ULTRA-HIGH DIMENSIONAL BAYESIAN VARIABLE SELECTION WITH LASSO-TYPE PRIORS

Can Xu
Southern Methodist University, canxu2014@gmail.com

Follow this and additional works at: https://scholar.smu.edu/hum_sci_statisticalscience_etds

Recommended Citation
Xu, Can, "ULTRA-HIGH DIMENSIONAL BAYESIAN VARIABLE SELECTION WITH LASSO-TYPE PRIORS" (2021). Statistical Science Theses and Dissertations. 27.
https://scholar.smu.edu/hum_sci_statisticalscience_etds/27

This Dissertation is brought to you for free and open access by the Statistical Science at SMU Scholar. It has been accepted for inclusion in Statistical Science Theses and Dissertations by an authorized administrator of SMU Scholar. For more information, please visit http://digitalrepository.smu.edu.
ULTRA-HIGH DIMENSIONAL BAYESIAN VARIABLE SELECTION
WITH LASSO-TYPE PRIORS

Approved by:

__________________________
Dr. Xinlei Wang
Professor in Department of Statistical Science, SMU

__________________________
Dr. Yichen Cheng
Assistant Professor in Robinson College of Business’s Institute, GSU

__________________________
Dr. Daniel F. Heitjan
Professor in Department of Statistical Science, SMU & Population and Data Sciences, UTSW

__________________________
Dr. Lynne Stokes
Professor in Department of Statistical Science, SMU

__________________________
Dr. Guanghua Xiao
Professor in Department of Population and Data Sciences, UTSW
ULTRA-HIGH DIMENSIONAL BAYESIAN VARIABLE SELECTION
WITH LASSO-TYPE PRIORS

A Dissertation Presented to the Graduate Faculty of the
Dedman College
Southern Methodist University
in
Partial Fulfillment of the Requirements
for the degree of
Doctor of Philosophy
with a
Major in Statistics
by
Can Xu

B.A., Civil Engineering, Tongji University
Ph.D., Civil Engineering, University of Houston

December 18, 2021
I would like to acknowledge the support and assistance of my advisor, Professor Xinlei Wang, who deserves all the credit. Without her, this dissertation would not have been possible at all. I would also like to thank all committee members, Dr. Cheng from GSU, Dr. Heitjan, Dr. Stokes, and Dr. Xiao from UTSW. Without their insightful guidance and excellent teaching during my Ph.D study, I would not have made headway in my research. In addition, I would like to thank my family: my parents, my husband Chris, and my daughter Chloe. They are always there for me and provide the greatest joy in my life. I also want to thank all the professors in the department, Sheila, my classmates, my friends, especially Danyi, whose help cannot be overestimated.
Ultra-high dimensional Bayesian variable selection with LASSO-type priors

Advisor: Dr. Xinlei Wang

Doctor of Philosophy degree conferred December 18, 2021
Dissertation completed October 12, 2021

With the rapid development of new data collection and acquisition techniques, high-dimensional data have emerged from various fields. Consequentially, new variable selection methods especially in ultra-high dimensional problems are demanding.

The first part of this dissertation focuses on developing a new Bayesian variable selection method for a differential expression analysis using raw NanoString nCounter data. The medium-throughput mRNA abundance platform NanoString nCounter has gained great popularity in the past decade, due to its high sensitivity and technical reproducibility as well as remarkable applicability to ubiquitous formalin fixed paraffin embedded (FFPE) tissue samples. Based on RCRnorm developed for normalizing NanoString nCounter data [26] and Bayesian LASSO for variable selection [36, 51], we propose a fully integrated Bayesian method, called RCRdiff, to detect differentially expressed (DE) genes between different groups of tissue samples (e.g. normal and cancer). Unlike existing methods that often require normalization performed beforehand, RCRdiff directly handles raw read counts and jointly models the behaviors of different types of internal controls along with DE and non-DE gene patterns. Doing so would avoid efficiency loss caused by ignoring estimation uncertainty from the normalization step in a sequential approach and thus can offer more reliable statistical inference. We also propose clustering-based strategies for DE gene selection, which do not require any external dataset and are free of any arbitrary cutoff. Empirical evidence of the attractiveness of RCRdiff is demonstrated
via extensive simulation and data examples.

The second part of this dissertation proposes a novel Bayesian variable selection method based on empirical likelihood for ultra-high dimensional data. Although a great amount of literature has shown that development of variable selection techniques can enable efficient and interpretable analysis of high dimensional data, variable selection involving ultra-high dimensional data, where the number of covariates $p$ is (much) large than the sample size $n$, remains a highly challenging task. Furthermore, many popular methods based on linear regression models assume Gaussian random noise. In the semi-parametric domain, under the ultra-high dimensional setting, we propose a Bayesian empirical likelihood method for variable selection, which requires no distributional assumptions but only estimating equations. Motivated by Chang et al. [9] on doubly penalized empirical likelihood (EL), we introduce priors to regularize both regression parameters and Lagrange multipliers associated with the estimating equations, to promote sparse learning. We further develop an efficient Markov chain Monte Carlo (MCMC) sampling algorithm based on the active set idea, which has been proved to be useful in reducing computational burden in several existing studies. The proposed method not only inherits merits from both Bayesian and EL inferences, but also has superior performance in both the prediction and variable selection, as shown in our numerical studies.
TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................... ix
LIST OF TABLES ................................................................................................................ xii

CHAPTER

1. RCRDIFF: A FULLY INTEGRATED BAYESIAN METHOD FOR A DIFFERENTIAL EXPRESSION ANALYSIS USING RAW NANOSTRING N COUNTER DATA .......................................................... 1
   1.1. Introduction ........................................................................................................... 1
   1.2. The RCRdiff Method .......................................................................................... 4
       1.2.1. Data model based on RCR ................................................................. 4
       1.2.2. Hierarchical structures for endogenous genes ................................. 5
       1.2.3. Prior specification and posterior computation ............................... 7
       1.2.4. Posterior inference .................................................................................. 9
   1.3. Simulation ............................................................................................................ 11
       1.3.1. Performance evaluation (Simulation I) ........................................... 13
       1.3.2. Sensitivity analysis (Simulation II and III) ....................................... 17
       1.3.3. Integrated vs. sequential approaches (Simulation IV) .................... 20
   1.4. Data Examples .................................................................................................... 21
       1.4.1. Lung cancer data .................................................................................... 22
       1.4.2. Colorectal cancer (CRC) data ........................................................... 25
   1.5. Discussion .......................................................................................................... 27

2. BAYESIAN VARIABLE SELECTION BASED ON EMPIRICAL LIKELIHOOD FOR ULTRA-HIGH DIMENSIONAL DATA ..................................................................................... 30
   2.1. Introduction ....................................................................................................... 30
   2.2. Method .............................................................................................................. 33
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Simulation I for performance evaluation: ROC curves for nine combinations of $P_{DE}$ (DE gene proportion) and $\mu_{DE}$ (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data).</td>
<td>14</td>
</tr>
<tr>
<td>1.2</td>
<td>Simulation I for performance evaluation: average AUC in five settings, each having 9 combinations of $P_{DE}$ (DE gene proportion) and $\mu_{DE}$ (mean DE level). Compared to the reference setting I1, gene expression variability is increased in I2, lane-by-lane variation is increased in I3, probe-level variation is increased in I4 and variability of random noise is increased in I5.</td>
<td>15</td>
</tr>
<tr>
<td>1.3</td>
<td>Simulation II and III for sensitivity analysis: line plots of average AUC under various distributional violations in Panel (a) and under the NanoStringDiff model [73] in Panel (b). In Simulation II, a thick-tail distribution $t_3$ was used in II1 and II2 and a right-skewed distribution $Ga(2, 1)$ was used in II3 and II4 to simulate probe effects and random errors, respectively; in II-5, $t_3$ was used to generate probe effects and $Ga(2, 1)$ to generate random errors; in II6, different $\sigma_e^2$'s were simulated from an inverse gamma distribution for different genes/positive controls. The DE proportion was fixed at 50% and the DE mean at 0.5. Except for the changes above, everything remains the same as in the reference setting I1, including all parameter values (so $t_3$ and $Ga(2, 1)$ need to be rescaled or shifted). In Simulation III, different mean values of the dispersion parameter in the natural log scale, 0, ±0.1, ±0.5, ±1, and ±2, were considered.</td>
<td>18</td>
</tr>
<tr>
<td>1.4</td>
<td>Lung cancer data example: heat map of identified DE genes when comparing FFPE samples of 15 adenocarcinoma and 13 squamous patients. 48 out of 83 genes identified by at least one method are shown above.</td>
<td>24</td>
</tr>
<tr>
<td>1.5</td>
<td>Colorectal cancer data example: Venn Diagram for DE genes detected by five methods in the training set that contains 193 targeted genes.</td>
<td>26</td>
</tr>
</tbody>
</table>
2.1 The flow of our proposed algorithm. It consists of two loops. The inner loop updates $\beta$ and other covariate-level latent variables in the active set. The outer loop updates global hyper-parameters and the active set based on the inner loop results. ........................................ 38

2.2 The average MSE (in the log10 scale) of all methods for Scenarios I–VI. ....... 44

2.3 Average recall, precision, and F scores for Simulation Scenarios I–IV. ........... 48

2.4 Recall, precision, and F scores of all methods in Simulation Scenarios V–VII. .......................................................... 49

2.5 The computational time on the log2 scale of all methods under different $p/n$ ratios. ............................................................... 50

A.1 Simulation I for performance evaluation: computing time (in log2 second) of two Bayesian methods, RCRdiff (using 6,000 MCMC iterations) and NanoStringDiff, under the reference setting I1, where we fix the DE mean and proportion ($p_{DE}$ and $\mu_\beta$) both at 0.5 and vary $I$ and $R$. ....... 59

A.2 Simulation I for performance evaluation: ROC curves for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in Setting I2, where gene expression variability is increased, compared to the reference setting I1. ............................................................. 60

A.3 Simulation I for performance evaluation: ROC curves for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in Setting I3, where lane-by-lane variation is increased is increased, compared to the reference setting I1. ............................................................. 61

A.4 Simulation I for performance evaluation: ROC curves for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in Setting I4, where probe-level variation is increased, compared to the reference setting I1. ............................................................. 62

A.5 Simulation I for performance evaluation: ROC curves for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in Setting I5, where variability of random noise is increased, compared to the reference setting I1. ............................................................. 63

A.6 Lung cancer data example: trace plots of key global parameters (the mean intercept $\mu_a$ and mean slope $\mu_b$, the nonspecific RNA-binding level due to the background noise $c$) and DE levels for three randomly selected regular genes ($\beta_{35}, \beta_{38}$, and $\beta_{77}$) in DE analysis between 28 FF and 28 FFPE samples. Five MCMC chains with over-dispersed starting points are shown using different colors. ................................. 68
A.7 Lung cancer data example: comparative analysis of expression levels of ECHDC2 and PTGDS genes between ADC and SCC patients in the Rousseaux_2013 dataset (with 85 lung ADC patients and 100 lung SCC patients) and the Sato_2013 dataset (with 183 lung ADC patients and 80 lung SCC patients). ................................. 70

A.8 Lung cancer data example: meta-analyses of PTGDS gene expression in tumor vs. normal samples across using six different datasets from the LCE database for lung ADC and SCC patients, respectively. ........ 71
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Simulation I for performance evaluation: F scores based on the FDR threshold 0.05 for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data unless specified otherwise).</td>
</tr>
<tr>
<td>1.2</td>
<td>Simulation IV for comparing integrated vs. sequential approaches: average AUC in the reference setting (parameters estimated from the FFPE lung cancer data unless specified otherwise). The DE mean was fixed at 0.5 and the DE proportion was changed from 0.1 to 0.9.</td>
</tr>
<tr>
<td>1.3</td>
<td>Lung cancer data example: counts of identified DE genes when (i) comparing 28 FF vs. 28 FFPE samples, in which no genes should be DE, and (ii) comparing FFPE samples of 15 adenocarcinoma (ADC) and 13 squamous cell carcinoma (SCC) patients.</td>
</tr>
<tr>
<td>2.1</td>
<td>The average MSE of all methods for Scenario VII.</td>
</tr>
<tr>
<td>2.2</td>
<td>An empirical study using US crime data: average MSPE of each method under settings with different $p/n$ ratios. The average numbers of original and noise variables selected by each method are listed in the parenthesis (original/noise). The smallest average MSPE is highlighted in bold for each setting.</td>
</tr>
<tr>
<td>2.3</td>
<td>An empirical study using US crime data: average MSPE of each method under settings where confounding variables with different correlations with the original variables were generated. The average numbers of original and noise variables selected by each method are listed in the parenthesis (original/noise). The smallest average MSPE is highlighted in bold for each setting.</td>
</tr>
<tr>
<td>2.4</td>
<td>Analysis of the gene expression data: average MSPE of each method.</td>
</tr>
<tr>
<td>A.1</td>
<td>Simulation I for performance evaluation: F scores based on the FDR threshold 0.01 for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data unless specified otherwise).</td>
</tr>
</tbody>
</table>
A.2 Simulation I for performance evaluation: F scores based on the FDR threshold 0.1 for nine combinations of \( p_{DE} \) (DE gene proportion) and \( \mu_{\beta} \) (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data unless specified otherwise). ............ 65

A.3 Simulation I for performance evaluation: F scores based on the FDR threshold 0.15 for nine combinations of \( p_{DE} \) (DE gene proportion) and \( \mu_{\beta} \) (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data unless specified otherwise). ............ 66

A.4 Simulation I for performance evaluation: false positive rates for each method when there is no DE genes for settings I1-I5 and II6. ............ 67

A.5 Lung cancer data example: lists of genes associated with adenocarcinoma and squamous cell carcinoma. ........................................ 69
I dedicate this dissertation to my family.
CHAPTER 1

RCRDIFF: A FULLY INTEGRATED BAYESIAN METHOD FOR A DIFFERENTIAL EXPRESSION ANALYSIS USING RAW NANOSTRING N_COUNTER DATA

1.1. Introduction

The NanoString nCounter system is a popular medium-throughput mRNA abundance platform. It has been widely used in biomedical studies, such as biomarker discovery, gene regulatory pathway analysis, disease diagnosis or prognosis, treatment plan selection, experimental validation in oncology, immunology, and neuroscience, and so on. Unlike traditional RNA profiling techniques, where the original RNA transcripts have to be converted to cDNA and then amplified as a proxy, NanoString nCounter reads the native RNA counts directly. Two probes are hybridized per target transcript in the nCounter system: a capture probe and a fluorescent barcode-labeled reporter probe. The capture probes lock the target RNAs so that a digital analyzer can detect the reporter probes. In that way, it avoids amplification or other enzymatic processing that can introduce substantial bias. NanoString nCounter has been proved to perform well in less ideal conditions, as is often the case with formalin-fixed paraffin-embedded (FFPE) tissues. Given FFPE samples can be stored at room temperature and kept for a long time, more and more researchers have turned to the vast libraries of archival Formalin-Fixed Paraffin-Embedded (FFPE) tissues for molecular analysis. However, there are many difficulties in analyzing FFPE samples, which involve high variability, low expression levels, and high degradation. The NanoString nCounter platform can remedy these issues given the digital quantification has less bias and higher sensitivity to low-abundance genes.
Another advantage of NanoString nCounter is related with the design of internal control probes, which play a vital role in normalizing raw reads. Besides (up to 800) probes for detecting genes of interest, there are six positive control probes, (typically eight) negative control probes, and (as few as three and up to 10+) housekeeping gene probes. The positive controls have different known RNA concentrations, ranging log-linearly from 128 fM to 0.125 fM. They can be used to remove lane-by-lane noise, resulting from variation in experimental conditions (such as humidity, temperature, etc.) between reaction systems. The negative controls have no corresponding target, and are used to assess background noise and non-specific binding; housekeeping genes are expected to remain constant between biological conditions, so they are chosen for their stability to control the amount of RNA.

Normalization is a crucial process for data generated by the nCounter system. It aims to remove systematic biases that affect the measured gene expression levels (e.g., variability in experimental conditions, sample collection and preparation, and machine parameters, etc.), while preserving the variation in gene expression that occurs because of biologically relevant changes in transcription. Using information from the different types of internal controls, several analysis methods have been developed for the nCounter platform to normalize and extract gene expression levels, such as NanoStringNorm [72], NAPPA [24], and NanoStringDiff [73]. Recently, Jia et al. [26] proposed a novel Bayesian method called RCRnorm, which consists of a system of random-coefficient hierarchical regression (RCR) models, attempting to accurately characterize different types of probes. It has been shown that RCRnorm compares favorably with other existing methods, especially for situations with an elevated level of heterogeneity from various sources.

After normalization, differential expression analysis (DEA) can be conducted to discover genes with quantitative changes in expression levels between experimental groups (conditions). Most normalization methods designed for the nCounter platform were published with a suggested DEA method, such as a t-test for NanoStringNorm, a likelihood
ratio test for NanoStringDiff. Note that both of them solve the problem in two steps, which is a strategy most commonly used by researchers even for other types of data. However, the main issue is that the DEA in the second step only relies on estimates from the normalization step and cannot take the estimation uncertainty into account.

Some normalization and differential expression analysis tools, developed for RNA-seq data, can be applied to NanoString nCounter data, too. For example, DESeq [1] has a built-in normalization strategy, Relative Log Expression (RLE); edgeR [57] is based on a normalization method called Trimmed Mean of M values (TMM); and RUVseq [56] is a normalization method that is paired with edgeR or DESeq for DEA, as indicated in the user's manual. However, these methods cannot take advantage of the internal controls designed in the nCounter system, with the exception of RUVseq that can only make use of the negative controls. Besides, all of them are two-step solutions as well.

We aim to integrate Bayesian LASSO into the system of RCR models [26] to identify the differentially expressed (DE) genes from NanoString nCounter data. Bayesian LASSO methods can provide not only closer parameter estimates as compared to original LASSO [68] but also reliable estimates for standard errors that can measure model uncertainty [36, 51]. In the past, such methods have been successfully applied to genomic settings [3, 33, 67]. However, they have not been used with the complex NanoString nCounter system that has a set of internal controls. The major advantage of our proposed method is to allow simultaneous normalization and DEA rather than using a sequential approach, which incorporates all kinds of uncertainties into the final estimates for improved performance in detection of DE genes. Furthermore, Bayesian estimates are more reliable than those from frequentist methods when the sample size is very small, as is typical in many biomedical applications.

The remainder of this paper is organized as follows. In Section 1.2, we describe the fully integrated Bayesian model for RCRdiff, which entails using a system of RCR models modified from RCRnorm in connection with Bayesian LASSO. We also give necessary
details about prior specification and posterior computation, and discuss strategies for statistical inference. Section 1.3 presents four simulation studies to benchmark the proposed RCRdiff with existing approaches under settings that vary key factors to mimic various realistic situations. In Section 1.4, we present two data examples to further illustrate the performance of RCRdiff. Section 1.5 concludes with a brief discussion.

1.2. The RCRdiff Method

1.2.1. Data model based on RCR

The NanoString nCounter system has probes designed for endogenous genes and probes of positive and negative controls. Among those endogenous, there are reference (housekeeping) and regular genes, where regular genes are of main interest in a study and carefully selected to fulfill the research goal. The nCounter system was initially designed to use positive controls, negative controls, and housekeeping genes for quantifying lane effects, non-specific background, and variation in sample loading amounts, respectively. However, information provided by different types of probes is intermingled. Instead of correcting the different biases in an isolated manner, which is typical in existing methods, Jia et al. [26] proposed an integrated system of random-coefficient regression models for jointly modeling log-transformed read counts from the different types of probes. For the $i$th sample ($i = 1, \ldots, I$), let $Y_{ip}^+$, $Y_{in}^-$, $Y_{ih}^\star$, and $Y_{ir}$ denote the log 10 transformed read count of the $p$th positive control ($p = 1, \ldots, P$), the $n$th negative control ($n = 1, \ldots, N$), the $h$th housekeeping gene ($h = 1, \ldots, H$), and the $r$th regular gene ($r = 1, \ldots, R$), respectively. The superscript “+” is used for indicating the membership of the positive control class, “−” for the negative control class, “∗” for the housekeeping gene class, and no superscript for the regular gene class. Note that we add one to the observed counts before applying the logarithm, to avoid $-\infty$ arising from zero counts. Further, let $X_{ip}^+$ represent
the log 10 transformed RNA input amount of the $p$th positive control, which is known and
does not vary across samples (i.e., $X_p^+ = X_p^+$); let $X_{ih}^*$ and $X_{ir}$ represent the (unknown)
log 10 RNA amount of the $h$th housekeeping gene and the $r$th regular gene from the $i$th
sample, respectively. Then, the system of RCR models is given by

$$
Y_{ip}^+ = a_i + b_i X_p^+ + d_p^+ + e_{ip}^+,
$$

$$
Y_{in}^- = a_i + b_i c + d_n^- + e_{in}^-,
$$

$$
Y_{ih}^* = a_i + b_i X_{ih}^* + d_h^* + e_{ih}^*,
$$

$$
Y_{ir} = a_i + b_i X_{ir} + d_r + e_{ir}.
$$

Here, $a_i$ and $b_i$ are the sample-specific random intercept and slope, satisfying $a_i \sim N(\mu_a, \sigma_a)$ and $b_i \sim N(\mu_b, \sigma_b)$; $c$ is an unknown constant, reflecting the nonspecific RNA-binding level due to background noise; $d_p^+$, $d_n^-$, $d_h^*$, and $d_r$ represent the probe-specific systematic deviation from the linear pattern for positive controls, negative controls, housekeeping genes, and regular genes, respectively; Similarly, $e_{ip}^+$, $e_{in}^-$, $e_{ih}^*$, and $e_{ir}$ are the random error terms for the four types of probes. Jia et al.[26] argued that negative controls have no known target, and all detected binding signals should be from nonspecific binding while positive controls, housekeeping and regular genes all have known targets, and so their working mechanisms may be similar. Thus, it is further assumed that $d_p^+, d_h^*, d_r \sim N(0, \sigma_d^2)$ and $e_{ip}^+, e_{ih}^*, e_{ir} \sim N(0, \sigma_e^2)$ while $d_n^- \sim N(0, \sigma_d^-)$ and $e_{in}^- \sim N(0, \sigma_e^-)$.

1.2.2. Hierarchical structures for endogenous genes

For endogenous genes, the latent log RNA amounts $X_{ih}^*$ and $X_{ir}$ for the $i$th sample
are modeled by the following hierarchical linear structures:

$$
X_{ih}^* = \phi_i + \kappa_{ih}^* \quad \text{and} \quad X_{ir} = \phi_i + \kappa_{ir},
$$

where $\phi_i$ is a constant representing the individual effect of sample $i$, satisfying

$$
\sum_i \phi_i = 0; \quad \kappa_{ih}^* \quad \text{and} \quad \kappa_{ir}
$$

reflect the remaining expression abundances after adjusting for the
sample effect $\phi_i$, and these quantities represent the normalized expression levels of the
corresponding genes in RCRnorm.
For regular genes, we assume $\kappa_{ir} \sim N(\mu_{ir}, \sigma^2_{\kappa})$, where the mean $\mu_{ir}$ depends on a binary covariate $Z_i$, indicating which group the $i$th sample is in (0 for the control group, and 1 for the treatment group). In contrast, we assume $\kappa_{ih}^* \sim N(\mu_{h}^*, \sigma^2_{\kappa}^*)$, where $\mu_{h}^*$ is the gene-specific mean that does not depend on $Z_i$. This is because housekeeping genes, by definition, are ubiquitously expressed in all tissue compartments and cell types and minimally fluctuate across all samples.

We consider a simple mean structure for regular genes: $\mu_{ir} = \alpha_r + \beta_r Z_i$, where $\alpha_r$ (or $\alpha_r + \beta_r$) is the mean log 10 RNA amount of the $r$th regular gene in the control (or treatment) group, and $\beta_r$ reflects the treatment effect. For DE genes, $\beta_r \neq 0$ while for non-DE genes, $\beta_r = 0$. To promote the sparsity of these treatment effects, we employ Bayesian LASSO [36] by assigning a conditional Laplace prior to each $\beta_r$, namely

$$
\pi(\beta_r | \sigma^2_{\kappa}) = \frac{\lambda_r}{2\sqrt{\sigma^2_{\kappa}}} \exp \left[ -\frac{\lambda_r |\beta_r|}{\sqrt{\sigma^2_{\kappa}}} \right].
$$

This Laplace distribution has mean 0 and variance $2\sigma^2_{\kappa}/\lambda^2_r$, where $\lambda_r \geq 0$ is a tuning parameter controlling the amount of penalty. Also, (1.1) can be represented by a scale mixture of normals with an exponential density:

$$
\beta_r | \sigma^2_{\kappa}, \tau^2_r \sim N(0, \sigma^2_{\kappa}\tau^2_r), \quad \tau^2_r | \lambda_r \sim \text{Exp}(\lambda_r^2/2),
$$

where $\text{Exp}(\theta)$ denotes an exponential distribution with mean $1/\theta$. Here, a unique tuning parameter $\lambda_r$ is allowed for each coefficient $\beta_r$. This flexibility can produce more accurate estimates as the penalty on zero coefficients needs to be larger than that for nonzero coefficients [36]. Unlike frequentist LASSO [68], we can assign a hyperprior to $\lambda_r$ rather than having to choose $\lambda_r$ explicitly. Here, for analytical tractability, a diffuse gamma prior with shape $\eta$ and rate $\nu$ will be considered [36, 51]. In this way, the full conditional of $\lambda^2_r$ is gamma with shape parameter $1 + \eta$ and rate parameter $\frac{\tau^2_r}{2} + \nu$. 

6
1.2.3. Prior specification and posterior computation

Let $Y$ denote the log 10 read count data observed; let $Z$ denote the collection of all $Z$’s; let $\Theta$ denote the collection of all latent random variables and model parameters involved in Sections 1.2.1 and 1.2.2. We use $\pi(\cdot)$ to denote a general prior distribution, use $N(x|\mu,\sigma^2)$ to denote a normal probability density function (pdf) with mean $\mu$ and variance $\sigma^2$, use $Exp(x|\lambda)$ to denote an exponential pdf with rate $\lambda$, $Ga(x|\alpha,\beta)$ to denote a gamma pdf with shape $\alpha$ and rate $\beta$. Then the full probability model can be written as follows:

$$p(Y, \Theta|Z) \propto \prod_{i=1}^{I} \left\{ \prod_{p=1}^{P} N \left( Y_{ip}^+ | a_i + b_i X_p^+ + d_p^+, \sigma_e^2 \right) \cdot \prod_{n=1}^{N} N \left( Y_{in}^- | a_i + b_i c + d_n^-, \sigma_e^- \right) \right\}$$

$$\cdot \prod_{h=1}^{H} \left[ N \left( Y_{ih}^+ | a_i + b_i (\phi_i + \kappa_{ih}) + d_h^+, \sigma_e^2 \right) \cdot N \left( \kappa_{ih} | \mu_{k_h}, \sigma_{k_h}^2 \right) \right]$$

$$\cdot \prod_{r=1}^{R} \left[ N \left( Y_{ir}^+ | a_i + b_i (\phi_i + \kappa_{ir}) + d_r, \sigma_e^2 \right) \cdot N \left( \kappa_{ir} | \alpha_r + \beta_r Z_i, \sigma_{k_r}^2 \right) \right]$$

$$\cdot N(a_i|\mu_a, \sigma_a^2) \cdot N(b_i|\mu_b, \sigma_b^2)$$

$$\cdot \prod_{p=1}^{P} N(d_p^+|0, \sigma_d^2) \cdot \prod_{n=1}^{N} N(d_n^-|0, \sigma_{d^-}^2) \cdot \prod_{r=1}^{R} N(d_r|0, \sigma_d^2) \cdot \prod_{h=1}^{H} N(d_h^+|0, \sigma_d^2)$$

$$\cdot \prod_{r=1}^{R} \left[ \pi(\alpha_r) \cdot N \left( \beta_r | 0, \sigma_{\beta_r}^2 \nu_r \right) \cdot Exp \left( \tau_r^2 | \lambda_r^2/2 \right) \cdot Ga \left( \left( \lambda_r^2 | \eta, \nu \right) \right) \right]$$

$$\cdot \pi(\mu_a) \cdot \pi(\mu_b) \cdot \pi(c) \cdot \prod_{i=1}^{I} \pi(\phi_i) \cdot \prod_{h=1}^{H} \pi(\mu_{k_h})$$

$$\cdot \pi(\sigma_a^2) \cdot \pi(\sigma_b^2) \cdot \pi(\sigma_{k_h}^2) \cdot \pi(\sigma_{d_r}^2) \cdot \pi(\sigma_{d^-}^2) \cdot \pi(\sigma_e^2) \cdot \pi(\sigma_{e^-}^2),$$

where the first five lines represent the joint likelihood of the observed data and latent random variables, the sixth line represents prior distributions of the regression parameters for modeling DE vs. non-DE genes and hyperprior distributions introduced by Bayesian LASSO; the last two lines represent prior distributions of the location and variance parameters in the RCR models, respectively.
Compared to RCRnorm, we have to specify priors for two extra parameters of each regular gene (i.e., $\alpha_r$ and $\lambda_2^r$), introduced by Bayesian LASSO for gene selection. For the intercept $\alpha_r$, we use a non-informative uniform distribution, with lower and upper bounds given by $\text{mean}(\hat{X}_{r0}) \pm m \times \text{sd}(\hat{X}_{r0})$, where $m$ is a user specified constant (e.g., 3, 5) to make the prior much more diffuse than what data suggest; $\hat{X}_{r0} = \{\hat{X}_{ir} : 1 \leq i \leq I \text{ and } Z_i = 0\}$ and $\hat{X}_{ir} = (Y_{ir} - \hat{a}_i)/\hat{b}_i$ with $\hat{a}_i$ and $\hat{b}_i$ being the (least square) estimated intercept and slope from fitting $Y_{ip}^+$ vs. $X_{ip}^+$ for each sample $i$ in the control group. In the gamma prior for the gene-wise penalty $\lambda_2^r$, $\eta$ and $\nu$ can be fixed at some small values (e.g., 0.1, 0.01). Actually, the choice of $\eta$ and $\nu$ would not matter much – the deeper in the hierarchy, the less impact these parameters have on the final inference [34].

For other parameters, we use the same prior distributions as specified in RCRnorm: an inverse gamma prior $IG(\delta, \zeta)$ is applied to all variance parameters, where $\delta$ and $\zeta$ are small positive numbers to make the prior diffuse (e.g., $\delta = \zeta = 0.01$); for $\mu_a$ and $\mu_b$, we consider vague normal priors $\mu_a \sim N(\hat{\mu}_a, m \times \text{se}(\hat{\mu}_a))$ and $\mu_b \sim N(\hat{\mu}_b, m \times \text{se}(\hat{\mu}_b))$, where $\hat{\mu}_a = \sum_{i=1}^I \hat{a}_i/I$ and $\hat{\mu}_b = \sum_{i=1}^I \hat{b}_i/I$; for any other location parameter (say $\theta$), a uniform prior $U(L_\theta, U_\theta)$ is used to provide a sufficiently wide coverage for all plausible values of $\theta$ suggested by data, for example, $c \sim U(-6, -1)$. For more detail, see Section 4.2 in [26].

We use a Gibbs sampler, the most commonly used Markov chain Monte Carlo (MCMC) algorithm, to draw samples from the joint posterior $p(\Theta|Y, Z)$. Bayesian LASSO is implemented through an inner loop embedded in each iteration of the Gibbs sampler. This allows a variable selection procedure that would increase the penalty for non-DE genes and shrink the value of $\beta_r$ of such genes close to zero. Given all remaining random variables and parameters sampled in each iteration, we generate $m$ samples of $\beta_r, \tau_r,$ and $\lambda_r$ (say $m = 50$ or 100). The median of these inner samples in the current iteration is used as the sampled value of $\beta_r$ for the next iteration. The full conditionals are listed in Section A.1 of Appendix. We can see that with the added LASSO component, all posterior conditionals are still known distributions, which guarantees an efficient Gibbs sampler that is
easy to implement.

We use trace plots and the Gelman–Rubin diagnostic [21] to detect the convergence of MCMC chains, as illustrated in Section A.3 of Appendix using the lung cancer data in Section 1.4.1. The posterior mean $\hat{\beta}_r$ is taken as the final estimate of $\beta_r$, on which DEA is based.

1.2.4. Posterior inference

Although Bayesian LASSO can shrink the treatment effect estimates for non-DE genes, we cannot get exact zeros as in the frequentist LASSO. Here, we discuss several methods to make the posterior inference about DE gene selection. The first is just ranking regular genes based on the absolute values of $\hat{\beta}_r$ from the highest to the lowest, then the top ranking genes should be identified as DE. Researchers can select how many top ranking genes for biological followup based on the resources available. The second method is to select DE genes based on an independent validation data set. For example, one can use different numbers of top ranking genes to train classification models (e.g., random forests, support vector machine, neural networks) to predict relevant clinical outcomes. The set of genes that gives the best predictive performance can then be selected as the DE genes.

The last method, also the one we applied in our simulation and real data analysis, is clustering based on $|\hat{\beta}_r|$’s, the absolute values of obtained posterior means of $\beta_r$’s. If there is only one cluster suggested by data, we then refer it to as a supercluster, containing either all DE genes or all non-DE genes. If there are more than one cluster, we further bisect them into two superclusters, one being the last cluster consisting of genes with the lowest $|\hat{\beta}_r|$’s and the other being the union of all other clusters. Finally, to label a supercluster, we check whether the 80th percentile of standardized beta estimates (SBEs), defined as $|\hat{\beta}_r|/\hat{\sigma}_{\beta_r}$ for gene $r$, is less than 1, where $\hat{\sigma}_{\beta_r}$ is the sample standard deviation of posterior draws of $\beta_r$. If yes, then all genes in the corresponding supercluster would be classified
as non-DE; otherwise they would be classified as DE.

As will be shown in Section 1.3.1, this strategy tends to produce more false positives when the DE gene proportion is low. Alternatively, if there are three or more clusters, we can apply the strategy of Silhouette scanning to bisect the clusters of genes [25]. Silhouette [59] is a statistic in the range \([-1, 1]\), which is used to measure how well each object (or data point) has been grouped. For data point \(i\), let \(C_i\) denote the cluster that it is assigned to, and \(d(i, j)\) denote the distance between points \(i\) and \(j\). Then Silhouette is defined as

\[
s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},\]

where

\[
a(i) = \frac{\sum_{j \in C_i, i \neq j} d(i, j)}{|C_i| - 1}
\]

is the mean distance between point \(i\) and any other point in the same cluster, and

\[
b(i) = \min_{k \neq i} \frac{\sum_{j \in C_k} d(i, j)}{|C_k|}
\]

is the smallest mean distance of point \(i\) to all points in any other cluster. The closer to one the average Silhouette of all data points is, the more appropriately the data is clustered. For \(k\) clusters of univariate data \((k \geq 3)\), there are naturally \(k - 1\) cutoffs for bisection. The genes are bisected at each cutoff and the corresponding average silhouette is computed. The one leading to the largest average Silhouette is used as the final threshold for creating two superclusters, and as mentioned above, each supercluster is classified as DE or non-DE using the 80th percentile of SBEs. Our simulation in Section 1.3.1 shows that this scanning strategy works better than the first strategy when the DE gene proportion is low.

We mention that a binary rule is imposed to label the two superclusters, namely, “the \(q\)th percentile of the SBEs in a supercluster is smaller than \(z\)” or not. Intuitively, the rule should not be overly stringent or sensitive, due to the following reasons. First of all, in our numerical experiments of an earlier version of this paper, we find that, even when we did not apply any rule and simply labeled the supercluster with smaller \(|\hat{\beta}_r|\)'s as non-DE and the other as DE, RCRdiff worked well in all the non-null settings. This may suggest no need for a stringent rule. Secondly, it has been reported in the literature that for LASSO based methods, standard error estimates can be unstable [30], and so we base the threshold on a percentile of all genes in a supercluster for more robustness.
Thus, our rule is intuition-driven, and the specific choice $q = 80$ and $z = 1$ is further established based on empirical evidence from our extensive numerical studies reported in Sections 1.3 and 1.4. Results from simulation and real data show that this explicit rule worked quite well.

For medium-throughput experiments using the NanoString nCounter platform, the targeted genes are usually carefully chosen based on some preliminary study or prior biological knowledge, and so the DE gene proportion may be much higher than whole-genome high-throughput experiments. Thus, in our implementation of RCRdiff, we set the first strategy as the default option. If Silhouette scanning is used for bisection, we use RCRdiff-S to denote our proposed method instead. Evidently, our method of choosing a cutoff for DE gene selection avoids the need of any external dataset and is data-adaptive.

There are many clustering methods, but the optimal number of clusters $k$ suggested by most methods is at least two, i.e. $k \geq 2$. However, in our problem, it is reasonable to allow the case that all genes are DE or non-DE, i.e. $k = 1$. Therefore, we choose a model-based clustering method proposed by Fraley and Raftery[18]. This clustering method assumes data are from a normal mixture model, where a model with $k$ distinct normal components implies $k$ clusters. The final optimal model and number of clusters can be found using modified Bayesian Information Criterion. For more detail, we refer readers to Fraley and Raftery’s paper[18]. We employ this method using an R package named “mclust”. For univariate data, one can assume a common variance for all clusters or different variances across clusters. We adopt the latter as it is more appropriate in our problem.

1.3. Simulation

We conducted simulation to illustrate the performance of the proposed method in DEA and compare it with nine existing approaches, including NanoStringDiff, NanoStringNorm+t-
test, NAPPA+t-test, NAPPA+DESeq, NAPPA+edgeR, RLE+DESeq, TMM+edgeR, RUVseq+DESeq, and RUVseq+edgeR. NanoStringDiff is a method for detecting DE genes without requiring normalized data as input, because normalization is incorporated in its model framework, from which size factors and background noise are estimated before a likelihood ratio test is conducted for DEA. NanoStringNorm is a normalization method, and a t-test is built in its R package for DEA. NAPPA is also a normalization method, but we considered three combinations for DEA since there is no recommendation in its user’s manual or package vignette. RLE+DESeq and TMM+edgeR are the original DESeq and edgeR, two DEA methods, with their built-in normalization strategies. RUVseq is a normalization method, and we considered two combinations for DEA, both suggested by the user’s manual. We did not consider the new method RUV-III [41] given this method requires replicates in the data.

We designed four simulation studies, each with different settings, in which model parameters, unless specified otherwise, were estimated/modified from the FFPE lung cancer data used in Section 1.4.1. We varied key factors to mimic various scenarios that could occur in practical situations, or used different data-generating models for sensitivity analysis. For each unique scenario, 30 replicates were independently simulated, each with 28 samples (14 for each group), six positive controls, eight negative controls, seven housekeeping and 83 regular genes, as in the lung cancer example. All the methods mentioned above were applied. For RCRdiff, 6000 iterations were simulated in each MCMC run and the first 3000 were used as the burn-in period. Also, RCRdiff, as a Bayesian method, gives $\hat{\beta}_r$’s, the estimates of the DE levels, whereas the other methods give p-values. The receiver operating characteristic (ROC) curve and the area under the ROC curve (AUC) based on $|\hat{\beta}_r|$’s were used to assess the overall performance of the methods without requiring any specific cutoff. The F score, defined as $F_1 = 2 \times \text{(precision} \times \text{recall)}/(\text{precision} + \text{recall})$, is used to compare the accuracy of the methods in identifying DE genes, which combines precision and recall, attempting to balance the tradeoff between false negatives and false positives.
1.3.1. Performance evaluation (Simulation I)

In Simulation I, we simulated data based on the RCR models in Section 1.2 and considered five settings (I1-I5) to compare the performance of the methods. The first setting I1 serves as a reference setting, in which we adopted parameters estimated from the FFPE lung cancer data. In Setting I2, we simulated larger variability in expression levels, therefore, $\sigma_k$ and $\sigma_k*$ were increased to two times of the reference setting. The third setting I3 mimics the situation with poor control of experiment conditions that leads to higher lane-by-lane variation, where $\sigma_a$ and $\sigma_b$ were increased to two times of the reference setting. In I4, we increased $\sigma_d$ and $\sigma_{d-}$ to two times of the reference setting, to mimic larger variance in probe affinity, which can result from a poor probe design. In I5, we increased $\sigma_e$ and $\sigma_{e-}$ to two times of the reference setting to study the influence of larger random errors. In each setting, we first varied $p_{DE}$ (the proportion of DE genes) and $\mu_\beta$ (the mean of the DE levels $\beta_r$’s), and considered nine combinations, where three values, 0.3, 0.5, and 0.7, were considered for both $p_{DE}$ and $\mu_\beta$. We set $1/3$ of the DE genes to be down-regulated, and then generated $\beta_r \sim N(\mu_\beta, 0.1)$ for up-regulated genes and $\beta_r \sim N(-\mu_\beta, 0.1)$ for down-regulated genes. For the non-DE genes, we set $\beta_r \equiv 0$. Besides the above nine combinations, we considered the null case under each setting to examine the false positive rate, in which no DE gene is present (i.e., $p_{DE} = 0$ and $\mu_\beta = 0$).

All the methods assume a common sample effect $\phi_i$ for all genes in a given sample $i$, to account for between-sample variations resulted from loading amounts or RNA degradation of different samples. This implies different genes in sample $i$ have the same degradation level, which cannot be true, especially for FFPE samples. Therefore, to accommodate gene-wise RNA degradation, we followed settings in Jia et al.’s paper [26] and added white noise $\omega_{ir}$ and $\omega_{ih}$ from $N(0, 0.4^2)$ to $\phi_i$ for regular and housekeeping genes in all five settings.
Figure 1.1: Simulation I for performance evaluation: ROC curves for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data).

ROC curves for the reference setting are shown in Figure 1.1. Obviously, RCRdiff outperforms the other methods under all combinations of DE means and proportions. The DE proportion $p_{DE}$ does not affect the performance as much as the DE mean $\mu_\beta$. When $\mu_\beta$ is small, all methods give similar results except for RCRdiff. As $\mu_\beta$ increases, they have
improved performance but are still beaten by RCRdiff. Among existing methods, NanoStringDiff, NanoStringNorm+t, and NAPPA+t have better results when $\mu_\beta$ is not that small. That is because these methods use information from internal controls of the NanoString nCounter system in the normalization step. NAPPA+DESeq and NAPPA+edgeR give worse results than NAPPA+t, which is perhaps because the former two methods conduct normalization twice (recall both DESeq and edgeR have built-in normalization steps). As for the methods not designed for NanoString data, RUVseq+DESeq gives similar results as original DESeq and edgeR (i.e., RLE+DESeq and TMM+edgeR), but results from RUVseq+edgeR are much worse.

Figure 1.2: Simulation I for performance evaluation: average AUC in five settings, each having 9 combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level). Compared to the reference setting I1, gene expression variability is increased in I2, lane-by-lane variation is increased in I3, probe-level variation is increased in I4 and variability of random noise is increased in I5.
ROC curves of settings I2-I5 are reported in Section A.2 of Appendix due to space limit. Here, we report the line plots of average AUC for all five settings in Figure 1.2. First of all, RCRdiff has the highest AUC in all individual scenarios. Secondly, the effect of $p_{DE}$ is much smaller than that of $\mu_3$. Increasing the DE mean increases the AUC for each method, as all line plots seem to have sizable jumps at different mean values. Increasing the DE proportion decreases the AUC for methods using DESeq or edgeR as the DEA tool while the AUC for those using a t-test or likelihood ratio test is less affected. This is perhaps because one major assumption of TMM and RLE, the normalization methods built in DESeq and edgeR, is low DE proportion.

Table 1.1 compares the accuracy of the methods for DE gene selection using F scores. For the nine existing approaches, p-values of all regular genes were ranked from smallest to largest and adjusted using the FDR (false discovery rate) control method [5]; then top-ranked DE genes were chosen using the common threshold $FDR \leq 0.05$. As described in Section 1.2.4, our proposed method has two data-adaptive strategies to select DE genes, where RCRdiff uses the default strategy and RCRdiff-S uses Silhouette scanning. From Table 1.1, we observe that the winner is either RCRdiff or RCRdiff-S in all scenarios. RCRdiff-S outperforms RCRdiff when the DE gene proportion $p_{DE}$ is small, but RCRdiff is better when $p_{DE}$ is large. We further varied the FDR threshold from the common choice 0.05 to 0.01, 0.1 and 0.15 for the existing methods, and report F scores in Tables A.1–A.3 of Appendix. We find that increasing the FDR threshold can increase the F scores of the existing methods; however, both RCRdiff and RCRdiff-S are always among the top performers in all cases.

Finally, we report false positive rates by each method for null cases of all five settings (i.e., no DE genes) in Table A.4 of Appendix. We can see that RCRdiff has competitive performance in all null cases. RCRdiff-S has more false positives than RCRdiff but still at an acceptable level.
\[
\begin{array}{|c|ccc|ccc|ccc|}
\hline
 & \mu_\beta = 0.3 & & \mu_\beta = 0.5 & & \mu_\beta = 0.7 \\
 & p_{DE} & 0.3 & 0.5 & 0.7 & p_{DE} & 0.3 & 0.5 & 0.7 & p_{DE} & 0.3 & 0.5 & 0.7 \\
\hline
\text{RCRdiff} & 0.35 & 0.60 & 0.64 & 0.60 & 0.77 & 0.80 & 0.67 & 0.85 & 0.86 \\
\text{RCRdiff-S} & 0.43 & 0.58 & 0.61 & 0.68 & 0.77 & 0.79 & 0.79 & 0.86 & 0.84 \\
\text{NanoStringDiff} & 0.33 & 0.38 & 0.41 & 0.60 & 0.66 & 0.72 & 0.77 & 0.84 & 0.86 \\
\text{NanoStringNorm+t} & 0.01 & 0.03 & 0.02 & 0.15 & 0.27 & 0.39 & 0.57 & 0.68 & 0.76 \\
\text{NAPPA+t} & 0.12 & 0.16 & 0.12 & 0.39 & 0.45 & 0.56 & 0.69 & 0.75 & 0.82 \\
\text{NAPPA+edgeR} & 0.26 & 0.32 & 0.32 & 0.59 & 0.62 & 0.63 & 0.75 & 0.79 & 0.78 \\
\text{NAPPA+DESeq} & 0.32 & 0.34 & 0.32 & 0.59 & 0.62 & 0.62 & 0.75 & 0.79 & 0.77 \\
\text{TMM+edgeR} & 0.30 & 0.35 & 0.36 & 0.59 & 0.65 & 0.66 & 0.78 & 0.82 & 0.83 \\
\text{RLE+DESeq} & 0.32 & 0.37 & 0.38 & 0.62 & 0.66 & 0.67 & 0.79 & 0.82 & 0.83 \\
\text{RUV+edgeR} & 0.29 & 0.33 & 0.35 & 0.57 & 0.62 & 0.61 & 0.73 & 0.73 & 0.69 \\
\text{RUV+DESeq} & 0.31 & 0.38 & 0.37 & 0.61 & 0.67 & 0.67 & 0.78 & 0.82 & 0.81 \\
\hline
\end{array}
\]

Table 1.1: Simulation I for performance evaluation: F scores based on the FDR threshold 0.05 for nine combinations of \( p_{DE} \) (DE gene proportion) and \( \mu_\beta \) (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data unless specified otherwise).

1.3.2. Sensitivity analysis (Simulation II and III)

In Simulation II, we examined the robustness of RCRdiff by considering six settings (II1-II6) modified from the reference setting I1. In the first five settings, the normality assumptions of probe effects and/or random errors were violated. In II1 and II2, we generated probe effects and random errors using a thick-tailed distribution \( t_3 \), respectively. In II3 and II4, we adopted a right-skewed distribution \( Ga(2,1) \) to generate probe effects and random errors, respectively. In II5, \( t_3 \) was used to generate the probe effects and \( Ga(2,1) \) to generate random errors. Note that data generated from \( t_3 \) were rescaled and data generated from \( Ga(2,1) \) were shifted and rescaled, to achieve mean zero and the same variance as in I1. In the last setting II6, we allowed different \( \sigma_e^2 \)'s for different genes/positive...
controls, which violates the equal-variance assumption assumed by the RCR models in Section 1.2.1. We generated $\sigma^2_e$’s from an inverse gamma distribution, whose shape and scale parameters were roughly estimated from FFPE lung cancer data in Section 1.4.1 using the method of moments. Here, we fixed the DE gene proportion $p_{DE}$ at 50% and the DE mean $\mu_\beta$ at 0.5.

Figure 1.3: Simulation II and III for sensitivity analysis: line plots of average AUC under various distributional violations in Panel (a) and under the NanoStringDiff model [73] in Panel (b). In Simulation II, a thick-tail distribution $t_3$ was used in II1 and II2 and a right-skewed distribution $Ga(2, 1)$ was used in II3 and II4 to simulate probe effects and random errors, respectively; in II-5, $t_3$ was used to generate probe effects and $Ga(2, 1)$ to generate random errors; in II6, different $\sigma^2_e$’s were simulated from an inverse gamma distribution for different genes/positive controls. The DE proportion was fixed at 50% and the DE mean at 0.5. Except for the changes above, everything remains the same as in the reference setting I1, including all parameter values (so $t_3$ and $Ga(2, 1)$ need to be rescaled or shifted). In Simulation III, different mean values of the dispersion parameter in the natural log scale, $0, \pm 0.1, \pm 0.5, \pm 1, \pm 2$, were considered.

In Simulation III, we simulated data from a convolution model [73], which is substantially different from our proposed model. That is, $Y_{ir}$, the read count of the $r$th regular gene in the $i$th sample, was simulated from the convolution of a negative binomial dis-
tribution and a Poisson distribution, namely, \( Y_{ir} \mid \mu_{ir}, \eta_{r}, \theta_i \sim NB (v_{ir}, \eta_{r}) + Pois(\theta_i) \), where \( NB(v_{ir}, \eta_{r}) \) denotes a negative binomial distribution with mean \( v_{ir} = c_i d_i \mu_{ir} \) and variance \( v_{ir} + v_{ir}^2 \cdot \exp(\eta_r) \), \( Pois(\theta_i) \) denotes a Poisson distribution with rate \( \theta_i \), \( \mu_{ir} \) is the mean expression rate, \( \eta_r \) is the dispersion parameter in the natural log scale (and thus, if \( \eta_r \to -\infty \), the negative binomial distribution converges to a Poisson distribution), and \( c_i, d_i \), and \( \theta_i \) are the internal control size factors. For DE genes, we set \( p_{DE} \) to 0.5; we generated \( \ln \beta_{ir} \), the DE level in the natural log scale, from a mixture distribution \( 0.7N(1.15, 0.23) + 0.3N(-1.15, 0.23) \), and then added \( \ln \beta_{ir} \) to \( \ln \mu_{ir} \); for non-DE genes, we set \( \ln \beta_{ir} \equiv 0 \). Note that \( \ln(10^{0.5}) \approx 1.15 \) and \( \ln(10^{0.1}) \approx 0.23 \), indicating a match with the case of \( \mu_\beta = 0.5 \) in the reference setting I1. We generated \( \eta_r \) from \( N(\mu_\eta, \sigma_\eta^2) \), where nine values of \( \mu_\eta \), 0, \( \pm 0.1 \), \( \pm 0.5 \), \( \pm 1 \), and \( \pm 2 \), were considered. Here, a larger \( \mu_\eta \) value indicates stronger over-dispersion. All the other parameters involved were estimated from the lung cancer data. Read counts of negative controls, positive controls and housekeeping genes were generated in the same manner as in Wang et al.’s paper [73].

As in Simulation I, to mimic varying degradation levels across genes, we added white noise from \( N(0, 0.4^2) \) to the sample effect \( \phi_i \) of each endogenous gene in Simulation II, and white noise from \( N(0, 0.92^2) \) in Simulation III. Line plots of average AUC are shown in Figure 1.3(a) for Simulation II along with the reference setting I1 and in Figure 1.3(b) for Simulation III. From Panel (a), we can see that RCRdiff is the least sensitive to the distributional disturbances among all. RCRdiff shows a significant advantage over the other methods in all these settings, meaning that our method is favorable even in presence of (moderate) violations of the normality or equal-variance assumptions. Panel (b) shows that as the over-dispersion becomes more severe, the performance of all methods decreases. Amazingly, RCRdiff maintains its strong performance even when data were not generated from our proposed RCR model system – when the dispersion parameter is relatively low, it is better than other methods and is comparable otherwise. Among existing methods, the model from NanoStringDiff was used to generate data in Simulation III. NanoStringDiff does perform better than most other methods for data with low over-
dispersion; however its performance deteriorates as the over-dispersion becomes large. This is perhaps due to the white noise added to the NanoStringDiff model for simulating gene-wise degradation, which has not been addressed by any of the methods using current data [26].

As requested by one of the reviewers, we report false positive rates by each method for the null case of II-6 in Table A.4 of Appendix. We can see that the results are not changed much as compared to those in I1-I5, suggesting that all the methods including ours are not sensitive to the violation of equal-variance assumption in the null setting.

1.3.3. Integrated vs. sequential approaches (Simulation IV)

We have observed from Simulation I–III that RCRdiff has superior performance compared to existing methods in various simulation settings and is robust against different types of disturbance. We conjecture that the improvement is from two aspects: (i) RCRdiff is built on RCRnorm, which has been shown to outperform other normalization methods [26]; (ii) RCRdiff employs a fully integrated approach for conducting normalization and gene selection simultaneously, as opposed to a sequential approach adopted by all existing methods.

In Simulation IV, to verify that the integrated approach can utilize information more efficiently, we compare RCRdiff with RCRnorm+t-test using the reference setting I1, where we fixed the DE mean $\mu_\beta$ at 0.5 but varied the DE proportion $p_{DE}$ from 0.1 to 0.9 with step size 0.2. Both methods rely on RCRnorm for normalization but the latter, like other existing methods, conducts normalization first, followed by DEA using a t-test. Table 1.2 reports the average AUC for RCRdiff and RCRnorm+t-test, along with other existing methods. Clearly, RCRdiff is the best, followed by RCRnorm+t, and then by other existing methods. This finding agrees well with our conjecture. Also, RCRdiff, NanoStringDiff and NanoStringNorm+t appear to be little affected by the DE proportion. Yet another interesting
observation is, as \( p_{DE} \) increases, the gain from using RCRdiff over RCRnorm+t increases as well. This is expected because data from two groups become similar to data from only one group when \( p_{DE} \) is low.

<table>
<thead>
<tr>
<th>DE proportion ( p_{DE} )</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCRdiff</td>
<td>0.92</td>
<td>0.91</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>RCRnorm+t</td>
<td>0.91</td>
<td>0.89</td>
<td>0.87</td>
<td>0.85</td>
<td>0.83</td>
</tr>
<tr>
<td>NanoStringDiff</td>
<td>0.81</td>
<td>0.81</td>
<td>0.80</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td>NanoStringNorm+t</td>
<td>0.82</td>
<td>0.82</td>
<td>0.83</td>
<td>0.83</td>
<td>0.82</td>
</tr>
<tr>
<td>NAPPA+t</td>
<td>0.77</td>
<td>0.80</td>
<td>0.82</td>
<td>0.72</td>
<td>0.80</td>
</tr>
<tr>
<td>NAPPA+edgeR</td>
<td>0.83</td>
<td>0.80</td>
<td>0.78</td>
<td>0.72</td>
<td>0.69</td>
</tr>
<tr>
<td>NAPPA+DESeq</td>
<td>0.85</td>
<td>0.80</td>
<td>0.77</td>
<td>0.73</td>
<td>0.71</td>
</tr>
<tr>
<td>TMM+edgeR</td>
<td>0.85</td>
<td>0.82</td>
<td>0.80</td>
<td>0.76</td>
<td>0.72</td>
</tr>
<tr>
<td>RLE+DESeq</td>
<td>0.85</td>
<td>0.82</td>
<td>0.79</td>
<td>0.75</td>
<td>0.73</td>
</tr>
<tr>
<td>RUV+edgeR</td>
<td>0.84</td>
<td>0.80</td>
<td>0.75</td>
<td>0.68</td>
<td>0.63</td>
</tr>
<tr>
<td>RUV+DESeq</td>
<td>0.86</td>
<td>0.82</td>
<td>0.79</td>
<td>0.75</td>
<td>0.72</td>
</tr>
</tbody>
</table>

Table 1.2: Simulation IV for comparing integrated vs. sequential approaches: average AUC in the reference setting (parameters estimated from the FFPE lung cancer data unless specified otherwise). The DE mean was fixed at 0.5 and the DE proportion was changed from 0.1 to 0.9.

1.4. Data Examples

We present two data examples to further illustrate the performance of our proposed method and to benchmark its performance against five existing methods including NanoStringDiff, NanoStringNorm+t-test, NAPPA+t-test, TMM+edgeR, and RLE+DESeq. They were selected because of their better performance in our simulation than the other existing methods.
In the first example, we analyzed the lung cancer data in Jia et al.'s paper [26], which motivated the development of RCRnorm. The same data were used in Xie et al.'s paper [74] for validation of a 12-gene signature for predicting Adjuvant ChemoTherapy (ACT) response in lung cancer. Given the response of lung cancer patients to ACT varies widely, the 12-gene signature was developed from freshly frozen (FF) tissue samples to identify who would not benefit from the treatment to avoid adverse effects [66]. The dataset was collected to verify that gene expression measurements of FFPE samples are reliable and consistent with those from paired FF samples, so that the 12-gene signature derived from FF samples can be applied to FFPE samples as well. It contains NanoString nCounter data of paired FF and FFPE samples from 30 lung cancer patients, including two subtypes of the cancer: 16 adenocarcinoma (ADC) and 14 squamous cell carcinoma (SCC) patients. The nCounter system involved has 87 targeted genes, including the 12-gene signature and other carefully selected genes. Besides, the system has 6 positive controls, 8 negative controls, and 7 housekeeping genes. Following Jia et al.'s paper [26], for the purpose of quality control, we removed one patient from each cancer subtype and four genes given their mean read counts are lower than the maximum count of negative controls.

<table>
<thead>
<tr>
<th>DEA</th>
<th>RCRdiff</th>
<th>NanoStringDiff</th>
<th>NanoStringNorm+t</th>
<th>Nappa+t</th>
<th>TMM+edgeR</th>
<th>RLE+DESeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFPE vs FF</td>
<td>0</td>
<td>34</td>
<td>32</td>
<td>47</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>ADC vs SCC</td>
<td>45</td>
<td>32</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>39</td>
</tr>
</tbody>
</table>

Table 1.3: Lung cancer data example: counts of identified DE genes when (i) comparing 28 FF vs. 28 FFPE samples, in which no genes should be DE, and (ii) comparing FFPE samples of 15 adenocarcinoma (ADC) and 13 squamous cell carcinoma (SCC) patients.

We conducted two analyses. In the first, we compared 28 FF samples with 28 FFPE samples, in which the ground truth should be no DE genes as they are from the same
set of patients. Although FF and FFPE samples were paired for each patient, this dependence was ignored and data were treated as two independent groups when performing DEA. Convergence diagnostics for RCRdiff are illustrated through this analysis and the detail can be found in Section A.3 of Appendix. The optimal number of clusters obtained from RCRdiff is \( k = 1 \) and 80% of SBEs of this one cluster are less than 1, so all genes are labeled as non-DE. For the other five methods, top-ranked DE genes were chosen to control \( \text{FDR} \leq 0.05 \) as mentioned in Section 1.3.1. In Table 1.3, which reports the number of identified DE genes for each method, only RCRdiff gives no false positive while the other methods give 30+ false positives.

In the second analysis, we conducted DEA by comparing FFPE samples of 15 ADC and 13 SCC patients. RCRdiff returns \( k = 2 \), and after checking the SBEs, genes in the cluster with the smaller DE mean were labeled as non-DE. In this example, RCRdiff and RCRdiff-S give the same results as the optimal number of clusters obtained from either analysis is less than three. Figure 1.4 shows the identified DE genes for all six methods. It is interesting to observe that RCRdiff identified the largest number of DE genes, followed by the original DESeq and then edgeR. In the original lung cancer study [74], 33 out of the 83 genes were included for gene expression profiling because they have been reported in the literature to be consistently associated with patient survival outcomes in either lung ADC or SCC (see Table A.5 in Appendix). Among those 33 genes, RCRdiff finds 18, while NanoStringDiff detects 10, NanoStringNorm+t detects 7, NAPPT+t detects 13, edgeR and DESeq detect 14 and 15, respectively. Since we expect that many of those 33 genes would be DE, RCRdiff shows better performance in this regard. Especially, two genes, ECHDC2 (Enoyl-CoA Hydratase Domain Containing 2) and PTGDS (Prostaglandin D2 Synthase), are only detected by RCRdiff. To validate whether these two genes are truly differentially expressed between lung ADC and SCC, we used two large datasets that contain gene expression from both lung ADC and SCC patients.
Figure 1.4: Lung cancer data example: heat map of identified DE genes when comparing FFPE samples of 15 adenocarcinoma and 13 squamous patients. 48 out of 83 genes identified by at least one method are shown above.

The Rousseaux_2013 dataset [58] (GEO Accession number: GSE30219) contains 85 lung ADC patients and 100 lung SCC patients, and the Sato_2013 dataset [55, 61] (GEO Accession number: GSE41271) contains 183 lung ADC patients and 80 lung SCC patients. We compared the expression of these two genes between lung ADC and SCC patients in these two datasets. The results are summarized in Figure A.7 of Appendix using boxplots. We can see that ECHDC2 gene expression in lung ADC is significantly higher than that in SCC in the Rousseaux_2013 dataset (Figure A.7(a), p-value = 2.8e−11) and in the Sato_2013 dataset (Figure A.7 (b), p-value= 2.2e−05), and PTGDS gene expression in lung ADC is also significantly higher than that in SCC in the Rousseaux_2013 dataset (Figure A.7(c), p-value= 5.6e−13) and the Sato_2013 dataset (Figure A.7 (d), p-value= 0.00041).

A recent study has shown that the ECHDC2 gene is involved in the rewiring of mitotic networks in lung ADCs and its expression is highly correlated with VRK1 (Vaccinia-related kinase), a drug target in lung ADC mitotic networks [28]. The role of the PTGDS gene in lung cancer is still largely unknown, so that we performed meta-analyses using Lung Cancer Explorer (LCE)[7] to test its expression in tumor vs. normal samples across six different datasets (Figure A.8). The meta-analysis results show that the PTGDS gene consistently expresses lower in tumor compared to normal lung tissues in lung ADC (p-value= 3e−21) and SCC (p-value=4.4e−08). Thus, PTGDS is a potential tumor suppres-
sor gene and worth further biological follow-up studies.

1.4.2. Colorectal cancer (CRC) data

Our second example involves a dataset from a study published in Low et al.'s paper [39], which was collected to identify FFPE-based biomarkers for a prognostic gene signature to predict metastases in early-stage CRC. It contains a training set and a testing set of NanoString nCounter data from FFPE samples of early-stage CRC patients, in which the internal controls include 6 positive controls, 8 negative controls, and 14 housekeeping genes. For the training set, 193 genes were selected and measured from 97 metastasis negative and 47 metastasis positive tumor samples. The authors applied the random forest method to the training set and identified a 19-gene signature to estimate the propensity for metastasis of early-stage, low-risk, micro-satellite stable, CRC patients. The testing set only contains expression profiles of the 19-gene signature, from 94 metastasis negative and 37 metastasis positive tumor samples. The authors used the testing set to validate the performance of the 19-gene predictive signature. Again, for the purpose of quality control, we removed samples whose mean counts are lower than the maximum count of negative controls. In the training set, 71 metastasis negative and 40 metastasis positive samples are kept, while 72 negative and 28 positive samples are kept in the testing set. Given the good performance of the 19-gene signature [39], we treat these 19 genes as DE genes. Also, because the 193 genes in the training set were carefully selected by researchers, they are very informative and so a large portion of these genes may be DE. In addition, we expect that analysis results from the two data sets should agree with each other largely. As the DE gene proportion is probably high in either set, we use RCRdiff (with the default option) instead of RCRdiff-S in this example.
Only five methods were applied to detect DE genes in this example. NanoStringDiff failed to work for both training and testing data. This is because even after the removal of some samples, the mean counts of some genes are still lower than the maximum count of negative controls, and NanoStringDiff is not robust in such scenarios. For the training set, we plot the Venn diagram for DE genes detected by each of the five methods in Figure 1.5. Both DESeq and edgeR identify 31 out of the 193 genes as DE, but only 9 from DESeq and 8 from edgeR are in the 19-gene signature. NanoStringNorm+t-test claims 55 DE genes, among which 9 belong to the signature. NAPPA+t-test labels 189 genes as DE, among which 18 are in the signature. RCRdiff claims 172 DE genes, which contain the whole 19-gene signature. The last two methods label most of the genes as DE, which appears to be reasonable, because most of the 193 genes are believed to be DE. For the testing set, RCRdiff detects all 19 signature genes as DE, while NAPPA+t-test detects 18, NanoStringNorm detects 6, and both DESeq and edgeR detect only 5. In summary, RCRdiff and NAPPA+t-test appear to work well in this example, where RCRdiff is slightly better than NAPPA+t-test.
1.5. Discussion

NanoString nCounter is a relatively new technology that offers several key advantages including high sensitivity and reproducibility, technical robustness, and utility for clinical application [71]. Among medium-throughput platforms, it is the most popular [20] and has been frequently used for gene expression profiling of human biospecimens, especially for FFPE tissues [44, 60, 62]. One fundamental task in analysis of NanoString nCounter data is differential expression analysis (DEA), which is to identify which genes are differentially expressed across different types of tissue samples (e.g. normal and cancer). The sample size of nCounter data can be small and the number of variables (genes) can be large. Moreover, the nCounter data from FFPE tissues tend to have high variability, low yield and in many cases, high degradation. Thus, the gene selection task can be challenging.

We develop a novel Bayesian method, named RCRdiff, to carry out DEA using raw nCounter data. Built upon a coherent system of RCR models modified from that of RCRnorm [26] for normalization, RCRdiff is carefully designed to capture key data features for different types of probes and genes, and more importantly, it seamlessly fuses Bayesian LASSO into the RCR system for gene selection, which assumes priors favoring sparseness in the DE parameters. An MCMC-based computation technique is employed to simulate parameters from the posterior distribution. Finally, using the posterior mean of the DE parameters, two strategies based on clustering (i.e., the default option and Silhouette scanning), are proposed for DE gene selection; and with a carefully chosen rule imposed on superclusters, they do not require users to select a cutoff arbitrarily or use any external validation dataset. This makes our Bayesian procedure fully automated with essentially no tuning parameters, which is convenient for an end user with little or no statistical training. On the other hand, if one would like to control false positive rates, he/she can employ bootstrapping to resample data from the same experimental condition but artificially force them into two groups so that no DE genes actually exist. Then he/she
can select $q$ and $z$ in the rule “the $q$th percentile of the SBEs in a supercluster is smaller than $z$” to achieve the FPR at a desirable level (e.g., 0.01, 0.05, or 0.1).

Numerical evidence from four simulation studies and two real data analyses shows that RCRdiff has superior performance, as compared to existing methods. It is quite robust to heterogeneities from various sources and violations of normality assumptions. It remains competitive even when data are simulated from a completely different model. We also find that Silhouette scanning works better than the default option in selecting DE genes when the DE gene proportion $p_{DE}$ is low. Thus, a heuristic procedure can be applied to make the choice: simply run the default RCRdiff and get an estimate $\hat{p}_{DE}$, and if it is low, then re-run RCRdiff-S. Alternatively, for medium-throughput gene expression experiments, existing biological knowledge is often used to select the target subset of genes and such knowledge can also help us judge whether the DE proportion is low or high.

The strong performance of RCRdiff is perhaps due to three aspects. First, RCRdiff retains the attractiveness of RCRnorm for elaborate (hierarchical) modeling of the complex nCounter system. Second, RCRdiff inherits the merit of Bayesian LASSO in producing sparse models consistently. Last but not the least, unlike existing methods, RCRdiff adopts a fully integrated Bayesian approach for normalization and DEA, and thus avoids information loss caused by a sequential approach.

As to the computation time, RCRdiff took around 60 seconds for 6000 MCMC iterations in the reference setting of our simulation, implemented using R+Rcpp on a 2.8 GHz Intel Core i7 processor. It is less computationally demanding than the other model-based (empirical Bayes) approach NanoStringDiff, which took around 200 seconds. The rest (non-Bayesian) methods finished in seconds. We refer readers to Figure A.1 and Section A.3 of Appendix for a more detailed discussion about the computational properties of RCRdiff. An R package implementing RCRdiff is freely available from GitHub (https://github.com/canx2021/RCRdiff).
In this paper, we focus on binary classification and consider independent samples without replicates (e.g., replicates from the same patient). Nevertheless, under the Bayesian framework, future research can easily extend RCRdiff to multi-category models or survival models or models that can handle dependence or replicates.
2.1. Introduction

With the rapid development of new data collection and acquisition techniques, high-dimensional data have emerged from various fields such as biology, engineering, health sciences, and business in numerous applications including disease diagnostics, text mining, image analysis, auto-driving, marketing etc. Such high dimensionality brings up challenges that traditional statistical methods might fail to address. The challenges are further elevated in the ultra-high dimensional case, where the dimensionality dominates the sample size ($p > n$). Naturally, variable selection becomes one of the commonly adopted solutions. While earlier methods were focused on the $p < n$ case, more recent developments have shifted to the ultra-high dimensional setting. In this paper, we focus on variable selection in linear regression models for problems involving ultra-high dimensional data.

In the frequentist domain, regularization is often utilized to enforce the sparsity of the model space. For example, LASSO [68] puts an $L_1$ penalty term on the model parameters; SCAD [16] uses a smoothly clipped absolute deviation penalty; adaptive LASSO [77] uses a parameter dependent $L_1$ penalty; MCP [75] employs a minimax concave penalty. Computationally, all of the above mentioned methods boil down to some optimization problems. Therefore, different algorithms have been proposed for finding the solution, for example,
in LARS [14] and SCAD [16]. However, none of the above methods were designed for the $p > n$ case, and therefore may not provide reliable results for such problems. In order to cope with this challenge, Fan and Lv [17] proposed a new method named as the Sure Independence Screening (SIS), which first screens covariates based on their correlations with the response variable in an ultra-high dimensional problem. More recently, Zhao et al. [76] proposed a general solution for sparse learning based on a path-wise coordinate optimization algorithm, named as PathwIse CAlibrated Sparse Shooting algOrithm (PI-CASSO). A key feature of PICASSO is the “active set” concept [19, 40], where at each iteration, the active set is updated to include indices of variables that are believed to have non-zero regression coefficients.

In the Bayesian domain, people make use of carefully chosen priors to achieve sparsity. One group of methods utilizes shrinkage priors, where the negative log of the priors can be viewed as the penalty terms in their frequentist counterparts. For example, Bayesian Lasso [51] uses a Laplace prior; Carvalho et al. [8] uses a horseshoe prior; Bhattacharya et al. [6] uses a Dirichlet Laplace priors. Another group of methods utilizes the spike-and-slab prior, which is a mixture of a point mass at zero (spike) and a flat distribution (slab), e.g. [22], [63], [27], and [64]. Similarly as in the frequentist domain, all of the above methods were not proposed for ultra-high dimensional problems except the last two. The last two papers studied nonlocal priors. Shin et al. [64] proposed a new stochastic model search algorithm called Simplified Shotgun Stochastic Search with Screening (S5). Combined with the nonlocal priors proposed by Johnson and Rossell [27] in a Bayesian framework, the resulting method called BayesS5 was shown to achieve better performance in ultra-high dimensional problems.

The above mentioned methods usually assume that the error term in the linear regression model follows a normal distribution. However, such an assumption is not always satisfied in practice, leading to potentially biased estimation and invalid inference. One could transform the response $y$ such that the transformed $y$ approximately follows a normal
distribution. Yet, in many areas, people are interested in understanding the relationship between the covariates and the outcome variable in its original form.

Empirical likelihood (EL) provides a solution that does not require distributional assumptions. It was first proposed for univariate mean estimation [48], and then extended to multivariate mean estimation and linear regression [46, 47]. Empirical likelihood enjoys several appealing properties, including estimation efficiency [52], Wilk’s theorem [48], and transformation preserving [49]. Also, it is Bartlett correctable [13]. It has been successfully applied to censoring data [38, 43, 50], truncation data [37], time series data [12, 29, 45], biased sampling problems [53, 54], and so on. Recently, EL based variable selection methods have been proposed to exploit the semi-parametric nature of EL. Tang and Leng [65] proposed penalized empirical likelihood for mean estimation and variable selection in linear regression by adding an appropriate penalty function to the standard empirical likelihood. They then extended it to general estimating equations [35]. They showed the oracle property under the smoothly clipped absolute deviation (SCAD) penalty. More recently, the Bayesian framework has been applied to the penalized empirical likelihood, based on the connection between prior distributions and penalty functions. Examples include ridge regression [4], LASSO [4, 11], and elastic net [42]. However, all of above empirical likelihood based methods can only deal with the case when the dimension $p$ diverges at a slower rate than the sample size $n$ (i.e., $p$ cannot grow faster than $n^d$ for some $d$ satisfying $0 < d < 1$). This is due to the fact that the true value of a parameter might escape from the convex hull formed by data with a positive probability if $p$ diverges too fast [10, 69]. Several attempts have been made for the $p > n$ case. Lahiri and Mukhopadhyay [31] proposed a modified EL method for the mean estimation problem. Chang et al. [9] introduced double penalties to encourage the sparsity of both the estimators and the Lagrange multipliers associated with the estimating equations, and thus, enabling variable selection in the $p > n$ case.
We propose a Bayesian variable selection solution based on empirical likelihood for linear regression models involving ultra-high dimensional data \((p > n)\). Motivated by Chang et al. [9], we introduce priors to doubly penalize the empirical likelihood, which regularize both regression parameters and Lagrange multipliers. To reduce the computational burden brought by the ultra-high dimensionality, we adopt the idea of active sets for Bayesian MCMC sampling and only update the so-called “active” parameters at each iteration. In this way, our proposed method not only retains the attractiveness of empirical likelihood mentioned earlier, but also provides reliable results in variable selection and parameter estimation for ultra-high dimensional problems. In addition, as a Bayesian method, our method inherits all merits that the Bayesian paradigm can offer.

The remainder of this paper is organized as follows. In Section 1.2, we first provide a brief introduction to empirical likelihood, then describe the proposed Bayesian model for doubly penalized empirical likelihood. We further develop an efficient algorithm for MCMC sampling. In Section 1.3, we conduct simulation to compare the performance of the proposed method with existing approaches in terms of variable selection and parameter estimation. Two examples are shown in Section 1.4 for further illustration using real data. Section 1.5 gives a brief discussion about limitations and possible future directions.

2.2. Method

2.2.1. Preliminaries

2.2.1.1. Empirical likelihood and penalized empirical likelihood

Let \(y\) be the outcome vector of length \(n\), centered to have mean 0; \(X\) be the \(n \times p\) covariate matrix, with each column representing a covariate and standardized to have mean 0 and standard deviation 1; \(\epsilon\) is the \(n \times 1\) vector of independent and identically
distributed (iid) errors with mean 0 and unknown variance $\sigma^2$. $X$ and $y$ are assumed to follow a linear model with the distribution of $\epsilon$ unspecified:

$$y = X\beta + \epsilon; \quad E(\epsilon) = 0; \ Var(\epsilon) = \sigma^2 I. \quad (2.1)$$

Empirical likelihood [47, 48] estimates the vector of regression coefficients $\beta$ by finding the maximizer of the following empirical likelihood function:

$$L(\beta) = \sup \{ \prod_{i=1}^{n} w_i : w_i \geq 0, \sum_{i=1}^{n} w_i = 1, \sum_{i=1}^{n} w_i U_i(\beta) = 0 \}, \quad (2.2)$$

where $w_i$ is the weight for the $i$th observation and $U_i(\beta) = 0$ corresponds some unbiased estimating equations for $\beta$ based on the $i$th observation. In the linear regression setting, $U_i(\beta) = x_i(y_i - x_i^T \beta)$. Here $y_i$ is the $i$th element of $y$ and $x_i$ is a vector being the transpose of the $i$th row of $X$. Using the Lagrange multiplier method, the constrained optimization problem is maximized when $w_i = n^{-1} \left[ 1 + \lambda^T U_i(\beta) \right]^{-1}$, where $\lambda$ satisfies the equation:

$$\sum_{i=1}^{n} \frac{U_i(\beta)}{1 + \lambda^T U_i(\beta)} = 0. \quad (2.3)$$

Note that finding the $\beta$ that maximizes (2.2) is equivalent to solving a dual optimization problem [9, 49], namely

$$\hat{\beta} = \arg \max_{\beta} \min_{\lambda} l(\beta),$$

where $l(\beta) = \sum_{i=1}^{n} -\log \left[ 1 + \lambda^T U_i(\beta) \right]$.

Variable selection based on penalized empirical likelihood [35, 65] is to find $\hat{\beta}_P$ that minimizes the combination of the negative log empirical likelihood and the extra penalty term

$$l_P(\beta) = \sum_{i=1}^{n} \log \{ 1 + \lambda^T U_i(\beta) \} + n \sum_{j=1}^{p} P(|\beta_j|),$$

where $P$ is a penalty function of $\beta$ that encourages sparsity and $\lambda$ satisfies the equation (2.3).
2.2.1.2. Bayesian empirical likelihood with a LASSO-type prior

As noted in [68] and [51], the LASSO estimates can be interpreted as the posterior mode when the regression parameters \( \beta_k \)'s have independent and identical Laplace priors, which can be viewed as a mixture of normal distributions [2]. Analogous with the original Bayesian LASSO, Cheng and Zhao [11] proposed a Bayesian approach for variable selection using empirical likelihood with the following hierarchical structure for the prior distributions:

\[
\pi(\beta | \tau_1^2, \ldots, \tau_p^2) \sim N(0, D_\tau) \quad \text{and} \quad \pi(\tau_1^2, \ldots, \tau_p^2 | \gamma^2) \sim \prod_{k=1}^{p} \frac{\gamma^2}{2} \exp(-\gamma^2 \tau_k^2 / 2),
\]

where \( D_\tau = \text{diag}(\tau_1^2, \ldots, \tau_p^2) \) and \( \tau_1^2, \ldots, \tau_p^2 \) are the latent variance parameters for \( \beta \) and \( \gamma \) is a hyper-parameter. Consequentially, the posterior (empirical) likelihood is given by

\[
p(\beta, D_\tau | X, y) \propto \prod_{i=1}^{n} \frac{1}{1 + \lambda^2 \mathcal{U}_i(\beta)} \prod_{k=1}^{p} \left[ (2\pi \tau_k^2)^{-1/2} \exp\left(-\frac{\beta_k^2}{2\tau_k^2}\right) \cdot \frac{\gamma^2}{2} \exp(-\gamma^2 \tau_k^2 / 2) \right],
\]

where \( \lambda \) solves Equation (2.3).

2.2.1.3. The doubly penalized empirical likelihood

As mentioned in the introduction, conventional EL based methods can be problematic when the dimension \( p \) diverges at a faster rate than the sample size \( n \). Thus, it is challenging to employ EL in the ultra-high dimensional (\( p \gg n \)) situation [9, 35, 65]. Under the estimating equation setting, Chang et al. [9] suggested to add a second penalty term on the Lagrange multipliers in addition to an existing penalty term on regression parameters, thus enforcing the sparsity of \( \lambda \) and thus the selection of estimating equations besides the selection of covariates. The sparse parameter estimates based on estimating equations
can be obtained by

\[ \hat{\beta} = \arg \min_{\beta} \max_{\lambda} \left[ \sum_{i=1}^{n} \log \left( 1 + \lambda^T g(x_i, \beta) \right) + n \sum_{k=1}^{p} P_{1,\pi}(|\beta_k|) - n \sum_{j=1}^{r} P_{2,\nu}(|\lambda_j|) \right], \quad (2.4) \]

where \( \beta = (\beta_1, \ldots, \beta_p)^T \), \( \lambda = (\lambda_1, \ldots, \lambda_r)^T \), \( g_j(x, \beta) \) is the \( j \)th estimating equation, \( r \) is the total number of estimating equations, and \( P_{1,\pi}(\cdot) \) and \( P_{2,\nu}(\cdot) \) are two penalty functions with tuning parameters \( \pi \) and \( \nu \). They showed that under the new double-penalized framework, the estimation is sparse and consistent when the number of parameters is allowed to grow exponentially with the sample size. Note that, for linear regression, the number of estimating equations is the same as the number of regression parameters (i.e. \( r = p \)).

2.2.2. The proposed Bayesian EL method

Motivated by the works described above, we propose a Bayesian empirical likelihood (BEL) based method for variable selection in the ultra-high dimensional setting, labeled BEL_HD.

Given the empirical likelihood \( L(\beta) \propto \prod_{i=1}^{n} \left[ 1 + \lambda^T U_i(\beta) \right]^{-1} \), we first apply Laplace priors on \( \beta \) to encourage sparsity

\[ \pi(\beta|\tau_1^2, \ldots, \tau_p^2) \sim \mathcal{N}(0, D_r) \quad \text{and} \quad \pi(\tau_1^2, \ldots, \tau_p^2|\gamma_1^2) \sim \prod_{k=1}^{p} \frac{\gamma_1^2}{2} \exp(-\gamma_1^2 \tau_k^2/2), \]

where \( D_r = \text{diag}(\tau_1^2, \ldots, \tau_p^2) \) are the latent variance parameters for \( \beta \) and \( \gamma_1 \) is a global hyper-parameter introduced by the exponential priors for \( \tau_k^2 \)'s. Therefore, the full conditional posterior for \( \beta \) can be written as follows:

\[ p(\beta|\cdot) \propto \prod_{i=1}^{n} \frac{1}{1 + \lambda^T U_i(\beta)} \prod_{j=1}^{p} (2\pi \tau_j^2)^{-1/2} \exp \left( -\frac{\beta_j^2}{2\tau_j^2} \right), \]
where \( \lambda \) is the maximizer of the following objective function

\[
\hat{\lambda} = \arg \max_{\lambda} \left\{ \prod_{i=1}^{n} \{1 + \lambda^T U_i(\beta)\} \prod_{j=1}^{p} (2\pi \nu_j^2)^{-1/2} \exp \left( -\frac{\lambda^2}{2\nu_j^2} \right) \right\}.
\]

Note that a penalty term is put on \( \lambda \) to achieve more shrinkage on the final estimates, where \( \nu_1^2, \ldots, \nu_p^2 \) are the shrinkage parameters for \( \lambda_1, \ldots, \lambda_p \).

We further specify \( \pi(\nu_1^2, \ldots, \nu_p^2 | \gamma_2^2) \sim \prod_{k=1}^{p} \frac{\gamma_2^2}{\nu_k^2} \exp(-\gamma_2^2 \nu_k^2 / 2) \), where \( \gamma_2 \) is a global hyper-parameter introduced by the exponential priors for \( \nu_k^2 \)'s. It is easy to see the full conditional posteriors of \( \tau_k^{-2} \)'s follow inverse-Gaussian distributions,

\[
\tau_k^{-2} | \cdot \sim \text{Inverse Gaussian} \left( \sqrt{\frac{\gamma_2^2}{\beta_k^2}}, \gamma_2^2 \right).
\]

We further apply a conjugate prior \( \text{Gamma}(\zeta = 0.1, \delta = 0.1) \) on the hyper-parameter \( \gamma_1^2 \) so that the full conditional posterior of \( \gamma_1^2 \) follows a Gamma distribution

\[
\gamma_1^2 | \cdot \sim \text{Gamma} \left( p + \zeta, \sum_{k=1}^{p} \frac{\tau_k^2}{2} + \delta \right).
\]

Similarly, the full conditional posteriors of \( \nu_k^{-2} \) and \( \gamma_2^2 \) are given by

\[
\nu_k^{-2} | \cdot \sim \text{Inverse Gaussian} \left( \sqrt{\frac{\gamma_2^2}{\lambda_k^2}}, \gamma_2^2 \right),
\]

\[
\gamma_2^2 | \cdot \sim \text{Gamma} \left( p + \zeta, \sum_{k=1}^{p} \frac{\nu_k^2}{2} + \delta \right).
\]

Posterior sampling can be achieved by iteratively sampling from the full conditionals listed above via a Gibbs sampler.
2.2.3. Algorithm

Bayesian computation for ultra-high dimensional problems can be slow if we were to update coefficients of all variables in each iteration. To accelerate the computational speed, we adopt the active set idea introduced for path-wise coordinate optimization [19, 40, 76]. The path-wise coordinate optimization algorithm proposed in [76] consists of three loops: the outer loop to update regularization parameters, the middle loop to update the active set, and the inner loop to update the estimates of regression coefficients for variables in the active set. Here, an active set includes all candidates of variables with non-zero coefficient.

We simplify this algorithm and adjust it for our Bayesian framework. A diagram for the flow of our algorithm is shown in Figure 2.1. The proposed algorithm consists of two loops: the outer loop updates (global) hyper-parameters $\gamma_1$ and $\gamma_2$, and the active set $A$; the inner loop updates covariate-level parameters including regression the coefficients $\beta_j$, the Lagrange multipliers $\lambda_j$, and $\tau_j^2$ and $\nu_j^2$ for $j \in A$. The details of two loops are shown in Algorithm 1 and Algorithm 2.

![Figure 2.1: The flow of our proposed algorithm. It consists of two loops. The inner loop updates $\beta$ and other covariate-level latent variables in the active set. The outer loop updates global hyper-parameters and the active set based on the inner loop results.](image-url)
Algorithm 1: The outer loop updates global hyperparameters $\gamma_1$ and $\gamma_2$ and the active set $A$ based on results from the inner loop.

**Input:** $X$, $y$

**Output:** $\beta_{final}$

**Initialize:** $\beta^{(1)}$, $\lambda^{(1)}$, $\gamma_1^{(1)}$, $\gamma_2^{(1)}$, $\tau^{(1)}$, $\nu^{(1)}$

Set $A^{(1)} = \{ j \mid |\beta_j^{(1)}| > s \}$.

for $k \leftarrow 2$ to $K$ do

1. Generate $\gamma_1^{(k)}$ from $\text{Gamma}(p_{A^{(k-1)}} + \zeta, \sum_{j \in A^{(k-1)}} (\tau_j^{(k-1)})^2 + \delta)$, where $p_{A^{(k-1)}}$ is the number of elements in the active set $A^{(k-1)}$, $\zeta$ and $\delta$ are user-defined small positive numbers with the default value set to 0.1.
2. Generate $\gamma_2^{(k)}$ from $\text{Gamma}(p_{A^{(k-1)}} + \zeta, \sum_{j \in A^{(k-1)}} (\nu_j^{(k-1)})^2 + \delta)$.
3. Generate $\beta_j^{(k)}$, $\lambda_j^{(k)}$, $\tau_j^{(k)}$, and $\nu_j^{(k)}$ using Algorithm 2, for $j \in A^{(k-1)}$.
4. Set $A^{(k)} = \{ j \mid |\beta_j^{(k)}| > s \} \cup T^{(k)}$, where $T^{(k)} = \{ j \mid |\beta_j^{(k)}| < s \text{ and } |\nabla_j l(\beta^{(k)})| \geq Q_{r,G^{(k)}} \}$ or $\emptyset$ for a very large $p$. Here, $Q_{r,G^{(k)}}$ is the $\left(100 - r\right)$th percentile of the absolute values of gradients for variables in $G^{(k)} = \{ j \mid |\nabla_j l(\beta^{(k)})| \geq t \text{ and } |\beta_j^{(k)}| < s \}$; $s$ and $t$ are pre-specified thresholds.

end

**return** $\beta_{j_{final}} \leftarrow \text{average of samples } \{ \beta_j^{(k)} : B \leq k \leq K \} \text{ for } \{ j : j \in A^{(K)} \}$ and $\beta_{i_{final}} \leftarrow 0 \text{ for } \{ i : i \notin A^{(K)} \}$.

* $B$ is the burn-in iteration number for the outer loop.
Algorithm 2: The inner loop implements the doubly penalized algorithm.

Input: $X_{A^{(k-1)}}$, $y$, $\beta_{A^{(k-1)}}$, $\lambda_{A^{(k-1)}}$, $\gamma_1$, $\gamma_2$

Output: $\beta_{A^{(k-1)}}^{(k)}$, $\lambda_{A^{(k-1)}}^{(k)}$, $\tau_{A^{(k-1)}}^{(k)}$, $\nu_{A^{(k-1)}}^{(k)}$

Set $\beta_{A^{(k-1)}}^{(k,1)} = \beta_{A^{(k-1)}}^{(k)}$ and $\lambda_{A^{(k-1)}}^{(k,1)} = \lambda_{A^{(k-1)}}^{(k)}$.

for $m \leftarrow 2$ to $M$ do

1. Generate $(\tau_{A^{(k-1)}}^{(k,m)})^{-2}$ from InvGauss\left(\frac{\gamma_1}{\beta_{A^{(k-1)}}}, \frac{\gamma_2}{\beta_{A^{(k-1)}}}\right).

2. Generate $(\nu_{A^{(k-1)}}^{(k,m)})^{-2}$ from InvGauss\left(\frac{\gamma_1}{\beta_{A^{(k-1)}}}, \frac{\gamma_2}{\beta_{A^{(k-1)}}}\right).

3. Generate $\beta^*$ from a multivariate normal distribution with mean: $\mu = yX_{A^{(k-1)}}^T \Sigma_n^{-1} X_{A^{(k-1)}} X_{A^{(k-1)}} / n$, covariance: $\Sigma = (X_{A^{(k-1)}} X_{A^{(k-1)}}^T + \Sigma A^{(k-1)}) / n + D_{\tau_{A^{(k-1)}}}^{-1}$, where $X_{A^{(k-1)}}$ consists of covariates in the active set $A^{(k-1)}$, $\Sigma_n = \sum_{i=1}^n (U_{OLS A^{(k-1)},i} x_{A^{(k-1)},i} y_i - x_{A^{(k-1)},i})$, $\beta_{OLS} = (X_{A^{(k-1)}}^T X_{A^{(k-1)}})^{-1} X_{A^{(k-1)}} y$, $x_{A^{(k-1)},i}$ is the $i$th row of $X_{A^{(k-1)}}$, and $D_{\tau_{A^{(k-1)}}}$ is a diagonal matrix with components of $(\tau_{A^{(k-1)}})^{-2}$.

4. Obtain $\lambda^*$ by maximizing $\sum_{i=1}^n \log(1 + \lambda_{A^{(k-1)}}^T U_{A^{(k-1),i}} (\beta^*)) - \sum_{j=1}^p \frac{\lambda_j^2}{2 (u_{A^{(k,m)},i}^*)^2}$ with respect to $\lambda_{A^{(k-1)}}$, where $U_{A^{(k-1),i}} = x_{A^{(k-1),i}} (y_i - x_{A^{(k-1),i}}^T \beta^*)$.

5. The acceptance rate is: $r = \min\left(1, \frac{p(\beta^* | X_{A^{(k-1)}}, y, \Sigma_{\tau_{A^{(k-1)}}}) \frac{g(\beta_{A^{(k-1)}}^{(k,1)} | \mu, \Sigma)}}{p(\beta_{A^{(k-1)}}^{(k,1)} | X_{A^{(k-1)}}, y, \Sigma_{\tau_{A^{(k-1)}}}) \frac{g(\beta^* | \mu, \Sigma)}}\right)$, where $p(\cdot)$ is the posterior likelihood and $g(\cdot)$ is the density function for the proposal multivariate Gaussian.

6. Generate $u \sim \text{Uniform}(0, 1)$ and set $\beta_{A^{(k-1)}}^{(k,m)} = \beta^*$ and $\lambda_{A^{(k-1)}}^{(k,m)} = \lambda^*$ if $u < r$.

Otherwise, set $\beta_{A^{(k-1)}}^{(k,m)} = \beta_{A^{(k-1)}}^{(k-1), m-1}$ and $\lambda_{A^{(k-1)}}^{(k,m)} = \lambda_{A^{(k-1)}}^{(k-1), m-1}$.

end

return $\beta_{A^{(k-1)}}^{(k)}$, $\lambda_{A^{(k-1)}}^{(k)}$, $\tau_{A^{(k-1)}}^{(k)}$, $\nu_{A^{(k-1)}}^{(k)}$, which are the averages of samples $\{m : B_{inner} \leq m \leq M\}$.

* $B_{inner}$ is the burn-in iteration number for the inner loop.
The active set is updated in the outer loop. At each iteration $k$, it is a union of two disjoint subsets of the set of all $p$ covariates. The first subset includes all variables with $|\beta_j^{(k)}| > s$, given regression coefficient estimates can never be exactly zero in a Bayesian setting. In practice, $s$ can be set to be a small positive value and then adjusted by the trial and error approach or by cross-validation. The second subset, denoted by $T$, consists of variables with small $|\beta_j^{(k)}|$ but large $|\nabla_j l(\beta^{(k)})|$ (i.e., small estimates but large gradients).

Let $G$ be the set of variables with $|\nabla_j l(\beta^{(k)})| \geq t$ and $|\beta_j^{(k)}| \leq s$. We rank the variables in $G$ by their gradients and select the top $r\%$ of the variables in $G$ to enter $T$. We suggest to set the threshold $t$ to the maximum value of $|\nabla_j l(\beta^{(k)})|$ in the active set of the previous iteration ($k-1$) multiplied by a constant. When $p$ is extremely large compared to $n$, $T$ can be directly set to an empty set, to accelerate the computation speed and achieve a sparse solution.

In the inner loop, we update $1/\tau^2$ and $1/\nu^2$ by sampling from inverse-Gaussian distributions. Following [11], we update $\beta_A$ using the Metropolis-Hasting (MH) algorithm with a multivariate normal distribution as the proposal distribution. The corresponding $\lambda_A$ is updated via optimization.

2.3. Simulation

We conducted simulation to illustrate the performance of the proposed method in parameter estimation and variable selection, and to compare it with six existing approaches, including LASSO [68], SCAD [16], SIS [17], PICASSO [76], BayesS5 [64], and the doubly penalized EL method (EL_DP) [9]. The first five methods were implemented through R packages “lars”, “ncvreg”, “SIS”, “PICASSO”, and “BayesS5”, whereas EL_DP was implemented using a demo code on the authors’ website. As we discussed in Section 2.1, there are many variable selection methods, which were not originally proposed for ultra-high dimensional problems. Among them, we chose LASSO and SCAD here for their popularity.
and/or overall better performance. The rest four methods were proposed for ultra-high dimensional problems and showed better performance than other alternatives in their numerical studies. For our proposed BEL_HD, we ran 300 iterations for the outer loop and 100 iterations for the inner loop, where the first 150 and 20 iterations were used as burn-in samples, respectively. We used the estimates obtained from PICASSO (with the default $\mathcal{L}_1$ regularizer) as our initial values. To start from a relatively conservative point, we tuned the smallest regularization parameter ratio (lambda.min.ratio) to be 0.01 in PICASSO, which would lead to more false positives but also include as many true positives as possible (i.e., high recall but low precision). We set $s = 0.01$ and $t = 3 \max(|\nabla_{j \in A^{(k-1)}} I(\beta^{(k)})|)$ as our thresholds, and $r = 10$ to select the top 10% of variables to enter the active set in each outer loop, whose gradients have absolute values larger than $t$. Note that the gradient threshold $t$ varies from iteration to iteration in the outer loop. For “SIS”, we set “greedy=TRUE”, which ran the greedy modification of the permutation-based ISIS and achieved overall better performance. For other methods, we used the default setting.

We designed seven simulation scenarios to explore the impacts of potentially important factors, including the number of the nonzero elements reflecting the sparsity of the selection problem, the mean and standard deviation (SD) of nonzero regression parameters reflecting the average (nonzero) effect size and their variability, respectively, the ratio $p/n$, covariate dependence, and types of error distributions. For each scenario, 100 replicates were independently simulated. The average mean squared error (MSE) is used to assess the performance of all methods in parameter estimation. Precision, recall, and F score (defined as $F_1 = 2 \times (\text{precision} \times \text{recall})/(\text{precision} + \text{recall})$) are used to assess the accuracies in variable selection.

2.3.1. Settings

We simulated data from $y = X\beta + \epsilon$ with $X$ generated from $\mathcal{N}(0, \Sigma)$, and $\epsilon$ generated from one of the following distributions independently and identically: normal, t, lognormal
or a mixture of these distributions. The first \( q \) elements of \( \beta \) were independently sampled from \( \mathcal{N}(\mu_\beta, \sigma^2_\beta) \), \( \mu_\beta \neq 0 \), and the remaining components of \( \beta \) were all set to zero. We centered our simulation around one basic setting, and for each scenario described below, we varied one design parameter only while fixing all the others. For the basic setting, \( \epsilon \) was generated from a mixture of two altered \( t_3 \) distributions (shifted and rescaled to have mean \( = 3 \) or \( -3 \) with SD=1), each with probability 0.5; the first \( q = 20 \) elements of \( \beta \) were randomly sampled from \( \mathcal{N}(5, 3^2) \); we set \( n = 100, p = 800, \Sigma_{kk} = 1 \) and \( \Sigma_{kl} = 0.5 \) for any \( l \neq k \). The following seven scenarios were considered:

- **Scenario I**: different numbers of the non-zero elements. The first \( q \) elements of \( \beta \) were randomly sampled from \( \mathcal{N}(5, 3^2) \), where \( q \in \{10, 15, 20, 25, 30\} \).

- **Scenario II**: different mean sizes of the non-zero elements of \( \beta \). The first 20 elements of \( \beta \) were randomly sampled from \( \mathcal{N}(\mu_\beta, 3^2) \), where \( \mu_\beta \in \{1, 3, 5, 7, 9\} \).

- **Scenario III**: different levels of variation for the non-zero elements of \( \beta \). The first 20 elements of \( \beta \) were randomly sampled from \( \mathcal{N}(5, \sigma^2_\beta) \), where \( \sigma_\beta \in \{1, 2, 3, 4, 5\} \).

- **Scenario IV**: different signal-to-noise ratios. We fixed the distribution of \( \beta \) at \( \mathcal{N}(5, 3^2) \), but varied the scale of random errors by generating \( \epsilon \) from a mixture of \( t_3 \) distributions rescaled to have standard deviation \( \sigma \in \{1, 2, 3, 4, 5\} \). Thus, the larger \( \sigma \) is, the lower the signal-to-noise ratio is.

- **Scenario V**: different \( p/n \) ratios. We fixed \( n \) to be 100, with \( p \) increased exponentially: 200, 400, 800, 1600, or 3200.

- **Scenario VI**: different levels of correlations of the covariates. We generated \( X \) from \( \mathcal{N}(0, \Sigma) \), where \( \Sigma_{kk} = 1 \) and \( \Sigma_{kl} = 0, 0.1, 0.3, 0.5, \) or 0.7 for any \( l \neq k \).

- **Scenario VII**: different types of error distributions. Besides the mixture of \( t \) distributions, we consider \( 3t_3 \) (a heavy tailed distribution); altered lognormal (a skewed distribution), i.e. lognormal \((0, 1)\) shifted and rescaled to have mean 0 and SD 3); \( N(0, 3^2) \); a mixture of two (altered) log normal distributions: one with mean 3 and
SD 1 and the other with mean -3 and SD 1, each with probability 0.5; and a mixture of normal distributions $\mathcal{N}(3, 1)$ and $\mathcal{N}(-3, 1)$, each with probability 0.5.

2.3.2. Results for parameter estimation

We report the average MSE (in the log10 scale) of each method in Figure 2.2 for Scenarios I – VI and in Table 2.1 for Scenario VII. The average MSE is defined as $\sum_{j=1}^{p} (\hat{\beta}_j - \beta_j)^2 / p$, averaged over all the replicates in each scenario, with $\hat{\beta}_j$ being the estimated $\beta_j$, the $j$th element of $\beta$.

Figure 2.2: The average MSE (in the log10 scale) of all methods for Scenarios I–VI.
First of all, we find from Figure 2.2 that the proposed BEL_HD produced the lowest (average) MSE for most cases. In the few other cases when the variance of $\epsilon$ is large or when the correlation among covariates is large, BEL_HD still produced competitive results that are very close to the best.

From Panel I, II and IV of Figure 2.2, we observe that increasing $q$ (the number of nonzero elements) or $\mu_\beta$ (the mean size of nonzero elements) or $\sigma$ (the error variance) increases the MSE in general for all methods. Note that selecting a zero element as an active one would not affect the MSE much because their estimates are usually quite close to zero; however, incorrectly shrinking a nonzero element (with larger $\beta_j$) to zero would yield a larger MSE. Thus, increasing $q$ or $\mu_\beta$ would increase the MSE. It is also evident why the MSE has an increasing trend over $\sigma$: the signal to noise ratio gets smaller as $\sigma$ increases, rendering parameter estimation more difficult. Panel III of Figure 2.2 further shows that increasing $\sigma_\beta$ (the variability of the nonzero elements) would decrease the MSE. In this experiment, as $\sigma_\beta$ gets large from 1 to 5, the nonzero elements have a larger chance to fall in a neighbor of zero (as the mean $\mu_\beta$ is fixed at 5), in which the MSE would not be as large as in the case when $\sigma_\beta$ is small so that most nonzero elements stay away from zero. Overall, changing $q$ or $\mu_\beta$ has a (much) larger impact on the MSE than changing the scale parameters $\sigma$ or $\sigma_\beta$.

Panel V of Figure 2.2 shows that the effect of the $p/n$ ratio on the MSE can be very different for different methods. As the $p/n$ ratio increases exponentially, BayesS5 and SCAD show a decreasing pattern; SIS and LASSO show a concave pattern; the proposed BEL_HD shows a somewhat convex pattern; PICASSO and EL_DP have similar performance and show a quite flat pattern, except for the highest value 5 of the log2 ratio, where EL_DP suddenly became the worst, caused by its computational instability. From Panel VI, we find that increasing covariate dependence typically increases the MSE, except for correlations close to zero.

Table 2.1 shows that the effect of type of error distribution on the MSE is perhaps
smallest, compared to the other six factors considered. Among all, BayesS5 and SIS are more sensitive to the change of the distribution type, and they tend to be more vulnerable in other scenarios as well. The proposed BEL_HD shows better performance regardless of the distribution type. It is also interesting to observe that EL_DP and PICASSO show similar MSE results in many settings across the different scenarios.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>LASSO</th>
<th>SCAD</th>
<th>SIS</th>
<th>PICASSO</th>
<th>BayesS5</th>
<th>EL_DP</th>
<th>BEL_HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>0.24</td>
<td>0.19</td>
<td>0.38</td>
<td>0.09</td>
<td>1.37</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>N</td>
<td>0.24</td>
<td>0.2</td>
<td>0.41</td>
<td>0.09</td>
<td>1.42</td>
<td>0.09</td>
<td>0.08</td>
</tr>
<tr>
<td>LN</td>
<td>0.24</td>
<td>0.19</td>
<td>0.44</td>
<td>0.09</td>
<td>1.41</td>
<td>0.09</td>
<td>0.08</td>
</tr>
<tr>
<td>Mix-t</td>
<td>0.25</td>
<td>0.19</td>
<td>0.42</td>
<td>1.38</td>
<td>0.09</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>Mix-N</td>
<td>0.24</td>
<td>0.19</td>
<td>0.39</td>
<td>1.37</td>
<td>0.09</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>Mix-LN</td>
<td>0.26</td>
<td>0.2</td>
<td>0.37</td>
<td>1.42</td>
<td>0.1</td>
<td>0.09</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: The average MSE of all methods for Scenario VII.

2.3.3. Results for variable selection

To evaluate the performance of variable selection, we present the recall, precision, and F score of each method for Scenarios I – VII in Figures 2.3 (I-IV) and 2.4 (V-VII). Overall, the proposed BEL_HD produced competitive recall/precision, leading to the best F score in nearly all the scenarios. The results indicate that BEL_HD achieved a better balance between recall and precision than its competitors. Like BEL_HD, BayesS5 has the highest (or second highest) precision. However, BayesS5 has the lowest recall, because it tends to select fewer variables, leading to low true positives and low false positives. By contrast, PICASSO has the highest recall but the lowest precision, because it tends to select more variables, leading to high true positives and high false positives. As mentioned at the beginning of Section 1.3, to use estimates from PICASSO as our conservative starting point, we set the tuning parameter lambda.min.ratio for PICASSO to 0.01. In fact, doing so has led to very small MSEs for PICASSO, as reported in Sec-
tion 2.3.2, because high false positives would not hurt the MSE as much as high false negatives. If we set lambda.min.ratio to the default value 0.05, PICASSO would have much worse performance in parameter estimation meanwhile a bit better performance in variable selection. Even with lambda.min.ratio=0.5, PICASSO still has quite many false positives (this is also why we selected PICASSO to generate initial points) and so it can not beat the proposed BEL_HD in variable selection.

It is interesting to observe that the trends for variable selection performance do not always match with those of the estimation performance. In particular, when the scale of non-zero elements of $\beta$ increases, the estimation accuracy increases while the variable selection performance decreases.
Figure 2.3: Average recall, precision, and F scores for Simulation Scenarios I–IV.
Figure 2.4: Recall, precision, and F scores of all methods in Simulation Scenarios V-VII.

2.3.4. Computing time

The computing time in seconds on the log2 scale of all methods for different $p/n$ ratios in Scenario 5 is shown in Figure 2.5. Our proposed method and the other Bayesian approach BayesS5 are slower than the other non-Bayesian methods except for EL_DP. While enjoying the merits of both EL and Bayesian approaches, BEL_HD inherits their disadvantage in computing speed, too. Still, with the aid of the active set concept, BEL_HD
is faster than EL_DP, which is also based on EL.

Figure 2.5: The computational time on the log2 scale of all methods under different $p/n$ ratios.

2.4. Empirical studies

We present two empirical studies to further illustrate the performance of our proposed method BEL_HD and to compare it with five existing methods including LASSO, SCAD, SIS, PICASSO, and BayesS5. The first study involves US crime data [15] that contains 15 variables besides the response variable (crime rate) for 47 states, in which we added noise variables to examine the variable selection performance in two ultrahigh-dimensional scenarios. The second involves a gene expression data [32] containing 45,037 gene expression levels for 60 mice. EL_DP was not included in the comparison, because it was time consuming in both studies. For instance, it took more than 24 hours to finish one replicate in our second study. LASSO was excluded in the second example as well, because the extremely high dimensionality caused some computational issue when using the R package “lars”. For PICASSO, we used the default setting with
lambda.min.ratio = 0.05 for better performance here.

Given the ground truth is unknown for real data, mean squared prediction error (MSPE) is used to assess the performance of all six methods in terms of predictive accuracy. Each data set was randomly split into test and training sets. Due to the limited sample size, we only chose 5 samples as the test set \( S_{test} \) and the rest as the training set. We define MSPE = \( \sum_{i \in S_{test}} (y_i - \hat{y}_i)^2 \) \(|S_{test}| \) with \( \hat{y}_i = x_i^T \hat{\beta} \) and \( \hat{\beta} \) computed using the corresponding training set, averaged over 100 replicates. As in our simulation, for BEL_HD, we still used estimates of PICASSO (lambda.min.ratio = 0.01) as the initial values. For the first problem, we ran 300 iterations for the outer loop and 100 iterations for the inner loop, where the first 150 and 20 iterations were used as burn-in samples. For the second problem involving an extremely large \( p \), we use the same iteration and burn-in numbers for the inner loop but run 1000 iterations for the outer loop and used 500 as burn-in samples. In order to speed up the convergence and also reduce the computational burden, we chose the threshold \( s \) in BEL_HD by 10 fold cross-validation using the training set from the first replicate. For the US crime data, we used the same threshold \( t \) as in our simulation; for the gene expression data, we directly set \( T = \emptyset \) as \( p \) is extremely large (over 40,000) and \( n < 100 \) and the sparsity may be extreme. We used Geweke’s diagnostic [23] to detect the convergence of an MCMC chain, which is based on a test for equality of the means of the first 30% and last 50% iterations of the chain.

### 2.4.1. US crime data example

US crime data was available in the Uniform Crime Reports of the FBI and first studied by Ehrlich [15] to investigate the effect of punishment regimes on crime rates. The errors in the data were corrected by Vandaele and others [70]. The data consists of crime rates for 47 states in US, and 15 variables. We treated the crime rate on the log scale as the response variable. To construct high-dimensional settings, we added noise (irrelevant) variables using the following two scenarios:
1. Noise variables were generated from $N(0, \Sigma)$, where $\Sigma_{kk} = 1$ and $\Sigma_{kl} = 0.1$ for any $l \neq k$. We consider three different $p/n$ ratios: 2, 4, and 8. Therefore, 79, 173, and 361 noise variables were added.

2. The $p/n$ ratio was fixed to be 4. For each original variable, we generated one correlated confounding variable from the standard normal distribution. The rest of the noise variables were still generated as in the first setting. Three different correlation values were considered for the confounding variables: 0.3, 0.5, and 0.7.

The data set was randomly split into 5 test samples and 42 training samples, and the average MSPE of 30 replicates is shown in Tables 2.2 and 2.3 for Scenarios 1 and 2, respectively, in which the smallest average MSPE in each setting is highlighted in bold. We also list how many original (the first number in the parenthesis) and noise variables (the second number in the parenthesis) selected by each method. Although the truth is unknown, a good method should select fewer noise variables at least. We can evaluate the false positive rate of each method by checking how many noise variables were selected. For most methods, increasing the $p/n$ ratio renders estimation and variable selection more challenging. They tend to select fewer original variables and more noise variables as the ratio increases, yielding high false positive rates and larger MSPEs. Similarly, adding the correlated confounding variables tends to worsen the performance as well.

We can see that our method gives the smallest MSPE and significantly fewer false positives. Although SIS always gives no false positive, it tends not to select any variable, yielding higher MSPE. PICASSO tends to select many more variables, which causes high false positives as expected. The other three, SCAD, LASSO and BayesS5 have MSPEs quite close to BEL_HD but all with higher false positives.
Table 2.2: An empirical study using US crime data: average MSPE of each method under settings with different \( p/n \) ratios. The average numbers of original and noise variables selected by each method are listed in the parenthesis (original/noise). The smallest average MSPE is highlighted in bold for each setting.

<table>
<thead>
<tr>
<th>( p/n )</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASSO</td>
<td>0.12(1.04/1.13)</td>
<td>0.13(0.91/0.84)</td>
<td>0.13(0.86/1.07)</td>
</tr>
<tr>
<td>SCAD</td>
<td>0.11(2.55/5.29)</td>
<td>0.11(2.15/6.45)</td>
<td>0.12(1.78/8.68)</td>
</tr>
<tr>
<td>SIS</td>
<td>0.16(0.02/0)</td>
<td>0.16(0.01/0)</td>
<td>0.15(0.02/0)</td>
</tr>
<tr>
<td>PICASSO</td>
<td>0.15(5.53/25.9)</td>
<td>0.15(3.83/31.00)</td>
<td>0.14(2.15/34.17)</td>
</tr>
<tr>
<td>BayesS5</td>
<td>0.11(1.66/0.35)</td>
<td>0.11(1.62/0.38)</td>
<td>0.13(1.41/0.59)</td>
</tr>
<tr>
<td>BEL_HD</td>
<td><strong>0.10</strong> (0.96/0.04)</td>
<td><strong>0.10</strong> (0.95/0.05)</td>
<td><strong>0.11</strong> (0.92/0.09)</td>
</tr>
</tbody>
</table>

Table 2.3: An empirical study using US crime data: average MSPE of each method under settings where confounding variables with different correlations with the original variables were generated. The average numbers of original and noise variables selected by each method are listed in the parenthesis (original/noise). The smallest average MSPE is highlighted in bold for each setting.

<table>
<thead>
<tr>
<th>Correlation</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASSO</td>
<td>0.13(0.87/0.73)</td>
<td>0.14(0.94/1.08)</td>
<td>0.15(0.85/1.29)</td>
</tr>
<tr>
<td>SCAD</td>
<td>0.12(2.11/7.99)</td>
<td>0.12(2.06/6.31)</td>
<td>0.13(1.89/7.82)</td>
</tr>
<tr>
<td>SIS</td>
<td>0.17(0/0)</td>
<td>0.18(0.03/0)</td>
<td>0.19(0.02/0)</td>
</tr>
<tr>
<td>PICASSO</td>
<td>0.15(3.44/31.25)</td>
<td>0.16(3.54/31.25)</td>
<td>0.16(3.16/31.16)</td>
</tr>
<tr>
<td>BayesS5</td>
<td>0.12(1.50/0.52)</td>
<td>0.12(1.52/0.50)</td>
<td>0.12(1.38/0.63)</td>
</tr>
<tr>
<td>BEL_HD</td>
<td><strong>0.10</strong> (0.96/0.07)</td>
<td><strong>0.11</strong> (0.94/0.07)</td>
<td><strong>0.11</strong> (0.91/0.08)</td>
</tr>
</tbody>
</table>

2.4.2. Gene expression data example

The gene expression data used here was collected by Lan et al. [32], which studied an F2 segregating for obesity and diabetes to expose key components of gene regulatory networks. An F2 represents an allelic block permutation. They found that some lipid metabolism genes showed strong correlations with stearoyl-CoA desaturase 1 (SCD1) expression.
From each liver of selected 60 mice, 45,037 expression measurements were obtained using the Affymetrix MOE430 microarrays (MOE430A and MOE430B). The data was normalized by Robust Multi-array Average (RMA) method and is available at the NCBI website with accession number GSE3330. Shin et al. [64] illustrated the performance of BayesS5 using the first set of data (MOE430A, \( p \approx 23,000 \)), while here we used the whole data set (MOE430A and MOE430B, \( p \approx 45,000 \)). The data was randomly split into 5 test samples and 55 training samples, and the expression of gene SCD1 was used as the response variable. The average MSPE based on 100 replicates are reported in Table 2.4. We can find that BEL_HD produced the smallest MSPE among all the competing methods.

<table>
<thead>
<tr>
<th>Methods</th>
<th>SCAD</th>
<th>SIS</th>
<th>PICASSO</th>
<th>BayesS5</th>
<th>BEL_HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSPE</td>
<td>0.182</td>
<td>0.137</td>
<td>0.106</td>
<td>0.242</td>
<td>0.097</td>
</tr>
</tbody>
</table>

Table 2.4: Analysis of the gene expression data: average MSPE of each method.

2.5. Discussion

Ultra-high dimensional data are ubiquitous nowadays, which is a major challenge to traditional statistical techniques that are mostly designed for relatively low dimensional problems.

In recent ten years, several new parametric-based variable selection methods have been developed for ultra-high dimensional problems. On the contrary, there are few methods developed in a semi-parametric field. In this paper, we have developed a new Bayesian semi-parametric method, labeled BEL_HD, which requires the assumption of a linear structure only but no distributional assumptions. Inspired by Chang et al. [9], we have applied Laplace priors on both the regression coefficients and the Lagrange multipliers in a Bayesian setup to boost the sparsity of the final estimates. Furthermore, we
have adapted the path-wise coordinate optimization algorithm [19, 40, 76], which was proposed and applied in traditional likelihood approaches, to cope with the features of a Bayesian framework. The adjusted algorithm has been then fused seamlessly into the doubly penalized empirical likelihood Bayesian framework, which can screen potentially important variables and reduce the computational demand for ultra-high dimensional problems. An R package implementing our method is freely available from GitHub (https://github.com/canx2021/BEL_HD).

The numerical evidence from seven simulation studies and two real data analyses shows that our proposed method BEL_HD has better performance in parameter estimation and variable selection as compared to existing methods. It shows steadily good performance against varying scale and location of nonzero regression coefficients, different levels of sparsity, signal-to-noise ratios, $n/p$ ratios, and correlations of covariates. It is also robust to the violation of the normality assumption which is often assumed in ordinary regression models. The better performance of BEL_HD can be attributed to three aspects. First, the proposed method inherits the merits of empirical likelihood. Second, the idea of double penalties leads to a sparser solution as compared to traditional “single-penalty” approaches. Last but not the least, the adjusted path-wise coordinate optimization algorithm serves well as the screening process in the variable selection. It is also worth mentioning that the improved performance of BEL_HD comes at the cost of computing time.

In this paper, we focus on a continuous response variable. Nevertheless, future research can extend the proposed framework to include binary or multi-category models. Furthermore, the adjusted path-wise coordinate optimization algorithm can be combined with all penalized regressions in the Bayesian framework to solve ultra-high dimensional problems.
APPENDIX A

APPENDIX of CHAPTER 1

This appendix provides technical details of Bayesian posterior computation, additional simulation results, gene lists and validation results of the lung cancer data example mentioned in Chapter 1.

A.1. Full conditionals for the Gibbs sampler

The full conditional posterior distributions based on the priors specified in Section 1.2.3 are given below.

\[
\begin{align*}
\sigma_a^2 | \cdot &\sim IG \left( \delta + \frac{I}{2}, \zeta + \frac{\sum_{i=1}^{I}(a_i - \mu_a)^2}{2} \right) \\
\sigma_b^2 | \cdot &\sim IG \left( \delta + \frac{I}{2}, \zeta + \frac{\sum_{i=1}^{I}(b_i - \mu_b)^2}{2} \right) \\
\sigma_d^2 | \cdot &\sim IG \left( \delta + \frac{N}{2}, \zeta + \frac{\sum_{n=1}^{N}(d_n^{-})^2}{2} \right) \\
\sigma_d^2 | \cdot &\sim IG \left( \delta + \frac{R + P + H}{2}, \zeta + \frac{\sum_{h=1}^{R}(d_h^{+})^2 + \sum_{p=1}^{P}(d_p^{+})^2 + \sum_{r=1}^{R}d_r^2}{2} \right) \\
\sigma_e^2 | \cdot &\sim IG \left( \delta + \frac{IN}{2}, \zeta + \frac{\sum_{n=1}^{N}\sum_{i=1}^{I}(Y_{in}^- - a_i - b_i c - d_n^-)^2}{2} \right) \\
\sigma_e^2 | \cdot &\sim IG \left( \delta + \frac{I(R + P + H)}{2}, \zeta + \frac{\sum_{i=1}^{I}E_i}{2} \right)
\end{align*}
\]
where \( E_i = \sum_{p=1}^{P} (Y_{ip}^+ - a_i - b_iX_p^+ - d_p^+)^2 + \sum_{r=1}^{R} (Y_{ir} - a_i - b_i(\phi_i + \kappa_{ir}) - d_r)^2 + \sum_{h=1}^{H} (Y_{ih}^* - a_i - b_i(\phi_i + \kappa_{ih}^*) - d_h^*)^2. \)

\[
\begin{align*}
\sigma_{\kappa^*}^2 \mid \cdot & \sim IG \left( \delta + \frac{IH}{2}, \zeta + \frac{1}{2} \sum_{h=1}^{H} \sum_{i=1}^{I} (\kappa_{ih}^* - \mu_h^*)^2 \right), \\
\sigma_{\kappa}^2 \mid \cdot & \sim IG \left( \delta + \frac{IR + R}{2}, \zeta + \frac{1}{2} \sum_{r=1}^{R} \sum_{i=1}^{I} (\kappa_{ir} - \alpha_r - \beta_r Z_i)^2 + \sum_{r=1}^{R} \frac{\tau_r^2}{\lambda_r} \right), \\
\mu_a \mid \cdot & \sim N \left( \frac{\sum_{i=1}^{I} a_i}{\sigma_a^2} + \frac{\mu_a}{\sigma_a^2}, \frac{1}{\sigma_a^2} + \frac{1}{\sigma_a^2} \right), \\
\mu_b \mid \cdot & \sim N \left( \frac{\sum_{i=1}^{I} b_i}{\sigma_b^2} + \frac{\mu_b}{\sigma_b^2}, \frac{1}{\sigma_b^2} + \frac{1}{\sigma_b^2} \right), \\
c \mid \cdot & \sim N \left( \frac{\sum_{i=1}^{I} \sum_{n=1}^{N} b_i (Y_{in}^* - a_i - d_n)}{N \sum_{i=1}^{I} b_i^2}, \frac{\sigma_{-e}^2}{N \sum_{i=1}^{I} b_i^2} \right) I_{(-6, -1)}(c), \\
\kappa_{ih}^* \mid \cdot & \sim N \left( \frac{b_i(Y_{ih}^* - a_i - b_i(\phi_i - d_i))}{\sigma_{e_i}^2} + \frac{\mu_{ih}^*}{\sigma_{e_i}^2}, \frac{1}{\sigma_{e_i}^2} + \frac{1}{\sigma_{e_i}^2} \right), \\
\kappa_{ir} \mid \cdot & \sim N \left( \frac{b_i(Y_{ir}^* - a_i - b_i(\phi_i - d_r))}{\sigma_{e_i}^2} + \frac{\alpha_r + \beta_r Z_i}{\sigma_{e_i}^2}, \frac{1}{\sigma_{e_i}^2} + \frac{1}{\sigma_{e_i}^2} \right), \\
\mu_{ih}^* \mid \cdot & \sim N \left( \frac{\sum_{i=1}^{I} \kappa_{ih}^*}{I}, \frac{\sigma_{\kappa_h^*}^2}{I} \right) I_{(\lambda_{\kappa_h^*}, \eta_{\kappa_h^*})}(\mu_{ih}^*), \\
\alpha_r \mid \cdot & \sim N \left( \frac{\sum_{i=1}^{I} Z_{i} \kappa_{ir}}{I - \sum_{i=1}^{I} Z_{i}}, \frac{\sigma_{\kappa_r^2}}{I - \sum_{i=1}^{I} Z_{i}} \right) I_{(\lambda_{\kappa_r^2}, \eta_{\kappa_r^2})}(\alpha_r), \\
\beta_r \mid \cdot & \sim N \left( \frac{\sum_{i=1}^{I} Z_{i}^2 \tau_i^2 (\kappa_{ir} - \alpha_r)}{\sum_{i=1}^{I} Z_{i}^2 \tau_i^2 + 1}, \frac{2\sigma_{\kappa_r^2}^2 \tau_r^2}{\sum_{i=1}^{I} Z_{i}^2 \tau_i^2 + 1} \right), \\
\tau_r^2 \mid \cdot & \sim \text{Inverse Gaussian} \left( \sqrt{\frac{\lambda_r^2 \sigma_{\kappa_r^2}^2}{\beta_r^2}}, \lambda_r^2 \right), \\
\lambda_r^2 \mid \cdot & \sim \text{Gamma} \left( 1 + \eta_r, \frac{\tau_r^2}{2} + \nu \right), \\
d_{hi} \mid \cdot & \sim N \left( \frac{\sum_{i=1}^{I} (Y_{ih}^* - a_i - b_i \phi_i - d_i)}{\sigma_{e_i}^2} + \frac{1}{\sigma_{e_i}^2}, \frac{1}{\sigma_{e_i}^2} + \frac{1}{\sigma_{e_i}^2} \right).
\end{align*}
\]
\[
\begin{align*}
\Phi &= \sum_{i=1}^{I} (Y_{ip}^+ - a_i - b_i X_p^+) / \sigma^2_e + 1 / \sigma^2_d, \\
\Phi^* &= \sum_{i=1}^{I} (Y_{ip}^+ - a_i - b_i \phi_i + \kappa^*_{ih}) / \sigma^2_e + 1 / \sigma^2_d, \\
\Phi_d^* &= \sum_{i=1}^{I} (Y_{ip}^+ - a_i - b_i \phi_i + \kappa^*_{ih}) / \sigma^2_e + 1 / \sigma^2_d, \\
\Phi_e^* &= \sum_{i=1}^{I} (Y_{ip}^+ - a_i - b_i \phi_i + \kappa^*_{ih}) / \sigma^2_e + 1 / \sigma^2_d, \\
\Phi_a^* &= \sum_{i=1}^{I} (Y_{ip}^+ - a_i - b_i \phi_i + \kappa^*_{ih}) / \sigma^2_e + 1 / \sigma^2_d, \\
\Phi_b^* &= \sum_{i=1}^{I} (Y_{ip}^+ - a_i - b_i \phi_i + \kappa^*_{ih}) / \sigma^2_e + 1 / \sigma^2_d.
\end{align*}
\]

where \( \Phi = \sum_{h=1}^{H}(Y_{ih}^* - a_i - b_i \kappa^*_{ih} - d_h^*) + \sum_{r=1}^{R}(Y_{ir} - a_i - b_i \kappa_{ir} - d_r), \ A_1 = \sum_{n=1}^{N}(Y_{in}^* - b_i c - d_n^*), \ A_2 = \sum_{p=1}^{P}(Y_{ip}^+ - b_i X_p^+ - d_p^*), \ A_3 = \sum_{h=1}^{H}(Y_{ih}^* - b_i \phi_i + \kappa^*_{ih} - d_h^*), \ A_4 = \sum_{r=1}^{R}(Y_{ir} - b_i \phi_i + \kappa_{ir} - d_r); \ B_1 = \sum_{n=1}^{N}c(Y_{in}^* - a_i - d_n^*), \ B_2 = \sum_{p=1}^{P}X_p^+(Y_{ip}^+ - a_i - d_p^*), \ B_3 = \sum_{h=1}^{H}(\phi_i + \kappa^*_{ih})(Y_{ih}^* - a_i - d_h^*), \ B_4 = \sum_{r=1}^{R}(\phi_i + \kappa_{ir})(Y_{ir} - a_i - d_r), \ B_5 = \sum_{p=1}^{P}X_p^+ + \sum_{h=1}^{H}(\phi_i + \kappa^*_{ih})^2 + \sum_{r=1}^{R}(\phi_i + \kappa_{ir})^2.

As in RCRnorm, we find that only using data from positive controls to update \( \sigma^2_d \) and \( \sigma^2_e \) would greatly facilitate the convergence of the algorithm, which has little influence on the final results. Thus, for RCRdiff, \( \sigma^2_d \) and \( \sigma^2_e \) are also updated using the following simplified distributions:

\[
\begin{align*}
\sigma^2_d | \sim IG \left( \frac{\delta + \frac{P}{2}}{2}, \frac{\zeta + \frac{\sum_{p=1}^{P} (d_p^+)^2}{2}}{2} \right), \\
\sigma^2_e | \sim IG \left( \frac{\delta + \frac{IP}{2}}{2}, \frac{\zeta + \frac{\sum_{i=1}^{I} \sum_{p=1}^{P} (Y_{ip}^+ - a_i - b_i X_p^+ - d_p^+)^2}{2}}{2} \right).
\end{align*}
\]
A.2. Additional Results for Simulation I

To study the computational properties of RCRdiff, we plot the computation time (in log2 second) vs. the no. of samples $I$ and the no. of regular genes $R$, respectively, in Figure A.1 under the reference setting I1 described in Section 3.1 of the main manuscript, where we fix the DE mean and proportion ($p_{DE}$ and $\mu_\beta$) both at 0.5 and vary $I$ and $R$. It appears that the computation time of RCRdiff (using 6000 MCMC iterations) has a log-linear relationship with both $I$ and $R$. Further, RCRdiff is computationally stable and much less time consuming as compared to the empirical Bayes method NanoStringDiff. All other methods are non-Bayesian and take less time than RCRdiff and NanoStringDiff.

![Figure A.1: Simulation I for performance evaluation: computing time (in log2 second) of two Bayesian methods, RCRdiff (using 6,000 MCMC iterations) and NanoStringDiff, under the reference setting I1, where we fix the DE mean and proportion ($p_{DE}$ and $\mu_\beta$) both at 0.5 and vary $I$ and $R$.](image)

In Simulation I, ROC curves for settings I2–I5 are shown in Figures A.2 – A.5. In all the settings, RCRdiff consistently outperforms the other methods under all combinations of DE means and DE proportions.
Figure A.2: Simulation I for performance evaluation: ROC curves for nine combinations of $\beta_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in Setting I2, where gene expression variability is increased, compared to the reference setting I1.
Figure A.3: Simulation I for performance evaluation: ROC curves for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in Setting I3, where lane-by-lane variation is increased, compared to the reference setting I1.
Figure A.4: Simulation I for performance evaluation: ROC curves for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_3$ (mean DE level) in Setting I4, where probe-level variation is increased, compared to the reference setting I1.
Figure A.5: Simulation I for performance evaluation: ROC curves for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in Setting I5, where variability of random noise is increased, compared to the reference setting I1.

Tables A.1–A.3 compare F scores for the different methods, in which we varied the FDR threshold from the common choice 0.05 to 0.01, 0.1 and 0.15. Note that this change affects the performance of the existing methods but not RCRdiff or RCRdiff-S. The results show that RCRdiff and RCRdiff-S are always among the top performers in all cases.
<table>
<thead>
<tr>
<th></th>
<th>$\mu_\beta = 0.3$</th>
<th>$\mu_\beta = 0.5$</th>
<th>$\mu_\beta = 0.7$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p_{DE}$</td>
<td>$p_{DE}$</td>
<td>$p_{DE}$</td>
</tr>
<tr>
<td></td>
<td>0.3 0.5 0.7</td>
<td>0.3 0.5 0.7</td>
<td>0.3 0.5 0.7</td>
</tr>
<tr>
<td>RCRdiff</td>
<td>0.35 0.60 0.64</td>
<td>0.60 0.77 0.80</td>
<td>0.67 0.85 0.86</td>
</tr>
<tr>
<td>RCRdiff-S</td>
<td>0.43 0.58 0.61</td>
<td>0.68 0.77 0.79</td>
<td>0.79 0.86 0.84</td>
</tr>
<tr>
<td>NanoStringDiff</td>
<td>0.17 0.26 0.21</td>
<td>0.46 0.55 0.52</td>
<td>0.74 0.77 0.77</td>
</tr>
<tr>
<td>NanoStringNorm+t</td>
<td>0.01 0.01 0.01</td>
<td>0.04 0.06 0.07</td>
<td>0.30 0.35 0.34</td>
</tr>
<tr>
<td>NAPPA+t</td>
<td>0.04 0.05 0.05</td>
<td>0.18 0.18 0.25</td>
<td>0.50 0.57 0.59</td>
</tr>
<tr>
<td>NAPPA+edgeR</td>
<td>0.13 0.17 0.15</td>
<td>0.40 0.49 0.43</td>
<td>0.68 0.70 0.67</td>
</tr>
<tr>
<td>NAPPA+DESeq</td>
<td>0.17 0.20 0.18</td>
<td>0.47 0.52 0.48</td>
<td>0.72 0.73 0.68</td>
</tr>
<tr>
<td>TMM+edgeR</td>
<td>0.15 0.21 0.19</td>
<td>0.44 0.52 0.50</td>
<td>0.75 0.76 0.72</td>
</tr>
<tr>
<td>RLE+DESeq</td>
<td>0.20 0.25 0.22</td>
<td>0.50 0.57 0.53</td>
<td>0.77 0.77 0.73</td>
</tr>
<tr>
<td>RUV+edgeR</td>
<td>0.15 0.20 0.19</td>
<td>0.43 0.51 0.48</td>
<td>0.70 0.68 0.62</td>
</tr>
<tr>
<td>RUV+DESeq</td>
<td>0.19 0.24 0.21</td>
<td>0.46 0.57 0.53</td>
<td>0.76 0.76 0.72</td>
</tr>
</tbody>
</table>

Table A.1: Simulation I for performance evaluation: F scores based on the FDR threshold 0.01 for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data unless specified otherwise).
Table A.2: Simulation I for performance evaluation: F scores based on the FDR threshold 0.1 for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data unless specified otherwise).

<table>
<thead>
<tr>
<th>Method</th>
<th>$\mu_\beta = 0.3$</th>
<th>$\mu_\beta = 0.5$</th>
<th>$\mu_\beta = 0.7$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p_{DE}$</td>
<td>$p_{DE}$</td>
<td>$p_{DE}$</td>
</tr>
<tr>
<td></td>
<td>0.3  0.5  0.7</td>
<td>0.3  0.5  0.7</td>
<td>0.3  0.5  0.7</td>
</tr>
<tr>
<td>RCRdiff</td>
<td>0.35 0.60 0.64</td>
<td>0.60 0.77 0.80</td>
<td>0.67 0.85 0.86</td>
</tr>
<tr>
<td>RCRdiff-S</td>
<td><strong>0.43</strong> 0.58 0.61</td>
<td><strong>0.68</strong> 0.77 0.79</td>
<td><strong>0.79</strong> 0.86 0.84</td>
</tr>
<tr>
<td>NanoStringDiff</td>
<td>0.40 0.48 0.47</td>
<td>0.63 0.71 0.76</td>
<td>0.76 0.84 <strong>0.89</strong></td>
</tr>
<tr>
<td>NanoStringNorm+t</td>
<td>0.06 0.08 0.11</td>
<td>0.32 0.46 0.5</td>
<td>0.73 0.81 0.82</td>
</tr>
<tr>
<td>NAPPA+t</td>
<td>0.28 0.23 0.23</td>
<td>0.51 0.61 0.67</td>
<td>0.70 0.85 0.88</td>
</tr>
<tr>
<td>NAPPA+edgeR</td>
<td>0.33 0.43 0.41</td>
<td>0.61 0.68 0.67</td>
<td>0.75 0.79 0.81</td>
</tr>
<tr>
<td>NAPPA+DESeq</td>
<td>0.35 0.45 0.42</td>
<td>0.61 0.68 0.68</td>
<td>0.77 0.79 0.79</td>
</tr>
<tr>
<td>TMM+edgeR</td>
<td>0.36 0.45 0.44</td>
<td>0.65 0.71 0.71</td>
<td>0.77 0.82 0.83</td>
</tr>
<tr>
<td>RLE+DESeq</td>
<td>0.40 0.47 0.46</td>
<td>0.65 0.71 0.71</td>
<td>0.77 0.82 0.82</td>
</tr>
<tr>
<td>RUV+edgeR</td>
<td>0.36 0.43 0.44</td>
<td>0.62 0.67 0.65</td>
<td>0.71 0.73 0.73</td>
</tr>
<tr>
<td>RUV+DESeq</td>
<td>0.39 0.46 0.47</td>
<td>0.64 0.71 0.71</td>
<td>0.77 0.81 0.81</td>
</tr>
</tbody>
</table>
The false positive rates for each of the methods in null cases (i.e., no DE genes) of settings I1-I5 and II6 are shown in Table A.4. We can see that RCRdiff has competitive performance in all null cases. RCRdiff-S has more false positives than RCRdiff but still at an acceptable level.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\mu_\beta = 0.3$</th>
<th>$\mu_\beta = 0.5$</th>
<th>$\mu_\beta = 0.7$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p_{DE}$</td>
<td>$p_{DE}$</td>
<td>$p_{DE}$</td>
</tr>
<tr>
<td></td>
<td>0.3   0.5  0.7</td>
<td>0.3   0.5  0.7</td>
<td>0.3   0.5  0.7</td>
</tr>
<tr>
<td>RCRdiff</td>
<td>0.35  0.60  0.64</td>
<td>0.60  0.77  0.80</td>
<td>0.67  0.85  0.86</td>
</tr>
<tr>
<td>RCRdiff-S</td>
<td>0.43  0.58  0.61</td>
<td>0.68  0.77  0.79</td>
<td>0.79  0.86  0.84</td>
</tr>
<tr>
<td>NanoStringDiff</td>
<td>0.43  0.52  0.54</td>
<td>0.63  0.74  0.79</td>
<td>0.74  0.84  0.89</td>
</tr>
<tr>
<td>NanoStringNorm+t</td>
<td>0.13  0.13  0.18</td>
<td>0.45  0.56  0.62</td>
<td>0.76  0.84  0.87</td>
</tr>
<tr>
<td>NAPPA+t</td>
<td>0.33  0.31  0.29</td>
<td>0.55  0.68  0.74</td>
<td>0.70  0.86  0.89</td>
</tr>
<tr>
<td>NAPPA+edgeR</td>
<td>0.38  0.47  0.47</td>
<td>0.62  0.71  0.70</td>
<td>0.75  0.78  0.83</td>
</tr>
<tr>
<td>NAPPA+DESeq</td>
<td>0.39  0.49  0.48</td>
<td>0.62  0.70  0.71</td>
<td>0.75  0.80  0.80</td>
</tr>
<tr>
<td>TMM+edgeR</td>
<td>0.43  0.50  0.49</td>
<td>0.65  0.73  0.75</td>
<td>0.75  0.82  0.84</td>
</tr>
<tr>
<td>RLE+DESeq</td>
<td>0.43  0.52  0.51</td>
<td>0.66  0.72  0.74</td>
<td>0.75  0.81  0.83</td>
</tr>
<tr>
<td>RUV+edgeR</td>
<td>0.42  0.49  0.51</td>
<td>0.63  0.69  0.69</td>
<td>0.69  0.73  0.76</td>
</tr>
<tr>
<td>RUV+DESeq</td>
<td>0.43  0.50  0.52</td>
<td>0.66  0.73  0.74</td>
<td>0.75  0.81  0.82</td>
</tr>
</tbody>
</table>

Table A.3: Simulation I for performance evaluation: F scores based on the FDR threshold 0.15 for nine combinations of $p_{DE}$ (DE gene proportion) and $\mu_\beta$ (mean DE level) in the reference setting I1 (parameters estimated from the FFPE lung cancer data unless specified otherwise).
Table A.4: Simulation I for performance evaluation: false positive rates for each method when there is no DE genes for settings I1-I5 and II6.

A.3. Convergence Diagnostics

We illustrate how we applied convergence diagnostics to detect the MCMC convergence of the RCRdiff algorithm using the lung cancer data example in Section 4.1. Recall that we compared 28 FF samples with 28 FFPE samples, in which we expect no DE genes as they are from the same set of patients. To check the convergence, we ran five MCMC chains with over-dispersed starting points. The trace plots of key global parameters (mean intercept $\mu_a$ and mean slope $\mu_b$, the nonspecific RNA-binding level due to the background noise $c$) and the DE levels for three randomly selected regular genes ($\beta_{35}, \beta_{38},$ and $\beta_{77}$) for 6,000 iterations are shown in Figure A.6. Clearly, all chains converged quickly for each of the parameters. We also applied the Gelman-Rubin convergence diagnostics to these parameters. Their potential scale reduction factors (PSRFs) are very close to 1 ($\leq 1.02$). So is the multivariate PSRF. All these indicate that convergence is reached.
Figure A.6: Lung cancer data example: trace plots of key global parameters (the mean intercept $\mu_a$ and mean slope $\mu_b$, the nonspecific RNA-binding level due to the background noise $c$) and DE levels for three randomly selected regular genes ($\beta_{35}$, $\beta_{38}$, and $\beta_{77}$) in DE analysis between 28 FF and 28 FFPE samples. Five MCMC chains with over-dispersed starting points are shown using different colors.

### A.4. Gene lists used in the lung cancer data example

Table A.5 gives the lists of genes associated with adenocarcinoma (ADC) and squamous cell carcinoma (SCC), respectively. Both lists were provided by our collaborators from The University of Texas MD Anderson Cancer Center, who collected the data in Section 4.1.
<table>
<thead>
<tr>
<th>ADC</th>
<th>SCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTSH</td>
<td>PTGDS</td>
</tr>
<tr>
<td>CIRBP</td>
<td>EPB41L2</td>
</tr>
<tr>
<td>H2AFZ</td>
<td>CMAHP</td>
</tr>
<tr>
<td>CCT6A</td>
<td>IGHV7-81</td>
</tr>
<tr>
<td>PFKP</td>
<td>MS4A1</td>
</tr>
<tr>
<td>TXNDC9</td>
<td>MORN3</td>
</tr>
<tr>
<td>ECHDC2</td>
<td>SUPT20HL1</td>
</tr>
<tr>
<td>AK4</td>
<td></td>
</tr>
<tr>
<td>PLEK2</td>
<td></td>
</tr>
<tr>
<td>PBXIP1</td>
<td></td>
</tr>
<tr>
<td>SLC2A1</td>
<td></td>
</tr>
<tr>
<td>UCK2</td>
<td></td>
</tr>
<tr>
<td>RHOBTB2</td>
<td></td>
</tr>
<tr>
<td>RACGAP1</td>
<td></td>
</tr>
<tr>
<td>CCNA2</td>
<td></td>
</tr>
<tr>
<td>CENPF</td>
<td></td>
</tr>
<tr>
<td>CBX7</td>
<td></td>
</tr>
<tr>
<td>BUB1B</td>
<td></td>
</tr>
<tr>
<td>FOXM1</td>
<td></td>
</tr>
<tr>
<td>CERS4</td>
<td></td>
</tr>
<tr>
<td>POLQ</td>
<td></td>
</tr>
<tr>
<td>TRIP13</td>
<td></td>
</tr>
<tr>
<td>NEIL3</td>
<td></td>
</tr>
<tr>
<td>CENPA</td>
<td></td>
</tr>
<tr>
<td>COL4A4</td>
<td></td>
</tr>
<tr>
<td>CLU</td>
<td></td>
</tr>
</tbody>
</table>

Table A.5: Lung cancer data example: lists of genes associated with adenocarcinoma and squamous cell carcinoma.
A.5. DE gene validation in the lung cancer data example

As mentioned in Section 4.1, two genes, ECHDC2 (Enoyl-CoA Hydratase Domain Containing 2) and PTGDS (Prostaglandin D2 Synthase), are only detected as DE by RCRdiff. In order to validate whether these two genes are truly differentially expressed between ADC and SCC, we conducted analyses based on two datasets: Rousseaux_2013 dataset [58] and Sato_2013 dataset [55, 61]. The comparative analysis results of expression levels of ECHDC2 and PTGDS genes between ADC and SCC patients are shown in Figure A.7, confirming the DE status of these two genes.

Figure A.7: Lung cancer data example: comparative analysis of expression levels of ECHDC2 and PTGDS genes between ADC and SCC patients in the Rousseaux_2013 dataset (with 85 lung ADC patients and 100 lung SCC patients) and the Sato_2013 dataset (with 183 lung ADC patients and 80 lung SCC patients).
We also performed meta-analyses using Lung Cancer Explorer (LCE) [7] to test PTGDS expression in tumor vs. normal samples across six different datasets for lung ADC and SCC patients, which is shown in Figure A.8. As mentioned in Section 4.1, the meta-analysis results show that the PTGDS gene consistently expresses lower in tumor compared to normal lung tissues in lung ADC (p-value = $3e^{-21}$) and SCC (p-value = $4.4e^{-08}$).

Figure A.8: Lung cancer data example: meta-analyses of PTGDS gene expression in tumor vs. normal samples across using six different datasets from the LCE database for lung ADC and SCC patients, respectively.
BIBLIOGRAPHY


