Theta}) = \prod_{g=1}^{G} \prod_{i=1}^{n_g} \left[\frac{h(t_{ig}; \boldsymbol{\theta_g})}{1 - H(t_{ig}^L; \boldsymbol{\theta_g})} \right]^{1-c_{ig}} \left[\frac{1 - H(t_{ig}; \boldsymbol{\theta_g})}{1 - H(t_{ig}^L; \boldsymbol{\theta_g})} \right]^{c_{ig}}$$

where t_{ig} is the observed failure or survival time of unit i in group g; t_{ig}^L is the left truncation time for unit i in group g; and c_{ig} is an indicator if unit i in group g is right censored. All times are again in hours.

In the hierarchical model, the parameters are modeled as random variables, varying across the different sub-populations. Let

$$\begin{aligned} \sigma_{1g} &\overset{ind.}{\sim} \text{Log-normal}\left(\eta_{\sigma_1}, \tau^2_{\sigma_1}\right) \\ \sigma_{2g} &\overset{ind.}{\sim} \text{Log-normal}\left(\eta_{\sigma_2}, \tau^2_{\sigma_2}\right) \mathrm{T}\,(0,1) \\ t_{p_1 g} = \exp\left(\mu_{1g} + \sigma_{1g}\, \Phi^{-1}(p_1)\right) &\overset{ind.}{\sim} \text{Log-normal}\left(\eta_{t_{p_1}}, \tau^2_{t_{p_1}}\right) \\ t_{p_2 g} = \exp\left(\mu_{2g} + \sigma_{2g}\, \Phi^{-1}(p_2)\right) &\overset{ind.}{\sim} \text{Log-normal}\left(\eta_{t_{p_2}}, \tau^2_{t_{p_2}}\right) \\ \pi_g &\overset{ind.}{\sim} \text{Logit-normal}(\eta_\pi, \tau^2_\pi). \end{aligned} \tag{4.3}$$

As discussed in Section 4.3.1, we reparameterize the Weibull distribution in terms of a quantile and the shape parameter because the data we encounter are more informative about the lower tail of their respective lifetime distribution (note that if T has a Weibull distribution, σ is a shape parameter for the distribution of T and a scale parameter for the distribution of $\log(T)$). We truncate the distribution of σ_{2g} at 1 (indicated by $T(0,1)$ above) restricting the wearout failure mode to have an increasing hazard function (i.e., $\beta_{2g} = 1/\sigma_{2g} > 1$).

In principle, we prefer this full model because we tend to believe that every population has a distribution with a distinct set of parameters. In practice, however, we may consider restrictions to reduce the number of parameters if the data do not support their inclusion. For example, we may consider a model with a common σ_1, letting $\sigma_{1g} = \sigma_1$ for all $g = 1, \ldots, G$. A practical strategy is to begin with a simple model and gradually add complexity, reevaluating model adequacy at each iteration. In Section 4.6.3, we illustrate this process, fitting a series of models with increasing complexity and evaluating their performance using a measure of leave-one-out predictive error.

4.6 Analysis of the Backblaze hard drive data

4.6.1 Modeling

4.6.1.1 Hierarchical models

We considered four models, all based on (4.3), differing by which model parameters are held constant across drive-models. These are (from most to least restrictive):

1. $\pi_g = \pi, \quad t_{p_1 g} = t_{p1}, \quad \sigma_{1g} = \sigma_1, \quad t_{p2g} = t_{p2}, \quad \sigma_{2g} = \sigma_2$

2. $\pi_g = \pi, \quad t_{p_1 g} = t_{p1}, \quad \sigma_{1g} = \sigma_1, \quad \sigma_{2g} = \sigma_2$

3. $\pi_g = \pi, \quad t_{p_1 g} = t_{p1}, \quad \sigma_{1g} = \sigma_1$

4. $t_{p_1 g} = t_{p1}, \quad \sigma_{1g} = \sigma_1$

The set of model specifications was chosen based data, interpretation of the model, as well as estimation considerations. Drive-model-specific parameters for the the wearout failure mode $(t_{p_2 g}, \sigma_{2g})$, and the proportion defective (π_g), are considered as a means to account for heterogeneity across drive-models in the right tails of the failure distribution. Going from a common model for all drive-models and gradually increasing the complexity of the model, we conclude that Model 4 provides the best description of the data. Model 4 allows for the probability of infant mortality as well as the shape and scale parameters for the wearout failure mode to vary by drive-model.

For all of the models we consider, the parameters for the infant mortality failure mode are held in common across drive-models. We found that there was often insufficient information to model these parameters hierarchically. Moreover, assuming a common distribution for infant mortality provides a meaningful interpretation and comparison of π_g and $\pi_{g'}$ $(g \neq g')$. Also the failure-time distribution of the defective subpopulation (as opposed to the proportion defective) is not of high interest.

4.6.1.2 Prior distributions

To complete the full probability model, we need to select prior distributions for the parameters governing the hierarchical model. We select proper prior distributions to ensure proper posterior distributions.

For Models 1–4, different sets of restrictions required different prior distribution specifications, which were assigned as follows:

Model 1 Constrain all drive-models to the same GLFP distribution. For this "reduced" model we assume the same prior distributions as in Example 1.

Model 2 $t_{p,g}$ varies by drive-model. To help with model identifiably, we tighten the priors on the infant mortality failure mode:

$$\pi \sim \text{Logit-normal}\langle 7.0 \times 10^{-3},\, 2.6 \times 10^{-1} \rangle,$$

$$\sigma_1 \sim \text{Log-normal}\langle 1.4 \times 10^{-1},\, 7.1 \rangle,$$

$$t_{p1} \sim \text{Log-normal}\langle 2.2 \times 10^{1},\, 5.5 \times 10^{4} \rangle.$$

Model 3 $t_{p,g}$ and σ_{2g} vary by drive-model. Prior distributions for the constrained parameters are the same as for Model 2.

Model 4 π_g, $t_{p,g}$ and σ_{2g} vary by drive-model. Priors for constrained parameters are the same as for Model 2.

Where applicable prior distributions on hyperparameters are as listed below. We follow the recommendation of Gelman et al. (2014c) to use half-Cauchy prior distributions on hierarchical scale parameters. As for the location hyperparameters, we choose weakly informative prior distributions centered around the corresponding prior mean for the non-hierarchical model.

$$\begin{aligned}
\eta_\pi &\sim \text{Normal}(-3, 1) \\
\tau_\pi &\sim \text{Half-Cauchy}(0, 1) \\
\eta_{\sigma,2} &\sim \text{Normal}(0, 2) \\
\tau_{\sigma,2} &\sim \text{Half-Cauchy}(0, 1) \\
\eta_{t_{.2_2},2} &\sim \text{Normal}(9, 2) \\
\tau_{t_{.2_2},2} &\sim \text{Half-Cauchy}(0, 1).
\end{aligned}$$

4.6.2 Computation

Each model was fit using the `rstan` (Stan Development Team, 2016) package in `R` (Ihaka and Gentleman, 1996), which implements a variant of Hamiltonian Monte Carlo (HMC) (Betancourt and Girolami, 2015). HMC jointly updates all model parameters by simulating energy preserving paths with random initial momentums along the posterior density. This is done to reduce autocorrelation and efficiently explore the posterior. Multiple chains were run, each with 1500 iterations after 1500 warmup iterations: 4 chains were run for Models 1, 2 and 3 for a total of 6,000 post burn-in iterations and 16 chains were run for Model 4 for a total of 24,000 post burn-in iterations. The Gelman-Rubin potential scale reduction factor was used to provide a check for adequate mixing of the multiple chains. Upon convergence, $\hat{R}$, converges to 1. The respective maximum values of $\hat{R}$ across all model parameters for Models 1 through 4 (5, 50, 95 and 140 parameters, respectively) were 1.005, 1.028, 1.009 and 1.004. Other diagnostics provided by the software (tree-depth, and divergent transitions) did not indicate problems with sampling. Plots of the posterior draws were inspected for parameters with the fewest effective samples; these did not suggest features that were inadequately explored.

4.6.3 Model selection

As we have discussed, while we prefer to allow all of the GLFP model parameters to be drive-model specific, there are practical issues with fitting the full model. For example, due to heavy left-truncation, there may be insufficient information for particular drive-models to estimate all of the parameters.

Figure 4.4 provides a visual comparison for Models 1 through 4 by plotting the pointwise posterior median of the cdf for each drive-model. The large left panel shows the adjusted K-M estimate. The smaller plots correspond to Models 1 through 4. To make the plots comparable, we use an adjustment based on the parametric model (Model 4). Note that while all the paths for the posterior estimates necessarily pass through the origin, the K-M estimates do not, because, for the K-M estimator, inference is entirely conditional on survival up to left-most left-trunctation time.

These plots suggest that Model 1 is insufficient to explain the observed data. Model 2 certainly captures more of the heterogeneity observed in the K-M estimates; we observe that the assumption of a common σ_2 results in very similar progressions in the cdf, which may be too restrictive. Model 3, which allows σ_2 to vary by group, suggests there is evidence to support variation in this parameter among drive-models. While Model 3 and 4 do display differences, they are relatively subtle.

We can also compare posterior estimates of time to failure under Models 1-4 for each drive-model individually. For these comparisons we calculate pointwise the posterior median of the proportion failing over a grid. That is, for every model, and for a fixed set, $\tilde{t} = \{\tilde{t}_1, \ldots, \tilde{t}_M\}$, we compute

$$\text{median}\left\{\mathrm{H}\left(\tilde{t}_m|\pi_g^{(s)}, t_{p1g}^{(s)}, \sigma_{1g}^{(s)}, t_{p2g}^{(s)}, \sigma_{2g}^{(s)}\right); s = 1, \ldots, S\right\}, \tag{4.4}$$

where $\theta^{(s)}$ is the s^{th} posterior sample from $p(\theta|y)$.

The GLFP fit for a representative selection of drive-models (Drive-models 2, 9, 14, and 40) are presented in Figure 4.5 (See the Supplementary Material for similar plots for the other drive-models). The black step functions again correspond to the adjusted K-M estimates. As model complexity increases, the hierarchical-model GLFP curves become more flexible and are better able to describe the observed failure data. All of the GLFP models appear to produce higher estimates of the proportion failing, relative to K-M, for Drive-model 2. This behavior is not unexpected, however, because hierarchical models improve model stability through shrinkage; in this context, shrinkage is toward an average cdf. Model 4 is in closest agreement.

Drive-model 9 shows a high proportion of early failures, a fact which Model 1 fails to capture. Model 2 addresses this by estimating an earlier time to wearout, but still results in a poor fit to the data. Models 3 and 4 conform more closely to the adjusted K-M estimate. The disagreement between Models 3 and 4 for Drive-model 9, is interesting: Model 4 allows the proportion of defective units to vary by drive-model, but Model 3 appears to fit just as well in this case, by increasing the scale parameter for the wearout failure mode.

For Drive-model 14, Models 1-3 produce lower estimates of the proportion of drives failing, and for Drive-model 40, they produce higher estimates. On the other hand, Model 4 follows the K-M closely across stages of lifetime. The differences between Models 3 and 4 and Models 1 and 2 are visually apparent. Differentiating between Model 3 and 4 is less clear; for instance, the estimates from Model 3 and 4 are nearly identical for Drive-model 40.

From the point of view of a consumer of hard disk drives, the goal of an analysis such as this is may be to rate manufacturers and/or to inform future purchases based on predicted product performance. In this case, we should choose the model that will provide the most accurate prediction for future units of previously observed drive-models. One appropriate criterion for model selection, then, is the log pointwise predictive density (lpd),

$$\begin{aligned}\text{lpd} &= \sum_{g=1}^{G}\sum_{i=1}^{n_g}\log[p(t_{new,g,i}|t, t^L, c)]\\ &= \sum_{g=1}^{G}\sum_{i=1}^{n_g}\log\left[\int p(t_{new,g,i}|\theta)p(\theta|t, t^L, c)d\theta\right].\end{aligned}$$

Using this criterion, we should choose the model with the highest expected lpd (elpd) for a new data set $\{t_{new,g,i} : g = 1, \dots, G; i = 1, \dots, n_g\}$. Notationally suppressing the conditioning on t^L and c,

$$\text{elpd} = \text{E}_h\,\text{lpd} = \sum_{g=1}^{G}\sum_{i=1}^{n_g}\int \log[p(t_{new,g,i}|t)]h(t_{new,g,i})dt_{new,g,i}, \tag{4.5}$$

where h is a density for the true distribution of $t_{new,g,i}$. Although h is unknown, leave-one-out cross-validation provides a robust estimator:

$$\widehat{\text{elpd}} = \sum_{g=1}^{G}\sum_{i=1}^{n_g}\log[p(t_{g,i}|t_{-(g,i)})]. \tag{4.6}$$

The R package, `loo` (Vehtari et al., 2016), computes an approximation of (4.6) provided that the observations t_i are conditionally independent. It employs an importance sampling method that uses S saved MCMC draws to compute smoothed importance weights, $w_{i,s}$, approximating $\log[p(t_{g,i}|t_{-(g,i)})]$ with $\sum_{i=1}^{S} w_{i,s} p(t_{g,i}) / \sum_{s=1}^{S} w_{i,s}$. In addition to providing an estimate of (4.5), their method also produces a standard error based on an asymptotic result. For details, refer to Vehtari et al. (2017). Results showing the estimated elpd and associated standard errors for all 4 models are shown in Table 4.2. We calculated the difference in expected predictive accuracy for Model 4 versus 3, Model 3 versus 2, and Model 2 versus 1, as well as the standard error of the difference (Table 1). For the models we considered, each successive increase in model complexity increases elpd. Of all the models, Model 4 has significantly better elpd compared to the the other 3 models.

4.6.4 Results

4.6.4.1 Model assessment

Having chosen Model 4 as the best of the candidate models, we would like to assess whether our model provides an adequate description of the observed data. One way to do this is to draw replicate data sets from the posterior predictive distribution. This is a general approach recommended by Gelman et al. (1996) for models where no classical goodness-of-fit test is available. In the hierarchical setting, there can be different formulations of the predictive distribution. In the present context, one could predict the lifetime of a new unit of a previously observed drive-model or a new drive-model. We choose to focus on the first of these and apply it to individual drive-models. There are many possible approaches to the posterior predictive check. For this problem, we chose to consider the posterior predictive distribution of the K-M estimate for each drive-model g.

Because the variability in the K-M estimator is primarily a function of the number of at-risk units, we want to replicate the pattern of censoring that was observed. That is, for unit i of drive-model g,

$$p(t_{rep,g}) = \int \prod_{i=1}^{n_g} f(t_{new}|t_{g,i}^{L}, c_{g,i}, \theta)\, p(\theta|t) d\theta.$$

Because we are working with posterior draws, $\theta^{(s)}$, we can draw from this distribution by choosing a draw, $\theta^{(s)}$, at random, then conditionally draw $t_{rep,g}$ from $f(t_{new}|t^L_{g,i}, c_{g,i}, \theta^{(s)})$.

An obstacle remains, in that the censoring mechanism itself is not fully observed. Our admittedly simple approach is to assume that failed units would have been the censored had they survived until the latest censoring or failure time for that drive model. We generated 19 replicates for each drive model. This gives a 1/20 chance under the hypothesis of no difference that the K-M estimate based on the observed data would be "most extreme" relative to those from the replicated data, but not too many so as to produce an overly cluttered plot. Our method for generating replicate data sets, $y^{(s)}_{rep,g}$, $s = 1 \ldots 19$, for each drive-model $g = 1, \ldots, 44$, is as follows:

1. Simulate $\tilde{t}^{(s)}_{rep,g,i} \sim \text{GLFP}(\pi^{(s)}_g, t^{(s)}_{p1}, \sigma^{(s)}_1, t^{(s)}_{p2g}, \sigma^{(s)}_{2g})$ where each set of parameters, $\left(\pi^{(s)}_g, t^{(s)}_{p1}, \sigma^{(s)}_1, t^{(s)}_{p2g}, \sigma^{(s)}_{2g}\right)$, is a random draw from the full posterior distribution.

2. Repeat step 1 until $\tilde{t}_{rep,g,i} > t^L_{g,i}$.

3. Set $d^{(s)}_{g,i} = \begin{cases} \max\{t_{g,i} : i \geq 1\}, & \text{if } c_{g,i} = 0 \\ t_{g,i}, & \text{if } c_{g,i} = 1. \end{cases}$

   Lastly, set $(t^{(s)}_{rep,g,i}, c^{(s)}_{rep,g,i}) = \begin{cases} (d^{(s)}_{g,i}, 1), & \text{if } \tilde{t}^{(s)}_{rep,g,i} > d^{(s)}_{g,i} \\ (\tilde{t}^{(s)}_{rep,g,i}, 0), & \text{if } \tilde{t}^{(s)}_{rep,g,i} \leq d^{(s)}_{g,i}. \end{cases}$

We can compare the 19 simulated data sets $y_{rep,g,i}$ to the observed data $y_{g,i}$ by using them to re-estimate the K-M estimator. The resulting plots are displayed in Figure 4.6. Only a representative sample of plots is shown here for brevity (see the Supplementary Material for similar plots for all of the drive-models). For most drive-models, the estimates based on the observed data look similar to those based on the replicated data. There are a few drive-models, however where this is not the case. Drive-model 14 shows a discrepancy between the GFLP fit and the K-M fit in the right tail. In particular, the GLFP predictions appear conservative, overpredicting the proportion failing after about 2.5 years. Due to the large number of failures for this drive-model, the posterior for the hierarchical model is similar to the stand-alone model.

Other drive-models showing a lack of fit are 4 and 34. We first address Drive-model 4; our diagnosis of 34 is analogous, so the following comments extend to that drive-model as well.

To diagnose the cause of the discrepancies between the observed data for Drive-model 4 and its posterior predictive distribution, we consider the event plot for a random sample of units for this Drive-model while looking at estimates of lifetime (Figure 4.7). From these plots, we can see that the discrepancy between the posterior and K-M estimates is driven by failures among a relatively small subset of units, namely those with left-truncation times smaller than 2500 hours. Further investigation reveals that, if we exclude the newest 75 drives, those with truncation times less than 800 hours, the K-M estimate falls within the posterior 90% pointwise credible band. This situation suggests that, despite belonging to the same drive-model, there may be something different about the newer drives leading to worse reliability.

A possible explanation for the discrepancy shown in Figure 7 is that the units of Drive-model 4 were introduced into the field over a period of 22 months. Manufactures of hard drives (and most other products) often make changes to the product design over time to either improve reliability or to reduce cost. Such changes will sometimes have an important effect on the product's failure-time distribution, invalidating one of our important assumptions that failure-time distribution for a drive-model does not depend on the date of manufacture.

Figure 8 shows a similar pair of plots for Drive-model 43. The data for this Drive-model are heavily truncated; the observed units had run for quite some time before data were made available with the earliest left-truncation time at 12,189 hours. In the bottom half of Figure 4.8 we contrast the posterior median for drive-model 43 to an average GLFP cdf for the entire population, which we call the "global average;" a description of the global average is given in Appendix C. We can see that the lack of data for this drive-model during early life results in a diffuse posterior centered around the global average until about 10,000 hours when they diverge. Although the posterior is close to the adjusted K-M, it is shrunk toward the global average.

4.6.4.2 Drive-model comparisons

We consider the problem of ranking drive-models with respect to economical value. From a business perspective, it is clear that we should prefer the drive-models which will provide more years of service. For ease of exposition, we will assume that the purchase price of a hard-drive is the same across drive-models.

There are two sources of variation at play: the posterior uncertainty in the parameters, which largely depends on the amount of data, and the uncertainty in future observations conditional on the parameters. The *posterior predictive* distribution incorporates both sources of variability.

$$p(t_{g,new}|t) = \int p(t_{g,new}|\theta_g)p(\theta_g|t_g)\, d\theta_g$$

We can sample from this distribution by drawing $t_{g,new}^{(s)}$ from GLFP$(\pi_g^{(s)}, t_{p1}^{(s)}, \sigma_1^{(s)}, t_{p2g}^{(s)}, \sigma_{2g}^{(s)})$, for $s = 1, ..., S$, using the saved posterior draws for the model parameters. The mean time-to-failure (MTTF) for drive-model d can be estimated by

$$\frac{1}{S}\sum_{s=1}^{S} t_{g,new}^{(s)}.$$

While MTTF is often an important metric, due to the anticipation of advancements in technology, we can expect that the relative value of computer hardware will depreciate rather quickly. In simple terms, given two hard drives, one would prefer to have both of them work for five years than for one to work for ten years and the other to not work at all. To account for depreciation, rather than using MTTF as the metric for drive-model comparison, we use the value at replacement. The US Internal Revenue Service considers computer hardware to be "5-year" equipment (US Department of Treasury, 2016). Using"the declining balance method" the rate of depreciation is 40% per year.

Let $L(t) = e^{-0.4t}$ represent the value at the time of failure relative to a new unit. We can rank the drive-models by

$$\mathrm{E}(L(t_{g,new})|t)) \approx \frac{1}{S}\sum_{s=1}^{S} L(t_{g,new}^{(s)}).$$

Posterior credible intervals for $L(t_{g,new})$ are shown in the left panel of Figure 4.10. A zoomed version of the top eight drive-models, as ranked by $\mathrm{E}(L(t_{g,new})|y)$ are shown in the bottom right panel. The panel above shows credible intervals for MTTF. We can see that ranking by MTTF would lead to a different result – a new Drive-model 8 unit is expected to last the longest, but the possibility of it failing early is higher than for the drive-models ranked highest by our depreciation-aware metric. The top four drive-models (4, 5, 2, 3) seem comparable, with a new Drive-model 4 unit expected to depreciate the most. However, given the discrepancy of the model fit for Drive-model 4 (noted in the previous subsection), in the absence of more early life data, we might harbor some lingering concern about possible quality-control issues with this Drive-model.

We can also use the posterior draws to compute quantiles of interest, which in reliability applications is typically more informative than a measure of central tendency. The top panel of Figure 9 shows estimates of $t_{0.10}$ (i.e., the amount of time it takes for 10% of hard drives to fail), for all drive-models. In this application 50% CIs are more meaningful as there is lots of uncertainty in some of the parameters: especially for those with few drives in operation. Interestingly, the top seven drive-models ranked by MTTF are also the drive-models with the largest $t_{0.10}$. Another interesting feature to compare is π, the proportion defective. The bottom panel of Figure 9 shows the posterior credible intervals of π for each drive-model. The plot is again sorted according to $t_{0.10}$ so it can be compared to the plot above. While for many of the drive-models the ordinal ranking is the same as on the top, there are some drive-model comparisons, for example 23 and 18, that differ in ranking if compared using $t_{0.10}$ or π. Depending on a user's application or the hard drive warranty length, using the π's as a comparison tool may be a relevant parameter of interest.

4.7 Discussion

This paper offers a new approach to modeling and making inference for the lifetime of consumer products using incomplete field reliability data. Products with long lifetimes often have few failures and when failures do occur the cause is frequently unknown to the analyst. The GLFP model provides a framework to accurately model data with evidence of multiple failure modes. Moreover, when there are multiple product populations within the same product class, our hierarchical modeling approach borrows strength across brands with many observed failures to help make inference for those populations with little information. This can be important when trying to leverage multiple sources of data with various sample sizes and levels of censoring and truncation. We found that the use of weakly informative priors on the hyperparameters can increase the overall efficiency of the MCMC. We now address some important model assumptions and their potential for impacting statistical inference.

First, we assume the failures occurring from various causes can be approximately assigned to two phases of product life. In practice, there might be additional phases, clearly supported by the data, making the GLFP model too rigid. This rigidity is illustrated by the right tail of the estimated distribution in Figure 4.2.

Second, we assume exchangability of units within a population. This assumption is due in part to ignorance about which units are defective but also of potentially important covariates. We do not account for batch effects or varying conditions in the facility which have impacted the observed failure rates. These factors are confounded with drive-model in our analysis. An example of a possible batch effect is noted in Section 4.6.4.1 and illustrated by Figure 4.7.

Consumers of manufactured goods usually lack the data and expertise to perform a failure analysis to determine the cause of failure. Moreover, their interest is primarily in the lifetime distribution of a product rather than the specific cause of failure. The ability to fit marginal models, such as GLFP, allow consumers and manufacturers to accurately model products with bathtub hazard lifetime distributions and make comparisons across different product groups. Because the Bayesian approach provides full posteriors distributions once a model is fit, the analyst can easily estimate a functional, such as a quantile or depreciation factor, that allow products to be compared using statistics that are meaningful in the context of the data.

While not presented here, another potential application is forecasting the number of future failures over a fixed period of time for a current population of drives. Hong et al. (2009) proposed a method, based on the Poisson-binomial distribution. It would be straightforward to adapt their approach to work with our method. For details and examples see Section 6 of Hong et al. (2009) and Xu et al. (2015).

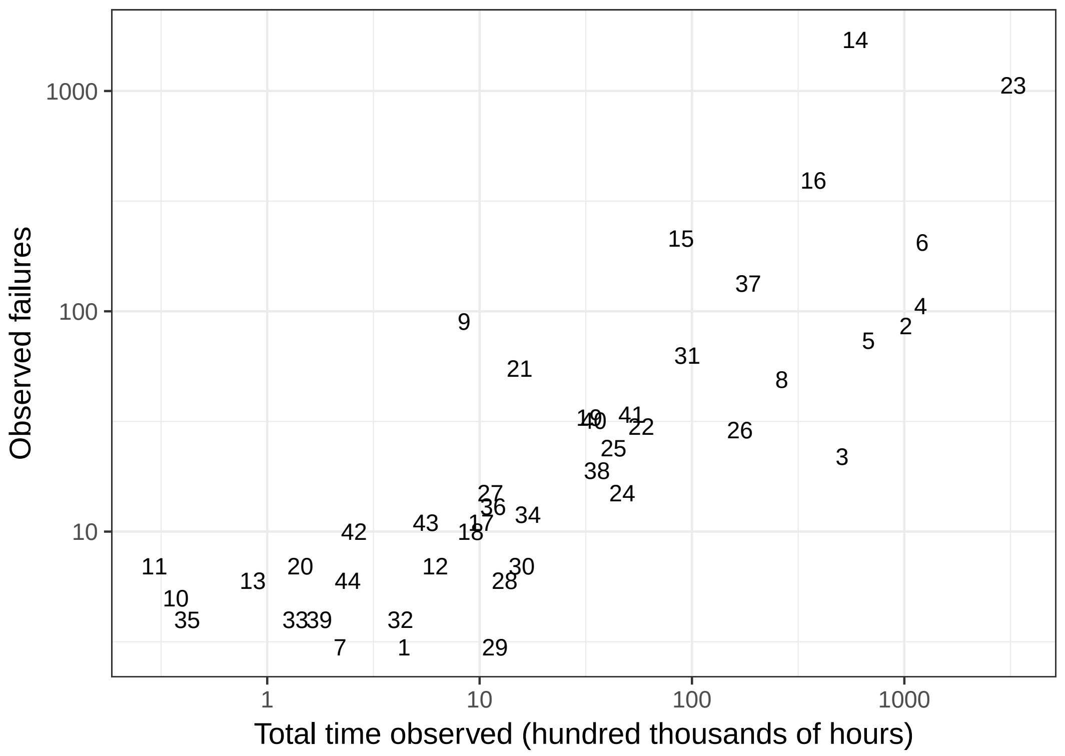


Figure 4.1: Scatterplot of total observed running time (hundred thousands of hours) versus total number of observed failures. Each number labels a unique drive-model. Axes are on the log scale.

	2.5%	50%	97.5%
π	0.033	0.054	0.099
$\beta_1 = 1/\sigma_1$	0.47	1.13	1.72
$\beta_2 = 1/\sigma_2$	4.47	4.70	4.95
$t_{0.5,1}$	1.03	2.28	3.99
$t_{0.2,2}$	18.0	18.2	18.6

Table 4.1: Posterior 95% Credible Intervals for the 5 GLFP parameters for Drive-model 14. Quantiles are in thousands of hours.

Table 4.2: Expected Log Pointwise Predictive Density (elpd) for each model specification. Each model is compared to the more parsimonious model below. The estimated difference of expected leave-one-out prediction errors between the two models, as well as the standard error of the difference, is also presented.

	$\widehat{\text{elpd}}$	Difference in $\widehat{\text{elpd}}$	SE of the Difference
Model 4	-13309.5	40.7	11.3
Model 3	-13350.2	458.8	31.0
Model 2	-13809.0	3674.6	96.2
Model 1	-17483.6		

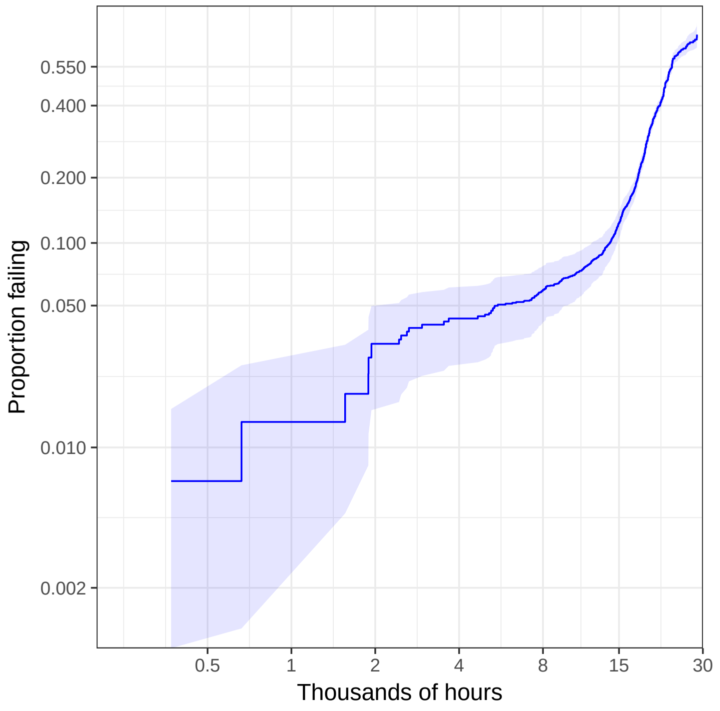


Figure 4.2: K-M estimate for Drive-model 14. Uncertainty band correspond to pointwise standard errors (using Greenwood's formula.)

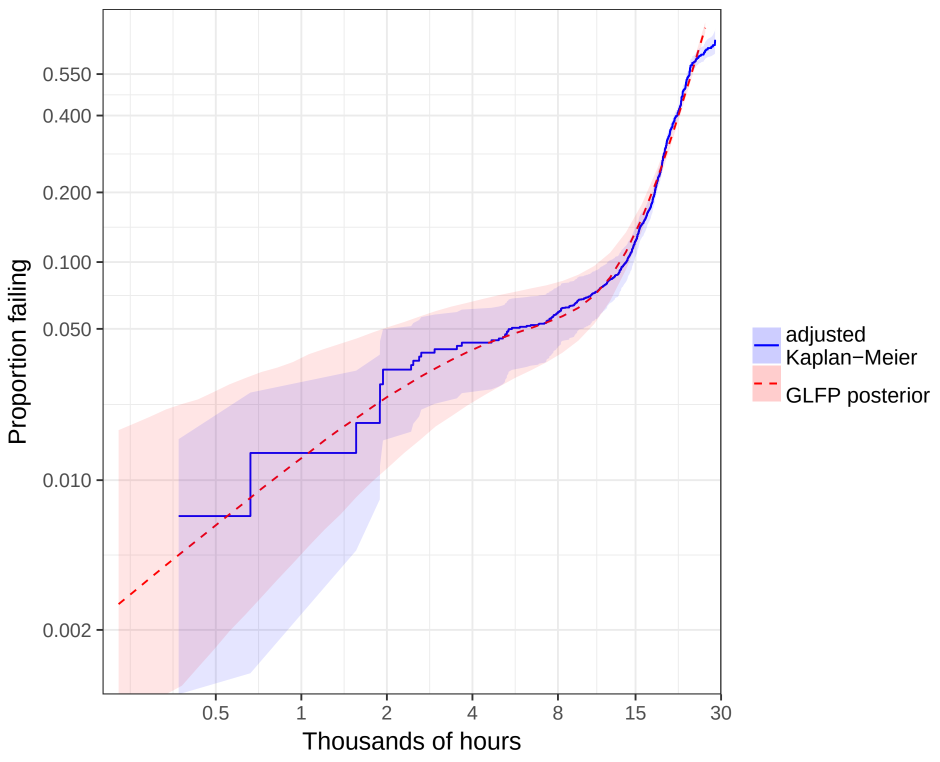


Figure 4.3: The estimated GLFP for Drive-model 14 plotted on Weibull paper. The dashed line corresponds to the median of the posterior draws; pointwise 90% credible bounds are shown in red. The solid line is an adjusted K-M estimate with pointwise 90% credible bounds in blue.

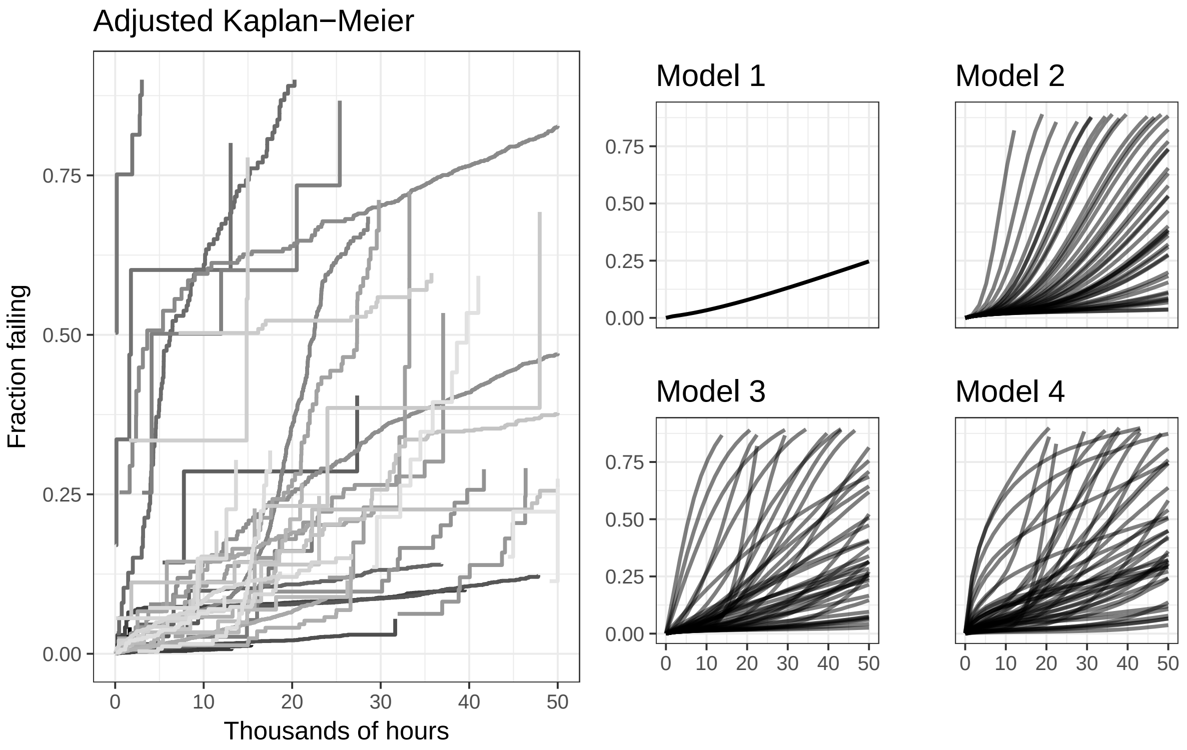


Figure 4.4: Left: Adjusted K-M estimates of the time to failure for each of the drive-models in the Backblaze data. Right: Pointwise posterior median cdf estimates for Models 1-4, ordered left to right, top to bottom.

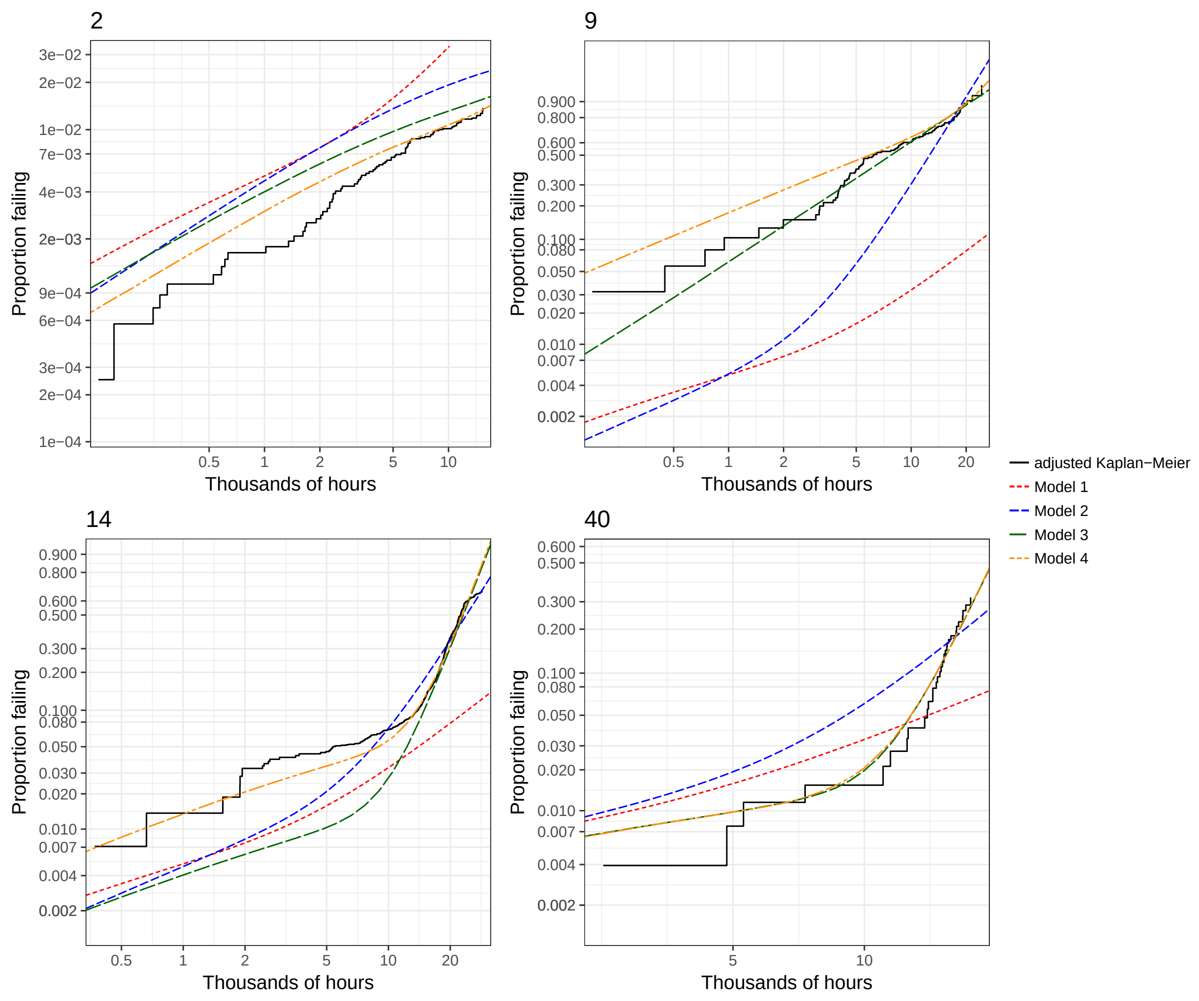


Figure 4.5: Posterior median of proportion failing as a function of time under Models 1-4 for sample of drive-models, with axes scales as for Weibull probability plotting. The solid step function is the adjusted K-M estimate.

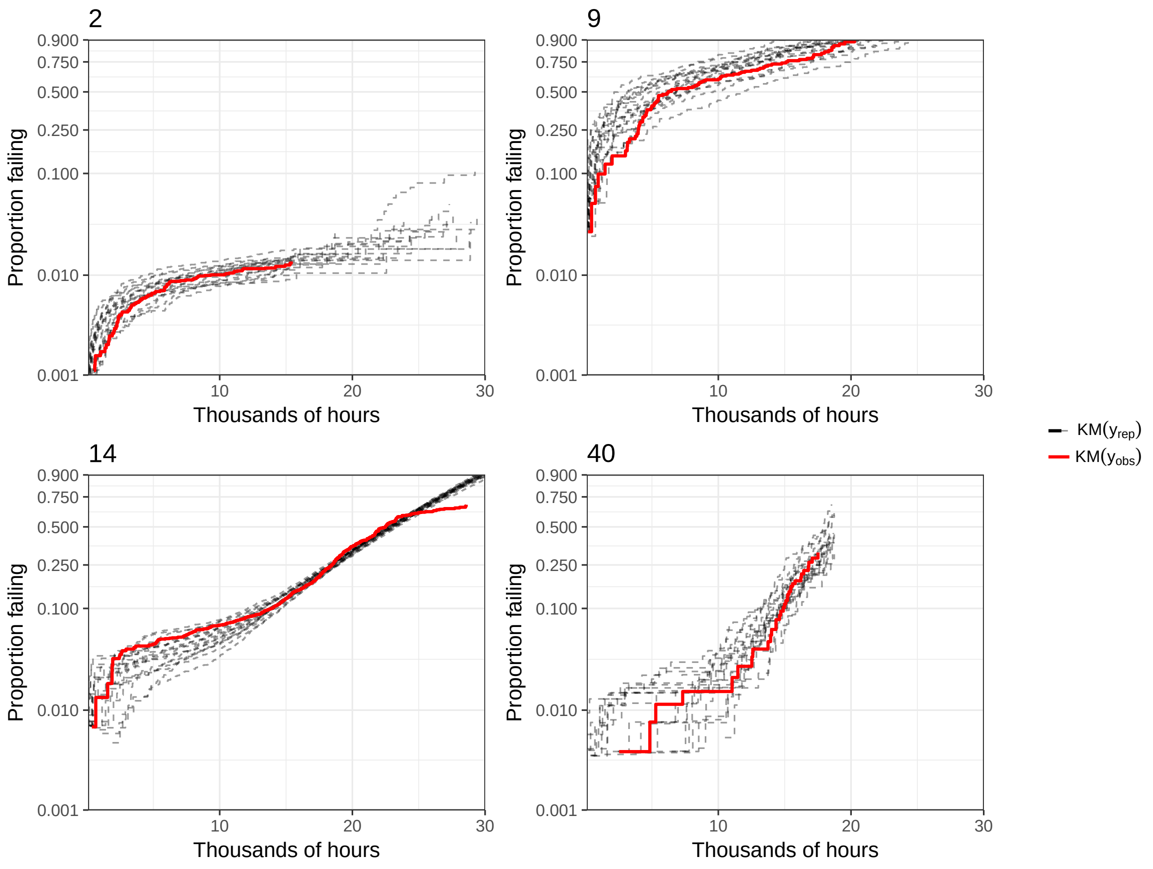
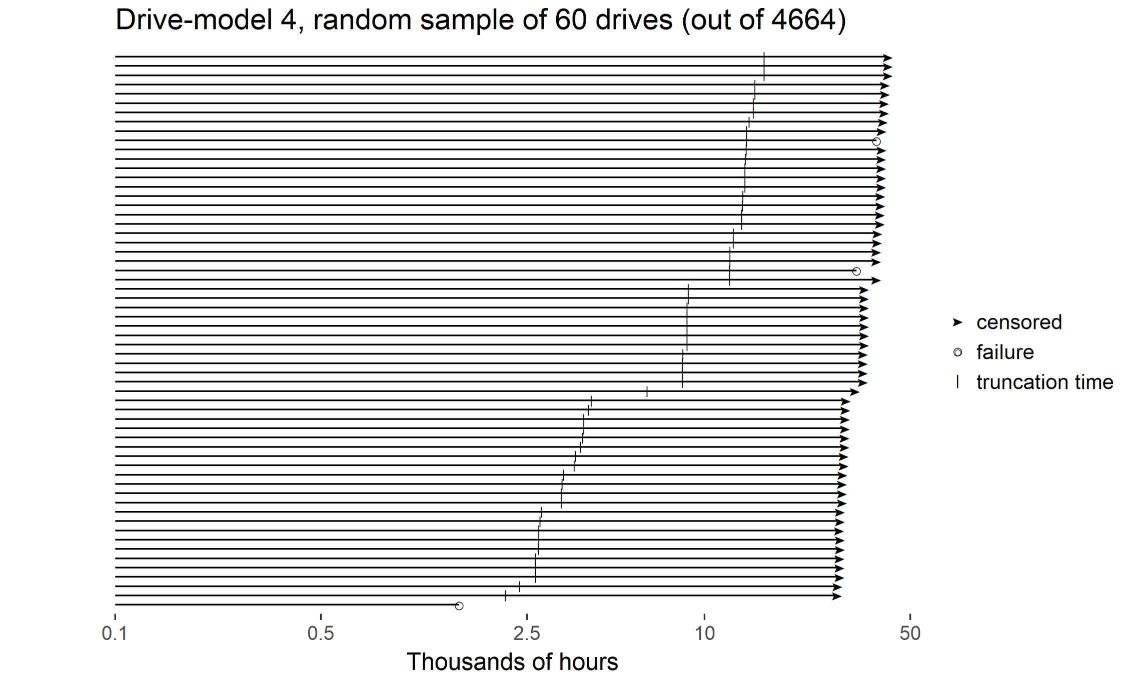


Figure 4.6: Adjusted K-M estimates of the proportion failing for a representative subset of drive-models. Both the original data (**bold** line) and 19 “replicated” data sets from the posterior predictive distribution (*dashed* lines) are shown.



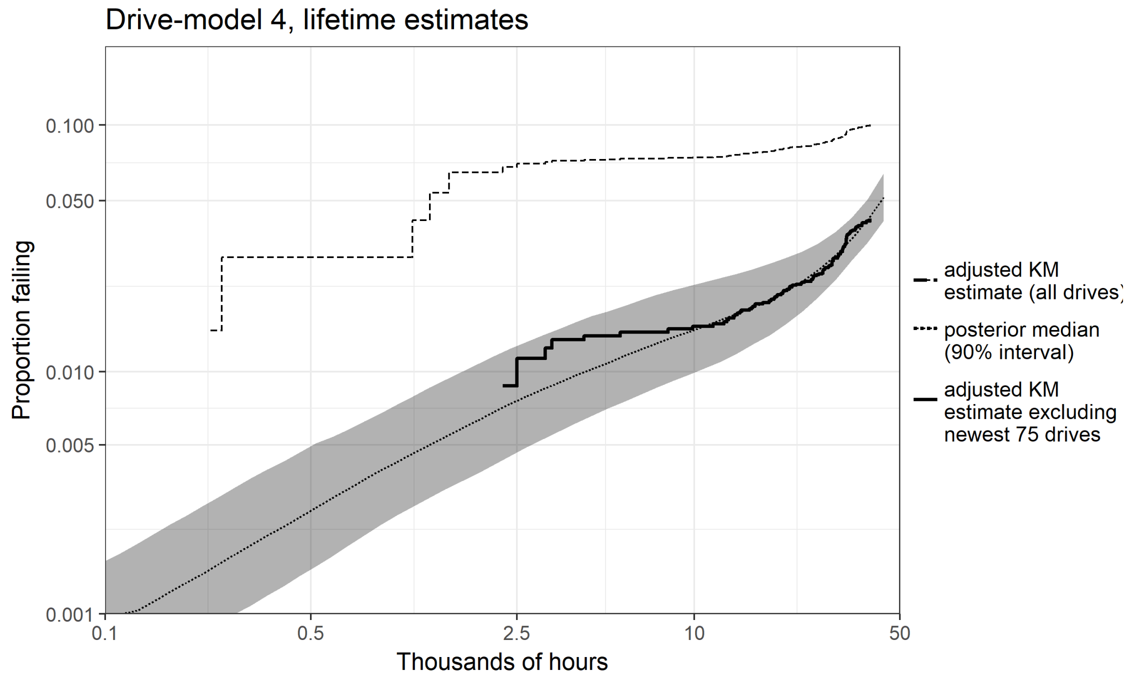


Figure 4.7: Top: Event plot for 60 randomly sampled drives of Drive-model 4. Note that all drives in the sample are left truncated well past 1000 hours. Bottom: The adjusted K-M estimate (dashed step function) is substantially higher than the pointwise posterior median (dashed line; shaded region showing 90% credible interval). However, this discrepancy is due to several early failures among the small set of drives with the earliest left-truncation times (the newest set). The same adjusted K-M estimator after the exclusion of the newest 75 drives (solid step function) shows close agreement with the posterior median.

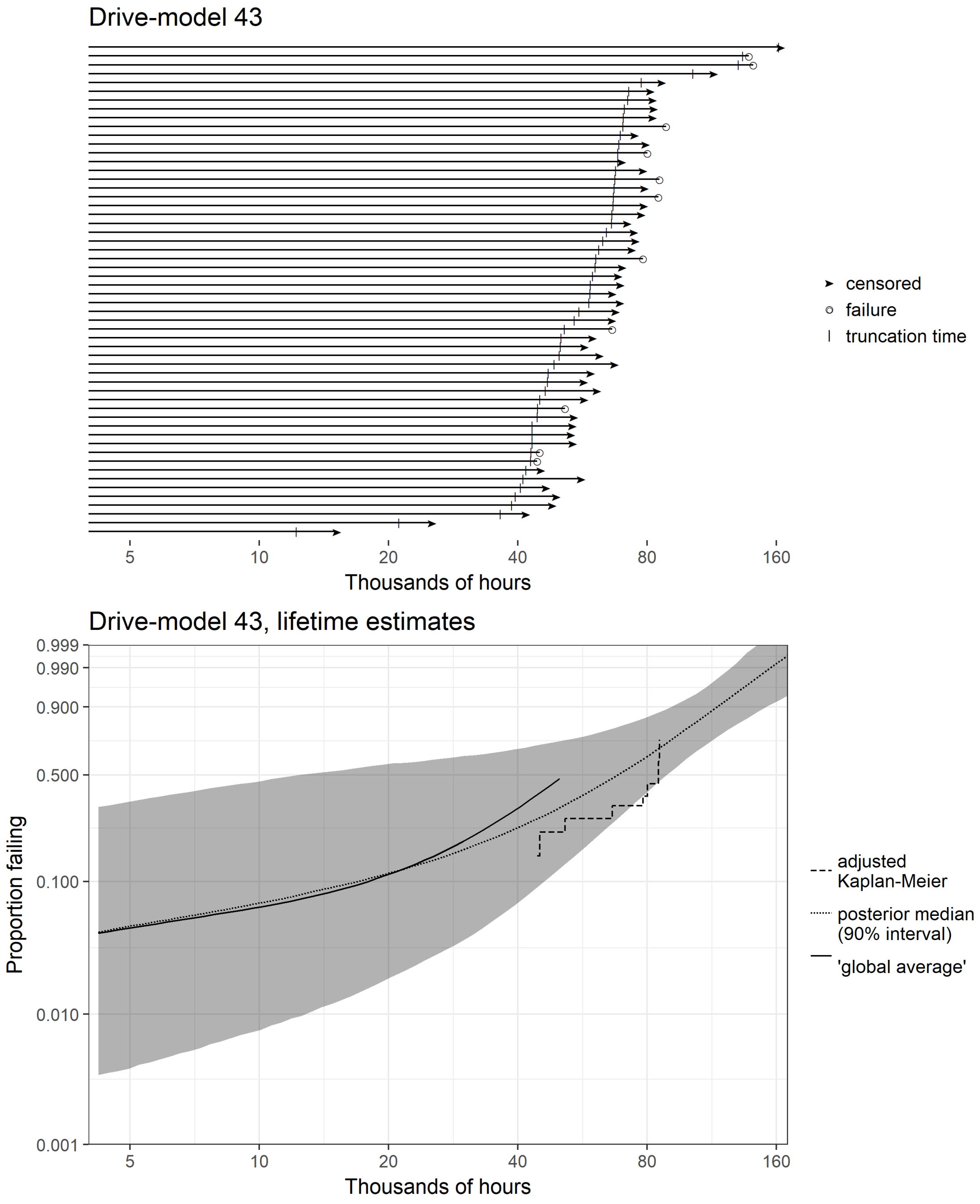
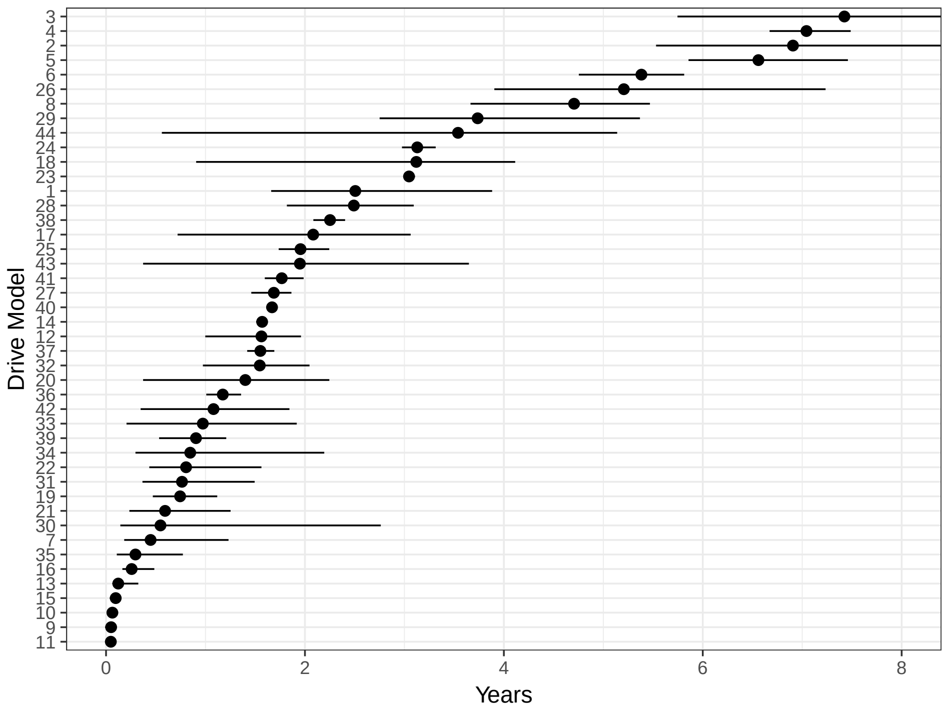
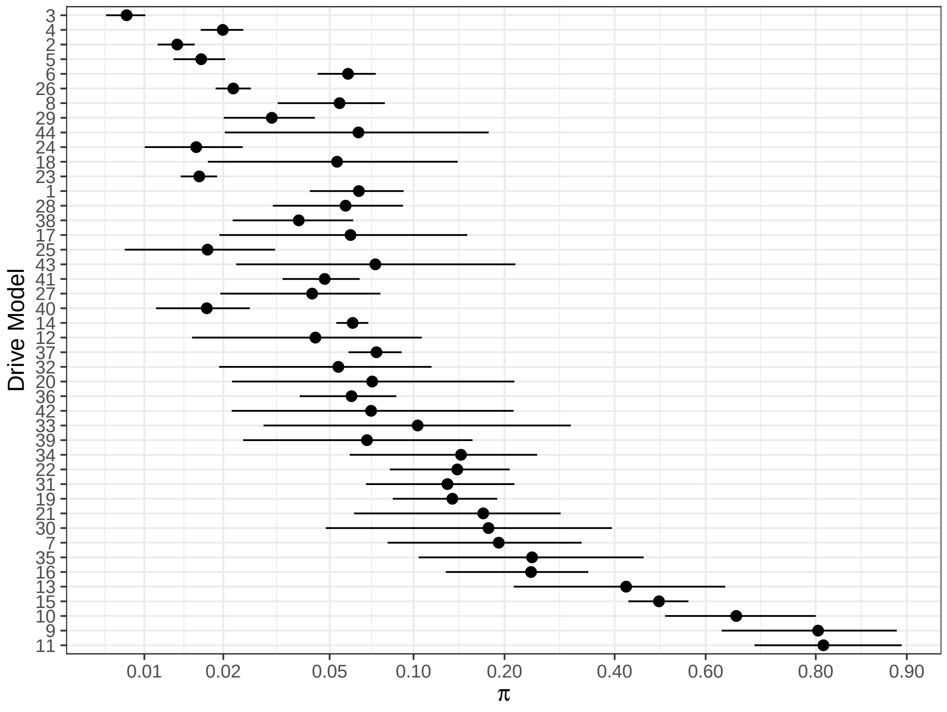


Figure 4.8: Top: Event plot for all units of Drive-model 43. Data for all units are left-truncated, being observed conditional on survival beyond 12,000 hours. Bottom: The dashed step function correponds to the adjusted K-M estimate. Due to the heavy left-truncation, the posterior median for Drive-model 43 (dashed line) coincides with the posterior median of the “global model” (dotted line) until the first left-truncation time.



(a) Point estimate and 50% CI for $t_{0.10}$



(b) Point estimate and 50% CI for π

Figure 4.9: (a) 50% CI (in years) for $t_{0.10}$ (b) 50% CI for π plotted on the logit scale. Both plots are sorted based on the median value of $t_{0.10}$.

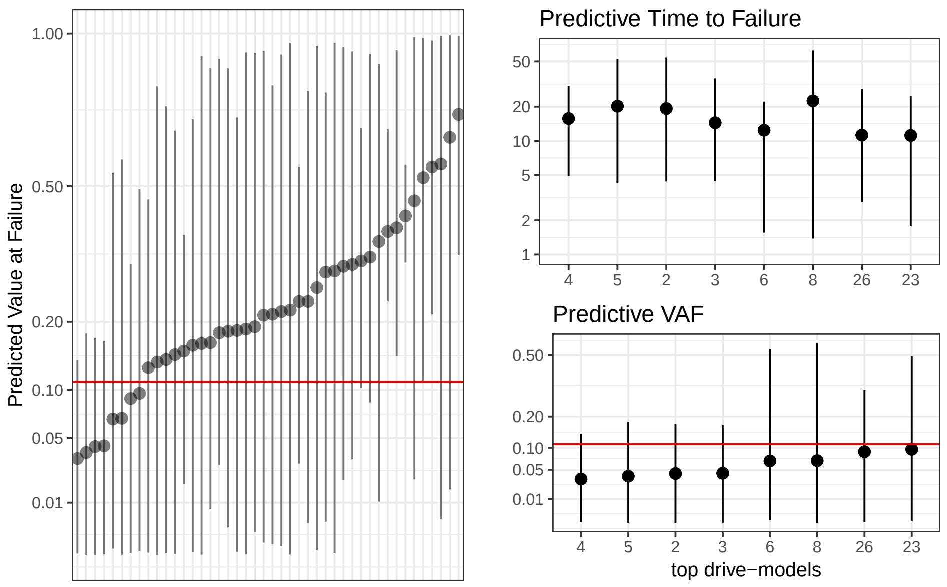


Figure 4.10: Left: 90% CI of the posterior predictive distributions for $e^{-0.4t_{g,new}}$ the depreciated value at failure (VAF) for a new unit for each drive-model. Right: A comparison (90% CI) of the predictive distribution of TTF (top) and VAF (bottom) based on 40% for best 11 drive-models, shows that rankings produced by the two measures are not equivalent.

CHAPTER 5. Summary and conclusion

We have presented two new applications of Bayesian hierarchical models and our implementations of them, demonstrating both their advantages and their practicality.

Our nonparametric Bayesian model for gene expression was in response to the observation that Bayesian models in use today rely on shrinkage priors that, in some instances, are falsified by the data. That this is true is not surprising. For instance, in a complex experiment, we doubt it reasonable to assume that the true effect sizes within a gene are independent; while it may represent our prior knowledge, we would like to be able update our knowledge after observing many, many genes. Bayesian nonparametrics addresses a problem in parametric statistics, that, even in a parametric hierarchical model, where the hyperparameters are learned from the data, the assumption of a particular parametric family is a strong one, and one that can potentially lead to posteriors which are inconsistent with the underlying truth.

By modeling the effects with a DP, we can make minimal assumptions about the distribution of the gene-by-gene effects. Comparisions of the results of our analysis to competing methods show significant differences in the conclusions that we would tend to draw from the two approaches. We think this is largely due to the small sample sizes we considered, where the number of samples per regression parameter within a gene was ≤ 4. We expect that as sample sizes increase the difference between methods would become negligible. Through experimentation and our simulation studies which we presented in Chapter 3, we expect that the relative performance of our BNP method will depend on the unknown true distributions in a particular experiment. In Simulation Study 1, the "truth" was constructed using BNP posterior draws and BNP had a substantial advantage in our assessments of thresholded log-fold change. For Simulation Study 2, where the "truth" was based on voom-limma point estimates, the advantage of BNP was less clear.

A disadvantage of our BNP method is that inference for $\mathcal{P}$ messy, consisting of thousands of weakly identified parameters $(\pi_k,\ \tilde{\beta}_k,\ \tilde{\sigma}_k^2)$. This makes it difficult to understand the reasons for the multimodal posterior we observed for certain genes, for example. A different approach which we considered, but did not pursue, would be to model the gene-specific parameters with a *Dirichlet process mixture* by replacing

$$(\beta_g^\top, \sigma_g^2) \sim \mathcal{P}, \quad \mathcal{P} \sim \mathrm{DP}(\alpha Q)$$

with

$$(\beta_g^\top, \log \sigma_g) \sim \mathrm{MVN}(\mu_g, \Sigma_g), \quad (\mu_g, \Sigma_g) \sim \mathcal{Q}, \quad \mathcal{Q} \sim \mathrm{DP}(\alpha R).$$

Effectively, this replaces what was an infinite discrete mixture with an infinite mixture of multivariate normals. This modification would reduce the required number of mixture components by orders of magnitude and thus produce a more parsimonious model fit with smoother and more accountable patterns of shrinkage. We can foresee challenges arising in the implementation of this model; for example, it may be difficult to select and good base measure for Σ_g.

In Chapter 4, we considered the problem of estimating the lifetime of multiple populations of interest where there is heterogenity arising from infant mortality and where the data available from some populations is severely limited due to truncation and right censoring, factors that are common in reliability field data. Being able to estimate lifetime in these circumstances has great potential value to consumers, as it can inform purchasing decisions.

We addressed the heterogenity within populations through a special kind of mixture model, the GLFP introduced by Chan and Meeker (1999), and used a hierarchical model to borrow information across groups. We illustrated our approach with the Backblaze hard drive data where we considered 4 nested models of varying complexity and performing model selection using approximate leave-one-out cross-validation. Our results show that our modeling approach can provide a good fit to complex data.

Future work in this area might consider the consequences of choosing different specifications of the hierarchical model, such as heavy tailed or multivariate distributions, possibly in conjunction with different data coding strategies for purposes of model identification (such as those considered in Chan and Meeker (1999)). Another possible extension of our approach, noted at the end of Chapter 4, is forecasting the number of units needing replacement over a fixed interval of time, which could improve resource management.

APPENDIX A. Use of CUDA libraries in implementation of BNP Gibbs sampler

CUDA libraries

There are several libraries that make implementing these algorithms easier in CUDA. Thrust (Bell and Hoberock, 2011) extends many concepts from the C++ Standard Template Library to Nvidia GPUs. Included are several useful generic algorithms:

- `thrust::for_each`: This algorithm accepts a functor (an instance of a class with a member `operator()` function) and an iterator. The serial version increments the iterator, passing each element to the `operator()` in turn. The parallel implementation produces the same results, only in thread parallel fashion. The `thrust::zip_iterator` is very useful, as it can be used to iterate over a `thrust::tuple`. This approach is very general, allowing for operations involving up to ten variables using an SoA design.

- `thrust::reduce/reduce_by_key`: As described in section 2.5, both reduce and cumulative sum are defined for any associative binary operators. Reduce by key accepts a key argument and a compatible binary predicate that identifies changes in the key. For example, for the binary predicate `x == y`, the key {`1,1,1,2,2,1`} would have the result be three quantities, the reductions of the first three values, the next two and the last, respectively.

- `thrust::inclusive/exclusive_scan/scan_by_key`: Thrust offers multiple versions of scan, which is another term for cumulative sum. The exclusive version results in the array element, a_i, containing the sum s_{i-1} (excluding the value v_i, while the inclusive version leaves a_i containing s_i.

For linear algebra on the GPU, standard installations of CUDA also include CUBLAS Nvidia (2008). CUBLAS has implementions of BLAS/LAPACK routines optimized for the GPU. Typically, calls to the CUBLAS functions are initiated by the CPU and act on device memory (host API). For newer GPUs (compute 3.5 and later), there are routines that can be initiated within a kernel (device API). From the host API, our algorithm uses `cublasDgemm` for multiplying large matrices in device memory. From the device API, our algorithm uses `cublasDtrsv` to solve many small triangular systems of equations in parallel.

Schemes for parallel pseudo-random number generation (PRNG) have been developed for CUDA. Since PRNGs are deterministic and sequential, a natural parallel adaptation is access the same sequence at locations distant enough to avoid overlap or to use a strided access pattern. CURAND Nvidia (2010), provides such functionality for generating normal and uniform random numbers on the GPU. For other distributions, such as gamma, one can write a custom kernel, making use of CURAND as a source of randomness, for threaded sampler.

APPENDIX B. Derivations of full conditionals for the BNP Gibbs sampler

Derivation of full conditionals

These are the full conditional distribution which we need for the Gibbs sampler for the BNP model, i.e.

$$y_g \overset{ind.}{\sim} \text{MVN}\left(X^\top \beta_g, \sigma_g^2 W_g^{-1}\right) \tag{B.1}$$

$$\left(\beta_g^\top, \sigma_g^2\right) \overset{ind.}{\sim} \mathcal{P} \quad \mathcal{P} = \sum_{k=1}^{\infty} \pi_k \delta_{\left(\tilde\beta_k^\top, \tilde\sigma_k^2\right)} \tag{B.2}$$

$$\tilde\beta_k \overset{ind.}{\sim} \text{N}(m_\beta, C_\beta), \quad \tilde\sigma_k^2 \overset{ind.}{\sim} \text{IG}(a_{\sigma^2}, b_{\sigma^2}) \tag{B.3}$$

where W_g is a diagonal matrix containing the precision weights.

We parameterize the gamma distribution using shape parameter, a, and rate parameter b, e.g. with density given by

$$p(x|a,b) = \frac{b^a}{\Gamma a} x^{a-1} e^{-bx}.$$

Cluster allocation parameters, ζ_g

$$\begin{aligned}
p(\zeta_g|\cdot) &= p(\zeta_g|y_g, \tilde\beta, \tilde\sigma^2, \pi) \\
&\propto p(\zeta_g|\pi)\, p(y_g|\zeta_g, \tilde\beta, \tilde\sigma^2) \\
&\propto \sum_{k=1}^{K} \left[\pi_k \text{I}(\zeta_g = k) p(y_g|\zeta_g, \tilde\beta, \tilde\sigma^2)\right] \\
&\propto \sum_{k=1}^{K} \left\{\pi_k \text{I}(\zeta_g = k) \tilde\sigma_k^{-N} |W_g|^{1/2} \exp\left[-\frac{1}{2\tilde\sigma_k^2}\left(y_g - X\tilde\beta_k\right)^\top W_g \left(y_g - X\tilde\beta_k\right)\right]\right\}
\end{aligned}$$

$$\implies Pr(\zeta_g = k) = \frac{\pi_k \tilde\sigma_k^{-N} |W_g|^{1/2} \exp\left[-\frac{1}{2\tilde\sigma_k^2}\left(y_g - X\tilde\beta_k\right)^\top W_g \left(y_g - X\tilde\beta_k\right)\right]}{\sum_{\ell=1}^{K} \pi_\ell \tilde\sigma_\ell^{-N} |W_g|^{1/2} \exp\left[1\frac{1}{2\tilde\sigma_\ell^2}\left(y_g - X\tilde\beta_\ell\right)^\top W_g \left(y_g - X\tilde\beta_\ell\right)\right]}$$

Here, $\text{I}(x = i)$ is an indictor function that is 1 when $x = i$ and 0 otherwise. Thus, $\zeta_g \overset{ind.}{\sim} \text{Categorical}(\hat\pi_{g1}, \ldots, \hat\pi_{gk})$

where

$$\hat{\pi}_{gk} = \frac{\pi_k \tilde{\sigma}_k^{-N} |W_g|^{1/2} \exp\left\{-\frac{1}{2\tilde{\sigma}_k^2}\left(y_g - X\tilde{\beta}_k\right)^\top W_g \left(y_g - X\tilde{\beta}_k\right)\right\}}{\sum_{\ell=1}^K \pi_\ell \tilde{\sigma}_\ell^{-N} |W_g|^{1/2} \exp\left\{1\frac{1}{2\tilde{\sigma}_\ell^2}\left(y_g - X\tilde{\beta}_\ell\right)^\top W_g \left(y_g - X\tilde{\beta}_\ell\right)\right\}}.$$

Cluster mean parameters, $\tilde{\beta}_k$

$$\begin{aligned}
p(\tilde{\beta}_k|\cdot) &= p(\tilde{\beta}_k|y, \zeta, \tilde{\sigma}_k) \\
&\propto \prod_{g:\zeta_g = k} \left[p(y_g|\tilde{\beta}_k, \tilde{\sigma}_k)\right] \; p(\tilde{\beta}_k) \\
&\propto \prod_{g:\zeta_g = k} \left\{\exp\left[-\frac{1}{2\tilde{\sigma}_k^2}\left(y_g - X\tilde{\beta}_k\right)^\top W_g \left(y_g - X\tilde{\beta}_k\right)\right]\right\} \\
&\quad \cdot \exp\left[-\frac{1}{2}\left(\tilde{\beta}_k - m\right)^\top C^{-1}\left(\tilde{\beta}_k - m\right)\right] \\
&\propto \exp\left[-\frac{1}{2}\tilde{\beta}_k^\top \left(\tilde{\sigma}_k^{-2} \sum_{\zeta_g = k} X^\top W_g X + C^{-1}\right)\tilde{\beta}_k - 2\left(\tilde{\sigma}_k^{-2} \sum_{\zeta_g = k} X^\top W_g y_g + C^{-1} m\right)\tilde{\beta}_k\right]
\end{aligned}$$

This is proportional to a multivariate normal density. Thus, $\tilde{\beta}_k \overset{ind.}{\sim} \text{MVN}(\hat{m}_k, \hat{C}_k)$ where

$$\hat{C}_k = \left(\tilde{\sigma}_k^{-2} \sum_{\zeta_g = k} X^\top W_g X + C^{-1}\right)^{-1} \quad \text{and} \quad \hat{m}_k = \hat{C}_k \left(\tilde{\sigma}_k^{-2} \sum_{\zeta_g = k} X^\top W_g y_g + C^{-1} m\right).$$

Cluster variance parameters, $\tilde{\sigma}_k^2$

$$\begin{aligned}
p(\tilde{\sigma}_k^2|\cdot) &= p(\tilde{\sigma}_k^2|y, \zeta, \tilde{\beta}_k) \\
&\propto \prod_{g:\zeta_g = k} \left[p(y_g|\tilde{\beta}_k, \tilde{\sigma}_k)\right] \; p(\tilde{\sigma}_k^2) \\
&\propto \prod_{g:\zeta_g = k} \left\{\tilde{\sigma}_k^{-N} \exp\left[-\frac{1}{2\tilde{\sigma}_k^2}\left(y_g - X\tilde{\beta}_k\right)^\top W_g \left(y_g - X\tilde{\beta}_k\right)\right]\right\} \\
&\quad \cdot (\tilde{\sigma}_k^2)^{-(a_{\sigma^2}+1)} \exp\left(-b_{\sigma^2}/\tilde{\sigma}_k^2\right) \\
&\propto (\tilde{\sigma}_k^2)^{-(a_{\sigma^2} + \frac{1}{2} N \cdot M_k + 1)} \exp\left[-\tilde{\sigma}_k^{-2}\left(b_{\sigma^2} + \frac{1}{2} \sum_{g:\zeta_g = k} \left(y_g - X\tilde{\beta}_k\right)^\top W_g \left(y_g - X\tilde{\beta}_k\right)\right)\right],
\end{aligned}$$

where M_k is the number of ζ_g for which $\zeta_g = k$. Thus, $\tilde{\sigma}_k^2 \overset{ind.}{\sim} \text{IG}(\hat{a}_k, \hat{b}_k)$ where

$$\hat{a}_k = a_{\sigma^2} + \frac{1}{2} N \cdot M_k \quad \text{and} \quad \hat{b}_k = b_{\sigma^2} + \frac{1}{2} \sum_{g:\zeta_g = k} \left(y_g - X\tilde{\beta}_k\right)^\top W_g \left(y_g - X\tilde{\beta}_k\right).$$

Cluster stick-breaking weights, ν

$$\begin{aligned}
p(\nu|\cdot) &= p(\nu|\zeta,\alpha) \propto p(\zeta|\nu)p(\nu|\alpha) \\
&\propto \prod_{g=1}^{G}\left[\nu_{\zeta_g}\prod_{l<\zeta_g}(1-\nu_l)\right]\prod_{k=1}^{K-1}\nu_k(1-\nu_k)^{\alpha} \\
&= \prod_{k=1}^{K-1}\nu_k^{M_k+1}(\nu_k^{T_k+\alpha}),
\end{aligned}$$

where $T_k = \sum_{l>k} M_k$. Thus, $\nu_k \overset{ind.}{\sim} \mathrm{Be}\,(M_k+1, T_k+\alpha)$.

Concentration parameter, α

$$\begin{aligned}
p(\alpha|\cdot) &= p(\alpha|\nu) \propto p(\nu|\alpha)\; p(\alpha) \\
&\propto \prod_{k=1}^{K-1}\left[\alpha(1-\nu_k)^{\alpha}\right]\; p(\alpha) \\
&= \alpha^{K-1}\pi_K^{\alpha}\;\cdot\;\alpha^{a_\alpha-1}\exp(-\alpha b_\alpha) \\
&= \alpha^{a_\alpha+K-2}\exp\left\{-\alpha\left[b_\alpha - \log(\pi_K)\right]\right\}.
\end{aligned}$$

Thus, $\alpha \sim \mathrm{Ga}(a_\alpha + K - 1, b_\alpha - \log\pi_K)$.

APPENDIX C. Definitions used in hierarchical GLFP methodology

Truncation adjusted Kaplan-Meier estimate of lifetime

We first start with a nonparametric estimate of the empirical cdf for each drive-model using the Kaplan-Meier estimator. With left truncation, however, the standard Kaplan-Meier estimator for drive-model g, denoted by $\widehat{F_g(t)}_{KM}$, is conditional on survival up to $t^L_{g,\min}$, the shortest reported running time of all units of drive-model g for which records are available. To produce unconditional estimates, we adapt the adjustment method outlined by Meeker and Escobar (1998) (Chapter 11). For each drive-model we select $t^L_{g,\min}$, the smallest left truncated time in the sample. By sampling from the full posterior distribution, since $\Pr(T > t^L_{g,\min}|\theta_g)$ (the probability a hard drive has survived up to $t^L_{g,\min}$) is a function of the model parameters, we can easily compute its posterior median, $\widehat{A}_{\text{med}} = \widehat{\Pr}(T > t^L_{g,\min}|\theta_g)$. We compute the adjusted estimate by

$$\widehat{F(t)}_{KMadj} = \widehat{A}_{\text{med}} + \left(1 - \widehat{A}_{\text{med}}\right) \widehat{F_g(t)}_{KM},\ t > t^L_{g,\min}.$$

While this adjustment is negligible for the majority of drive-models, five drive-models receive upward adjustments of greater than 5 percent and the estimated time to failure distribution of one drive-model (30) was adjusted by nearly 16 percent, in part because the shortest truncation time for all observed units was approx. 2.3 years.

Definition of global average for Model 4

In Section 4.6.4.1, to illustrate the concept of shrinkage in our hierarchical model, we refer to a "global average" which represents an average GLFP cdf for the entire population, $\mathrm{H}(\cdot|\bar{\pi}, \mu_1, \sigma_1, \bar{\mu}_2, \bar{\sigma}_2)$. Since μ_1 and σ_1 are shared across drive-models, these can already be interpreted as "global". For the parameters that vary across drive models, we select values corresponding to the medians of the hierarchical distributions (4.3) conditional on the hyperparameters. In particular, we set

$$\bar{\pi} = \text{logit}^{-1}(\eta_\pi),\ \bar{\mu}_2 = \eta_{t_2} - m_{\sigma_2}\Phi^{-1}(.2) \text{ and } \bar{\sigma}_2 = m_{\sigma_2}.$$

Let $J(\cdot|a,b)$, $J^{-1}(\cdot|a,b)$ denote the cdf and inverse cdf, respectively, for a log-normal distribution with log-location parameter a and log-scale parameter b. Then

$$m_{\sigma_2} = J^{-1}[0.5 \cdot J(1|\eta_{\sigma_2}, \tau_{\sigma_2})|\eta_{\sigma_2}, \tau_{\sigma_2}],$$

which is the median of a log-normal distribution with parameters η_{σ_2}, and τ_{σ_2}, truncated to the interval $(0, 1)$.

We use posterior draws, $H(\tilde{t}|\eta_{\pi}^{(s)}, \mu_1^{(s)}, \sigma_1^{(s)}, \eta_{t_2}^{(s)}, \eta_{\sigma_2}^{(s)}, \tau_{\sigma_2}^{(s)})$, $s = 1, 2, \ldots, S$ to estimate the global average pointwise, . This computation is similar to that shown in (4.4).

BIBLIOGRAPHY

Antoniak, C. E. (1974). Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, pages 1152–1174.

Backblaze (Accessed April 1, 2016). Backblaze hard drive data sets. https://www.backblaze.com/b2/hard-drive-test-data.html.

Basu, S., Sen, A., and Banerjee, M. (2003). Bayesian analysis of competing risks with partially masked cause of failure. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 52(1):77–93.

Bates, D., Maechler, M., Bolker, B., Walker, S., et al. (2014). lme4: Linear mixed-effects models using Eigen and S4. *R package version*, 1(7):1–23.

Bell, N. and Hoberock, J. (2011). Thrust: A productivity-oriented library for CUDA. *GPU computing gems Jade edition*, 2:359–371.

Berger, J. O. and Sun, D. (1993). Bayesian analysis for the poly-weibull distribution. *Journal of the American Statistical Association*, 88(424):1412–1418.

Betancourt, M. and Girolami, M. (2015). Hamiltonian monte carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, pages 79–101.

Blelloch, G. E. (1990). Prefix sums and their applications.

Chan, V. and Meeker, W. Q. (1999). A failure-time model for infant-mortality and wearout failure modes. *IEEE Transactions on Reliability*, 48(4):377–387.

Datta, S. and Nettleton, D. (2014). *Statistical analysis of next generation sequencing data*. Springer.

Escobar, M. D. (1994). Estimating normal means with a dirichlet process prior. *Journal of the American Statistical Association*, 89(425):268–277.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pages 209–230.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2014a). *Bayesian data analysis*, volume 2. CRC press Boca Raton, FL.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2014b). *Bayesian data analysis*, volume 2. CRC press Boca Raton, FL.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2014c). *Bayesian Data Analysis*, volume 2. Chapman & Hall/CRC Boca Raton, FL.

Gelman, A., Meng, X.-L., and Stern, H. (1996). Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica*, 6:733–760.

Ha, S.-W. and Han, T.-D. (2013). A scalable work-efficient and depth-optimal parallel scan for the gpgpu environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(12):2324–2333.

Harris, M., Sengupta, S., and Owens, J. D. (2007). Parallel prefix sum (scan) with CUDA. *GPU gems*, 3(39):851–876.

Hoberock, J. and Bell, N. (2008). Thrust. (PowerPoint slides).

Hong, Y., Meeker, W. Q., and McCalley, J. D. (2009). Prediction of remaining life of power transformers based on left truncated and right censored lifetime data. *The Annals of Applied Statistics*, 3(2):857–879.

Ihaka, R. and Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314.

Ishwaran, H. and James, L. F. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173.

Ishwaran, H. and Zarepour, M. (2000). Markov chain Monte Carlo in approximate Dirichlet and beta two-parameter process hierarchical models. *Biometrika*, 87(2):371–390.

Landau, W. and Niemi, J. (2016). A fully bayesian strategy for high-dimensional hierarchical modeling using massively parallel computing. *arXiv preprint arXiv:1606.06659*.

Landau, W. M. (2016). *High-dimensional hierarchical models and massively parallel computing.* PhD thesis, Iowa State University.

Law, C. W., Chen, Y., Shi, W., and Smyth, G. K. (2014). Voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*, 15(2):R29.

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):555–576.

Love, M. I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550.

McCarthy, D. J., Chen, Y., and Smyth, G. K. (2012). Differential expression analysis of multifactor rna-seq experiments with respect to biological variation. *Nucleic Acids Research*, 40(10):4288–4297.

McCarthy, D. J. and Smyth, G. K. (2009). Testing significance relative to a fold-change threshold is a TREAT. *Bioinformatics*, 25(6):765–771.

Meeker, W. and Escobar, L. (1998). *Statistical Methods for Reliability Data*. Wiley Series in Probability and Statistics. John Wiley & Sons Hoboken, NJ.

Meeker, W., Hahn, G., and Escobar, L. (2017). *Statistical Intervals: A Guide for Practioners and Researchers*. Wiley Series in Probability and Statistics. John Wiley & Sons, Hoboken, NJ.

Neal, R. M. (2000). Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Nelson, W. B. (1982). *Applied Life Data Analysis*. John Wiley & Sons, Hoboken, NJ.

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, C. (2008). CUBLAS library. *NVIDIA Corporation, Santa Clara, California*, 15(27):31.

Nvidia, C. (2010). CURAND library.

Oshlack, A., Robinson, M. D., and Young, M. D. (2010). From RNA-seq reads to differential expression results. *Genome Biology*, 11(12):220.

Papaspiliopoulos, O. and Roberts, G. O. (2008). Retrospective markov chain monte carlo methods for dirichlet process hierarchical models. *Biometrika*, 95(1):169–186.

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., Best, N., Cowles, K., and Vines, K. (2010). coda: Output analysis and diagnostics for MCMC. R package version 0.14-2.

Ranjan, R., Singh, S., and Upadhyay, S. K. (2015). A Bayes analysis of a competing risk model based on gamma and exponential failures. *Reliability Engineering & System Safety*, 144:35–44.

Reiser, B., Guttman, I., Lin, D. K., Guess, F. M., and Usher, J. S. (1995). Bayesian inference for masked system lifetime data. *Applied Statistics*, 44:79–90.

Robinson, M. D., McCarthy, D. J., and Smyth, G. K. (2010). edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140.

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

Sengupta, S., Harris, M., and Garland, M. (2008). Efficient parallel scan algorithms for GPUs. *NVIDIA, Santa Clara, CA, Tech. Rep. NVR-2008-003*, (1):1–17.

Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica*, pages 639–650.

Smyth, G. K. (2005). Limma: linear models for microarray data. In *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 397–420. Springer.

Stan Development Team (2015). Stan: A c++ library for probability and sampling, version 2.10.0.

Stan Development Team (2016). RStan: the R interface to Stan. `http://mc-stan.org`. R package version 2.14.1.

Suchard, M. A., 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.

US Department of Treasury, I. (2016). Instructions for form 4562. `https://www.irs.gov/pub/irs-pdf/i4562.pdf`.

Vehtari, A., Gelman, A., and Gabry, J. (2016). loo: Efficient leave-one-out cross-validation and WAIC for Bayesian models. https://CRAN.R-project.org/package=loo. R package version 1.1.0.

Vehtari, A., Gelman, A., and Gabry, J. (2017). Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and Computing*, 27(5):1413–1432.

Walker, S. G. (2010). *Bayesian nonparametric methods: motivation and ideas.* Cambridge: Cambridge University Press.

Wang, Z., Gerstein, M., and Snyder, M. (2009). RNA-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63.

Xu, Z., Hong, Y., and Meeker, W. Q. (2015). Assessing risk of a serious failure mode based on limited field data. *IEEE Transactions on Reliability*, 64(1):51–62.