Developing Statistical Methods For Data From Platforms Measuring Gene Expression

Gaoxiang Jia

Southern Methodist University, gjia@smu.edu

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

Part of the Applied Statistics Commons, Biostatistics Commons, Statistical Methodology Commons, and the Statistical Models Commons

Recommended Citation

https://scholar.smu.edu/hum_sci_statisticalscience_etds/1

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.
DEVELOPING STATISTICAL METHODS FOR DATA FROM
PLATFORMS MEASURING GENE EXPRESSION

Approved by:

Dr. Xinlei Wang
Professor of Statistics

Dr. Guanghua Xiao
Associate Professor of Biostatistics

Dr. Daniel Heitjan
Professor of Biostatistics

Dr. Lynne Stokes
Professor of Statistics
DEVELOPING STATISTICAL METHODS FOR DATA FROM
PLATFORMS MEASURING GENE EXPRESSION

A Dissertation Presented to the Graduate Faculty of the
Dedman College of Humanities and Sciences
Southern Methodist University
in
Partial Fulfillment of the Requirements
for the degree of
Doctor of Philosophy
with a
Major in Biostatistics
by
Gaoxiang Jia

B.S., Shandong University, China, 2012
M.S., University of Texas Southwestern Medical Center, 2014
M.S., Southern Methodist University, 2017

May 19, 2018
Copyright (2018)

Gaoxiang Jia

All Rights Reserved
ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisors Drs. Xinlei Wang and Guanghua Xiao. They are the greatest mentors I could imagine: worked with me closely on my research and always provided the help and advice I needed. Their enthusiasm for statistics and science inspired me in all the time of my research. Without their training, I would not be able to become a statistician. They do not only train me in statistics, but also showed me how to treat people with respect and kindness.

Secondly, I would like to thank the rest of my thesis committee members Drs. Daniel Heitjan and Lynne Stokes, for their insightful comments on my dissertation and dedication to organizing this new joint biostatistics PhD program between SMU and UTSW. I also want to thank all the other faculty members in the Department of Statistical Science at SMU for their excellent teaching and help during my PhD study. My appreciation also goes to the faculty members who have helped me during my research and statistical consulting projects in the Department of Clinical Science at UTSW.

I would also like to thank my friends Dateng Li, Zhiyun Ge and Zhengyang Zhou for helping me on my coursework and research. Thanks for making my life at SMU enjoyable even when the pressure from courses and research projects is high.

I would like to thank my friends Kai Huang, Tao Wang and his wife, Yulei Zhang and his wife, Yandan Yang, Yixun Xing, Xue Li, Lin Qiu and Xiujun Zhu for their help on my application to the PhD programs at UTSW or SMU and/or their suggestions and encouragement on my career.

Last but not the least, I would like to express my gratitude to my wife Jinyan Chan, my parents and other family members for always loving, supporting and believing in me.
Developing Statistical Methods for Data from Platforms Measuring Gene Expression

Advisors: Drs. Xinlei Wang & Guanghua Xiao
Doctor of Philosophy degree conferred May 19, 2018
Dissertation completed February 19, 2018

This research contains two topics: (1) PBNPA: a permutation-based non-parametric analysis of CRISPR screen data; (2) RCRnorm: an integrated system of random-coefficient hierarchical regression models for normalizing NanoString nCounter data from FFPE samples.

Clustered regularly-interspaced short palindromic repeats (CRISPR) screens are usually implemented in cultured cells to identify genes with critical functions. Although several methods have been developed or adapted to analyze CRISPR screening data, no single specific algorithm has gained popularity. Thus, rigorous procedures are needed to overcome the shortcomings of existing algorithms. We developed a Permutation-Based Non-Parametric Analysis (PBNPA) algorithm, which computes p-values at the gene level by permuting sgRNA labels, and thus it avoids restrictive distributional assumptions. Although PBNPA is designed to analyze CRISPR data, it can also be applied to analyze genetic screens implemented with siRNAs or shRNAs and drug screens. We compared the performance of PBNPA with competing methods on simulated data as well as on real data. PBNPA outperformed recent methods designed for CRISPR screen analysis, as well as methods used for analyzing other functional genomics screens, in terms of Receiver Operating Characteristics (ROC) curves and False Discovery Rate (FDR) control for simulated data under various
settings. Remarkably, the PBNPA algorithm showed better consistency and FDR control on published real data as well.

Formalin-fixed, paraffin-embedded (FFPE) samples have great potential for biomarker discovery, retrospective studies, and diagnosis/prognosis of diseases. However, their application is hindered by the unsatisfactory performance of traditional gene expression profiling techniques on damaged RNAs. NanoString nCounter platform is well suited for profiling of FFPE samples and measures gene expression with high sensitivity, which may greatly facilitate realization of scientific and clinical values of FFPE samples. However, methodological development for normalization, a critical step when analyzing this type of data, is far behind that for traditional technologies such as microarray. Existing methods designed for the platform use information from different types of internal controls separately and rely on an overly-simplified assumption that expression of housekeeping genes is constant across samples for global scaling. We construct an integrated system of random-coefficient hierarchical regression models to capture main patterns and characteristics observed from NanoString data of FFPE samples, and develop a Bayesian approach to estimate parameters and normalize gene expression across samples. Our method, labeled RCRnorm, incorporates information from all aspects of the experimental design, and simultaneously removes biases from various sources. Further, it eliminates the unrealistic assumption on housekeeping genes and offers great interpretability. Simulation and applications showed its superior performance.
LIST OF FIGURES ......................................................................................... x
LIST OF TABLES .......................................................................................... xi

CHAPTER

1. PBNPA: A PERMUTATION-BASED NON-PARAMETRIC ANALYSIS OF CRISPR SCREEN DATA ................................................................. 1

1.1. Introduction .......................................................................................... 1

1.1.1. CRISPR mediated genomic screens and other type of genomic screens 1

1.1.2. Review of existing methods for analyzing CRISPR screen data .... 2

1.1.3. Motivation of our study ................................................................. 4

1.1.4. Research overview ....................................................................... 4

1.2. Methods ............................................................................................. 5

1.2.1. A permutation-based non-parametric method for calculating \( p \)-values for genes ................................................................. 5

1.2.2. Combining \( p \)-values to handle replicates ................................... 6

1.3. Simulation studies .............................................................................. 6

1.3.1. Simulation strategy ................................................................. 6

1.3.2. Positive selection performance ............................................. 9

1.3.3. Negative selection performance ............................................. 9

1.4. Comparison of recall, precision and estimation of \( p \) values ........... 14

1.4.1. Handling replicates ................................................................. 15

1.5. Real data applications ........................................................................ 17

1.5.1. Consistency between replicates ........................................... 17

1.5.2. FDR control ................................................................. 19

1.6. Discussion ......................................................................................... 19
1.7. R package ................................................................. 21

2. RCRNORM: AN INTEGRATED SYSTEM OF RANDOM-COEFFICIENT HIERARCHICAL REGRESSION MODELS FOR NORMALIZING NA- NOSTRING NCOUNTER DATA FROM FFPE SAMPLES .................. 23

2.1. Introduction .............................................................. 23

2.2. Motivating example ...................................................... 27
  2.2.1. Data description .................................................. 27
  2.2.2. Exploratory analysis ............................................. 29
  2.2.3. Proposed data model based on RCR ............................ 33

2.3. Bayesian approach ...................................................... 36
  2.3.1. Full probability model .......................................... 36
  2.3.2. Prior specification .............................................. 37
  2.3.3. Posterior computation and Bayesian inference ............... 38

2.4. Simulation ............................................................... 38
  2.4.1. Settings .......................................................... 38
  2.4.2. Results .......................................................... 40

2.5. Real data applications ................................................. 43
  2.5.1. Lung cancer data ............................................... 43
  2.5.2. Colorectal cancer data ........................................ 49

2.6. Discussion ............................................................. 50

2.7. R package ............................................................... 53

APPENDIX

A. PBNPA: A PERMUTATION-BASED NON-PARAMETRIC ANALYSIS OF CRISPR SCREEN DATA .................................................. 54

A.1. Figures ................................................................. 54

A.2. R package code for PBNPA ........................................... 54
B. RCRNORM: AN INTEGRATED SYSTEM OF RANDOM-COEFFICIENT HIERARCHICAL REGRESSION MODELS FOR NORMALIZING NANOS-TRING NCounter DATA FROM FFPE SAMPLES .............................. 63

B.1. Tables and figures .......................................................... 63
   B.1.1. Additional information about lung cancer data ......................... 63
   B.1.2. Additional simulation results ............................................ 64
   B.1.3. Results for colorectal cancer data ...................................... 64

B.2. Full conditionals ............................................................. 64

B.3. R package code for RCRnorm ................................................ 72

BIBLIOGRAPHY ........................................................................ 85
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Simulation evaluation of positive selection performance</td>
<td>10</td>
</tr>
<tr>
<td>1.2</td>
<td>Simulation evaluation of positive selection performance</td>
<td>11</td>
</tr>
<tr>
<td>1.3</td>
<td>Simulation evaluation of negative selection performance</td>
<td>12</td>
</tr>
<tr>
<td>1.4</td>
<td>Simulation evaluation of negative selection performance</td>
<td>13</td>
</tr>
<tr>
<td>1.5</td>
<td>Simulation evaluation of positive selection performance</td>
<td>16</td>
</tr>
<tr>
<td>1.6</td>
<td>Comparing consistency of MAGeCK and PBNPA on replicates</td>
<td>18</td>
</tr>
<tr>
<td>2.1</td>
<td>Exploratory analysis of lung cancer data</td>
<td>31</td>
</tr>
<tr>
<td>2.2</td>
<td>Simulation study I</td>
<td>42</td>
</tr>
<tr>
<td>2.3</td>
<td>Simulation study II</td>
<td>44</td>
</tr>
<tr>
<td>2.4</td>
<td>Simulation study III</td>
<td>45</td>
</tr>
<tr>
<td>2.5</td>
<td>Lung cancer data: posterior densities of global parameters</td>
<td>46</td>
</tr>
<tr>
<td>A.1</td>
<td>Simulation evaluation of positive selection performance</td>
<td>55</td>
</tr>
<tr>
<td>A.2</td>
<td>Simulation evaluation of positive selection performance</td>
<td>56</td>
</tr>
<tr>
<td>A.3</td>
<td>Simulation evaluation of negative selection performance</td>
<td>57</td>
</tr>
<tr>
<td>A.4</td>
<td>Simulation evaluation of negative selection performance</td>
<td>58</td>
</tr>
<tr>
<td>A.5</td>
<td>Simulation evaluation of negative selection performance based on recall,</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>precision and F_1</td>
<td></td>
</tr>
<tr>
<td>B.1</td>
<td>Lung cancer data: read counts of 6 positive controls</td>
<td>64</td>
</tr>
<tr>
<td>B.2</td>
<td>Simulation study I</td>
<td>65</td>
</tr>
<tr>
<td>B.3</td>
<td>Simulation study II</td>
<td>66</td>
</tr>
<tr>
<td>B.4</td>
<td>Simulation study III</td>
<td>67</td>
</tr>
<tr>
<td>B.5</td>
<td>Colorectal cancer data: posterior densities of global parameters</td>
<td>68</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Comparison of FDR control between MAGeCK and PBNPA</td>
<td>19</td>
</tr>
<tr>
<td>2.1</td>
<td>Lung cancer data: posterior summary statistics</td>
<td>45</td>
</tr>
<tr>
<td>2.2</td>
<td>Lung cancer data: summary statistics of Pearson correlation coefficients</td>
<td>48</td>
</tr>
<tr>
<td>2.3</td>
<td>Colorectal cancer data</td>
<td>50</td>
</tr>
<tr>
<td>B.1</td>
<td>Lung cancer data: data structure from the NanoString nCounter platform</td>
<td>63</td>
</tr>
<tr>
<td>B.2</td>
<td>Colorectal cancer data: posterior summary statistics</td>
<td>67</td>
</tr>
</tbody>
</table>
This dissertation is dedicated to my family and friends.
Chapter 1

PBNPA: A PERMUTATION-BASED NON-PARAMETRIC ANALYSIS OF CRISPR SCREEN DATA

1.1. Introduction

1.1.1. CRISPR mediated genomic screens and other type of genomic screens

The CRISPR (clustered regularly-interspaced short palindromic repeats) interference technique is widely used in biomedical studies to investigate gene functions. Large-scale screening with this technique has become a powerful tool in identifying cancer-promoting genes, drug-resistant genes, and genes that play pivotal roles in various biological processes (Shalem et al., 2015; Hart et al., 2015; Wang et al., 2014). The CRISPR/Cas9 system is composed of sgRNAs (single guide RNA) and Cas9s (CRISPR associated protein 9); an sgRNA contains around a 20-bp guide sequence that complements a DNA sequence and thus targets a gene of interest, and a Cas9 is a nuclease that induces double-strand breaks in the DNA and results in non-homologous end joining (NHEJ) repair. NHEJ is an error-prone repair mechanism that usually introduces an indel mutation that is highly likely to cause a coding frameshift, which leads to a premature stop codon and initiates the nonsense-mediated decay of the transcribed mRNA (Shalem et al., 2015). Thus, the CRISPR system abolishes the gene function by interfering with gene expression from the DNA level. This is more powerful than siRNA (small interfering RNA) or shRNA (short hairpin RNA) screens. An siRNA contains 20 ~ 25 bp short synthesized RNAs that function in the RNA interference pathway, and it cannot be integrated into a host genome. An shRNA contains synthesized double-stranded RNA molecules with a tight hairpin turn, which can be integrated into a host genome; however, it inhibits the gene function at the mRNA level (Gilbert et al.,
All three types of screens are usually implemented on cultured cells: siRNA screens are carried out in multi-well plates with each well containing one or several siRNAs targeting the same gene, and the signal in each well is collected as the read for that well; by contrast, CRISPR and shRNA screens are carried out in a pooled manner, where a mixture of lentivirus that contains RNAi reagents (either shRNA or sgRNA) targeting different genes is transfected into the same plate of cultured cells, and the microarray or next generation sequencing (NGS) technique can be used to collect reads. Cas9-sgRNA screens are performed with pre-designed sgRNA libraries that contain sgRNA redundancy. Generally, multiple sgRNAs (usually ranging from 3-10) with different sequences that target distinct locations on the same gene are utilized to ensure screening accuracy (Shalem et al., 2015). All genome-wide CRISPR screens use cell growth as a phenotypic measure. Based on the goal of the screens, they can be divided into positive selection screens and negative selection screens (Shalem et al., 2014). Positive screens aim to identify genes that inhibit cell growth in certain circumstances or that sensitize cells to a drug treatment or toxin. For example, genes protecting cells against toxins, which are likely to be receptors for the toxins, or genes involved in downstream signaling pathways (Koike-Yusa et al., 2014), may be targeted by positive screens. Under a strong selective pressure, cells with sgRNAs that confer resistance against that pressure would be enriched, and thus their signals are often strong and easy to detect. Negative selection screens aim to identify genes that promote cell growth or housekeeping genes (Morgens et al., 2016). In this scenario, cells that carry sgRNAs targeting such genes will be depleted during selection. Signals from negative screens are typically not as strong as those from positive screens, because the depletion level is usually mild and the number of depleted sgRNAs is large when considering the number of housekeeping genes (and thus they can be hard to separate from the background.)

1.1.2. Review of existing methods for analyzing CRISPR screen data

There are existing methods that can be used to analyze genome-wide RNA interfering screening results, including RSA (Konig et al., 2007), RIGER (Luo et al., 2008), MAGeCK
The Redundant siRNA Activity (RSA) method was originally developed to analyze data generated by large-scale small interfering RNA (siRNA) screens in mammalian cells (Konig et al., 2007). RSA calculates a \( p \)-value for each gene based on an iterative hypergeometric distribution formula, where a smaller \( p \)-value indicates the gene is more likely to have higher activity. RNAi Gene Enrichment Ranking (RIGER) was originally designed to identify essential cell genes in genome-scale pooled shRNA screens (Luo et al., 2008). It calculates the rank of each sgRNA based on a signal-to-noise metric and then synthesizes information on sgRNAs targeting the same gene in a way similar to that of Gene Set Enrichment Analysis to rank genes (Subramanian et al., 2005). Model-based Analysis of Genome-wide CRISPR/Cas9 Knockout (MAGeCK) and Screening Bayesian Evaluation and Analysis Method (ScreenBEAM) were both designed to analyze CRISPR screen data. MAGeCK evaluates sgRNAs based on \( p \)-values calculated from fitting a negative binomial model (Li et al., 2014), and then the ranks of sgRNAs targeting the same gene are combined with a modified version of robust ranking aggregation (RRA) called -RRA. ScreenBEAM assesses the gene level activity with Bayesian hierarchical models (Yu et al., 2016), in which within-gene variances were modeled as random effects. Among the above methods, RIGER, MAGeCK and ScreenBEAM can perform both positive and negative selection. In addition, several algorithms used for analysis of Next Generation Sequencing (NGS) data, such as edgeR (Robinson and McCarthy, 2010), DESeq (Anders and Huber, 2012) baySeq (Hardcastle and Kelly, 2010), NOISeq (Tarazona et al., 2012) and SAMseq (Li and Tibshirani, 2013), can also be used to analyze RNAi screening data. Although such methods can only assign ranks at the sgRNA level, they can be used to conduct gene-level inference (Li et al., 2014) when combined with existing methods of integrating group information. It is worth noting that NOISeq and SAMseq both take nonparametric approaches. Unlike our method that is based on permutation, SAMseq mainly relies on the two-sample Wilcoxon statistic to estimate the significance; and NOISeq assesses the significance of the treatment effect with the reference distribution generated by comparing reads of each gene in samples under the same condition.
1.1.3. Motivation of our study

Although many CRISPR screen analysis methods are available, no single specific algorithm has gained popularity from researchers, mainly due to one or more of the drawbacks listed below: (1) Distributions assumed are doubtful or incorrect and thus incapable of modeling data variability from different sources. Researchers generally use negative binomial or Poisson distributions to model read counts from NGS (Seyednasrollah et al., 2015). However, these distributions do not reflect certain characteristics of NGS data generating processes and are weak in handling over-dispersion. (2) Most studies compared their model performance using some ‘oracle’ datasets. However, the performance may be compromised when generalizing these methods to datasets from different conditions or platforms. This is reflected by the fact that the number of consistently identified genes across different algorithms is often small (Diaz et al., 2015). (3) Published methods usually have loose or no false discovery rate (FDR) control. FDR reflects the rate of type I errors when performing multiple hypothesis tests and influences the credibility of the tests if not carefully controlled. False discovery is a big concern for functional genomic studies when a large number of statistical tests are performed (Pawitan et al., 2005). The above-mentioned methods tend to overlook FDR or be ineffective in controlling it, as will be shown in detail in the Real Data Application section. Without stringent FDR controlled $p$-values, it is difficult to evaluate the statistical significance of selected genes.

1.1.4. Research overview

Our proposed method, Permutation-Based Non-Parametric Analysis (PBNPA) of CRISPR screen data, mitigates the three major drawbacks of existing CRISPR methods. First, PBNPA computes $p$-values at the gene level by permuting sgRNA labels, and thus it avoids restrictive distributional assumptions. Second, PBNPA shows superior performance to other algorithms in simulation using data generated to mimic the NGS sequencing process, which avoids overfitting based on specific datasets. Application to real data confirms better con-
sistency of PBNPA. Last, our data application reveals that PBNPA outperformed its competitors in terms of FDR control.

1.2. Methods

1.2.1. A permutation-based non-parametric method for calculating p-values for genes

In a CRISPR screen dataset, assume $Y_{ij}$ is the read count for the $j$th sgRNA in the library under condition $i$, where $j = 1, 2, \cdots, J$ indexes sgRNAs in the library; and $i = 0, 1$ indexes two experimental conditions, with $i = 0$ for the control and $i = 1$ for the treatment. We use $I_g$ to denote the index set of the sgRNAs that target the same gene $g$ and $\bigcup_{g=1}^{G} I_g = \{1, 2, \cdots, J\}$, where $g = 1, 2, \cdots, G$ and $G$ is the total number of genes in the library. Raw read counts in each condition $i$ were normalized by multiplying a factor of $\text{mean}(\sum_{j=0}^{J} Y_{0j}, \sum_{j=1}^{J} Y_{1j})/\sum_{j=1}^{J} Y_{ij}$. This makes total read counts in each condition equal without losing any useful information. Our PBNPA algorithm is outlined below.

1. For each sgRNA $j$ ($j = 1, 2, \cdots, J$), calculate the natural logarithm fold change of normalized read counts: $r_j = \log(Y_{1j}/Y_{0j})$. Then for each gene $g$, use the median of $r_j$’s ($j \in I_g$) as the $R$ score, denoted by $R_g$.

2. Randomly permute gene labels while holding $(Y_{0j}, Y_{1j})$ pairs unchanged to get permuted $R$ scores for each gene, denoted by $R_{gt}$’s, where $g = 1, 2, \cdots, G$.

3. Repeat step 2 for $T$ times and pool all $R_{gt}$’s over the $T$ permutations and all genes to form a null distribution of $R$.

4. Calculate the $p$ value for gene $g$ if it is a positively selected gene as:

$$p = \frac{\text{# of permuted R scores} > R_g}{\text{total # of permuted R scores}};$$

and the value for gene if it is a negatively selected gene as:

$$p = \frac{\text{# of permuted R scores} < R_g}{\text{total # of permuted R scores}}.$$
5. After getting $p$ values for all genes, remove genes with $p$ values smaller than a threshold, which are considered to be significant genes. Then repeat step 2 and 3 to get the null distribution with significant genes removed. Get updated $p$ values for each gene as described in step 4.

6. Use the Benjamini-Hochberg procedure to control FDR (Benjamini and Hochberg, 1995).

In this algorithm, the median log fold change of sgRNAs targeting a gene is used as the score of that gene, which makes it more robust against any outliers and influences from potential off-target effects. In step 5, we remove a small portion of genes with the purpose of removing any significant genes to get a more accurate estimate of the null distribution, as the null distribution is likely to be distorted if these significant genes are kept in the permutation process.

1.2.2. Combining $p$-values to handle replicates

A CRISPR screen experiment may contain several replicates. We analyzed each replicate using the proposed algorithm and then employed Fisher’s method to combine $p$-values from replicates for each gene (Brown, 1975; Rau et al., 2014). According to Fisher’s method, the statistic $-2 \sum_{s=1}^{S} \ln p_{gs}$, with $p_{gs}$ representing gene $g$’s $p$ value from the $s$th replicate, follows an $\chi^2$ distribution with $2S$ degrees of freedom under the null hypothesis $H_0$: gene $g$ has no effect, from which a combined $p$ value for each gene $g$ is obtained (Brown, 1975).

1.3. Simulation studies

1.3.1. Simulation strategy

To mimic the nature of RNA-seq experiments, the read counts of all sgRNAs under a given condition were generated from a Dirichlet-multinomial (DM) distribution. Considering the experimental setup of CRISPR screening with RNA-seq, each sgRNA in a library can be viewed as an outcome category in a multinomial distribution when the total read count
(sequencing depth) is fixed. However, the literature indicates that multinomial distributions are inadequate to model the extra variability that is usually observed in NGS data (Chen and Li, 2013; Bonafede et al., 2016). To account for over-dispersion, the probability vector of an NGS read falling into the different sgRNA categories is modeled as random variables from a Dirichlet distribution. After combining the multinomial model with the Dirichlet model, the mixture model is a Dirichlet-multinomial model with the probability mass function (PMF) shown below:

\[
f(Y_i) = \frac{\Gamma(Y_{i+} + 1)\Gamma(\gamma_{i+})}{\Gamma(Y_{i+} + \gamma_{i+})} \prod_{j=1}^{J} \frac{\Gamma(Y_{ij} + \gamma_{ij})}{\Gamma(Y_{ij} + 1)\Gamma(\gamma_{ij})}
\]

where \(Y_i = [Y_{i1}, Y_{i2}, \cdots, Y_{ij}]\), \(Y_{i+} = \sum_{j} Y_{ij}\), \(\gamma_{i+} = \sum_{j} \gamma_{ij}\) with \(\gamma_{ij}\)'s being the parameters of the DM distribution; and \(E(Y_{ij}) = Y_{i+} \frac{\gamma_{ij}}{\gamma_{i+}}\) and \(Var(Y_{ij}) = Y_{i+} \frac{\gamma_{ij}}{\gamma_{i+}} (1 - \frac{\gamma_{ij}}{\gamma_{i+}}) \frac{Y_{i+} + \gamma_{i+}}{1 + \gamma_{i+}}\) (Chen and Li, 2013; Tu, 2014). Compared to the variance of the multinomial model, the variance of the DM model is increased by a factor of \(\frac{Y_{i+} + \gamma_{i+}}{1 + \gamma_{i+}}\). When the total read count \(Y_{i+}\) is fixed, \(\gamma_{i+}\) controls the degree of overdispersion with a smaller value indicating larger overdispersion.

To simulate read counts for a screen experiment, we first generated \(\gamma_{0j}\)'s for a control sample from a negative binomial distribution \(NB(q, p)\) where \(q\) is the number of successful trials to be reached and \(p\) is the probability of success in each trial. We set \(q = 3\) and \(p = 0.08\) so that the generated DM read counts are right skewed, which approximately mimics real data. We link \(\gamma_{ij}\) to the effect of sgRNA \(j\) through the relationship \(\gamma_{ij} = \exp(\alpha_j + \beta_j \times i)\), where \(\alpha_j\) loosely reflects the log mean read count under the control and \(\beta_j\) represents the \(j\)th sgRNA effect (i.e., the log difference in mean read count between the treatment and control). The total number of genes \(G\) was set to be 10000. For genes that have effects during the screen processes under different conditions (which are referred to as true hits), we first generated the sgRNA effects targeting gene \(g\) from a normal distribution, \(\beta_j \sim N(\mu_g, \sigma^2)\) for \(j \in I_g, g = 1, 2, \ldots, G\), with gene-specific mean \(\mu_g\) and constant standard deviation \(\sigma = 0.4\) (0.4 was chosen to be close to the standard deviation estimated from real data); and then we forced all \(\beta_j\)'s for gene \(g\) to have the same sign as \(\mu_g\). The vector, which
contains different levels of $\mu_g$ in our simulation, was set to be $[1.5, 1, 0.5, -1, -2, -3]$, where a positive number indicates that a gene’s ablation promotes cell growth while a negative number indicates a gene is necessary for cell growth. The three levels of $\mu_g$ for each sign represent the high/medium/low effects of positively/negatively selected genes, respectively. There are 50 genes simulated from each level of $\mu_g$. Thus, among the 10000 genes, there are 150 positively selected genes and 150 negatively selected genes. For those genes with no effects, $\beta_j$’s were set to be 0.

Off-target effects of CRISPR are often caused by unintended DNA cleavage at non-targeting sites as a result of mismatch between DNA and sgRNA (Cho et al., 2014). If an sgRNA is an off-target effect, its read count may either decrease, increase, or remain the same since most DNA sequences in the human genome have no known function. In our simulation, off-target $\beta_j$’s were simulated from $N(0, \sigma^2)$ and then used to replace a certain proportion of randomly-selected on-target sgRNAs. The off-target rate of a library can be considered an important characteristic reflecting the quality of the library, which is determined by the algorithm used to design the sgRNAs (Xu et al., 2015). Although several experimental approaches exist, it is still challenging to get accurate estimates of sgRNA off-target rates (Wu et al., 2014; Zhang et al., 2015). Reported off-target rates vary greatly in the literature (Fu et al., 2013; Haeussler et al., 2016) and can range between 1% and 20% in most sgRNA libraries. Thus, we tested 4 off-target proportion values: 1%, 5%, 10% and 20%, to represent sgRNA libraries of different quality.

Besides the library quality, the number of sgRNAs per gene is another factor that is known to influence the screen performance dramatically. Thus, we varied the number of sgRNAs per gene from 2 to 6 as well.

With $\beta_j$’s simulated for all sgRNAs, we obtained $\gamma_{1j} = \gamma_{0j} \exp(\beta_j)$. Then we simulated $Y_{ij}$ from the DM distribution with $\gamma_{ij}$’s from statistical packages ‘multinomRob’ (Mebane Jr and Sekhon, 2009) and ‘dirmult’ (Tvedebrink, 2010).
1.3.2. Positive selection performance

We compared the performance of PBNPA, RSA, ScreenBEAM and MAGeCK for the four different off-target rates (1%, 5%, 10%, 20%), as mentioned in the simulation strategy section, when there are 3 sgRNAs targeting each gene. A receiver operating characteristic (ROC) curve plots the true positive rate against the false positive rate of a binary classifier for different possible cut-off points and visualizes the performance of the classifier. As shown in Figure 1.1, PBNPA works better for positive screening than RSA, MAGeCK and ScreenBEAM in terms of the ROC curve and area under the curve (AUC), regardless of the off-target proportion. Also, all the algorithms show worse performance with an increasing off-target rate except for RSA, whose AUC increases from 0.592 to 0.637. Figure 1.2 indicates that PBNPA outperforms the other algorithms with varying numbers of sgRNAs per gene from 2 to 5. As expected, the AUC of each method increases with an increasing number of sgRNAs per gene, as more sgRNAs enable better estimation of gene effects.

As we have discussed previously, $\gamma_{ij}$ controls the degree of over-dispersion. To check the performance of the algorithms with an increased over-dispersion level, we divided every $\gamma_{ij}$ by 10 and report the results in Figures A.1 and A.2 of Appendix: the performance of nearly all algorithms decreases compared with the low over-dispersion setting, but the performance of PBNPA and ScreenBEAM is comparable, and it is better than RSA and MAGeCK.

1.3.3. Negative selection performance

For negative selection, PBNPA and RSA have similar AUCs and perform better than MAGeCK and ScreenBEAM when the proportion of off-target sgRNAs is low, as shown in Figure 1.3. When the proportion of off-target sgRNAs increases, RSA shows some advantage over PBNPA and is robust against this increase. Figure 1.4 shows that when we fix the off-target proportion at 10% and vary the number of sgRNAs per gene, PBNPA and RSA have comparable performance, and they are significantly better than MAGeCK and ScreenBEAM when the number of sgRNAs per gene is low.
Figure 1.1. Simulation evaluation of positive selection performance. ROC curves and AUCs are shown for different algorithms with an increasing off target proportion while the number of sgRNAs per gene is fixed at 3. Each curve represents the average of ROC curves for 50 simulated datasets and above.
Figure 1.2. Simulation evaluation of positive selection performance. ROC curves and AUCs are shown for different algorithms with an increasing number of sgRNAs per gene, while the off target proportion is fixed at 10%.
Figure 1.3. Simulation evaluation of negative selection performance. ROC curves and AUCs are shown for different algorithms with an increasing off target proportion, while the number of sgRNAs per gene is fixed at 3.
Figure 1.4. Simulation evaluation of negative selection performance. ROC curves and AUCs are shown for different algorithms with an increasing number of sgRNAs per gene, while the off target proportion is fixed at 10%.
In the setting of high over-dispersion, RSA is the best among all and PBNPA is only second to RSA with increasing off-target proportion in the simulated datasets, as shown in Figure A.3 (Appendix). Figure A.4 (Appendix) shows that when we fix the off-target proportion and vary the number of sgRNAs per gene, RSA is slightly better than PBNPA, and they are better than the other two algorithms across different numbers of sgRNAs per gene. Overall, for negative selection, RSA seems to be the winner; but PBNPA provides quite close or comparable performance to RSA, which is much better than MAGeCK and ScreenBEAM.

1.4. Comparison of recall, precision and estimation of $p$ values

When multiple statistical tests are performed simultaneously in the analysis of a dataset, adjustment of $p$ values is needed. Among the four algorithms, RSA does not provide a method to adjust for multiple comparison. We applied the Benjamini-Hochberg (BH) procedure (Benjamini and Hochberg, 1995) to the results from RSA and obtained FDR-adjusted $p$ values. The other three methods use the BH procedure by default. Then we controlled FDR at 5% and compared recall (percent of identified true hits among all true hits), precision (percent of identified true hits among all selected genes) and $F_1$ of the four algorithms, where $F_1$ is a metric that balances recall and precision and is defined as $F_1 = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$. To our surprise, when FDR was controlled at 5%, neither RSA nor ScreenBEAM was able to identify any significant genes. Actually, under most settings, all genes in the RSA results had an adjusted $p$-value of 1. This suggests that RSA and ScreenBEAM cannot accurately estimate the statistical significance of the genes. Thus, we compared the recall, precision and $F_1$ of PBNPA and MAGeCK. Figure 1.5 shows the recall, precision and $F_1$ of PBNPA and MAGeCK for different combinations of sgRNA number per gene (2, 3, 4, 5, 6) and off-target rates (1%, 5%, 10%, 20%) for positive screens. From the bottom panel of Figure 1.5, it is clear that under most settings, $F_1$ of PBNPA is the same as or slightly better than that of MAGeCK. However, the recall of PBNPA is significantly better than that of MAGeCK, especially when the number of sgRNAs per gene is small. In the middle
panel, MAGeCK consistently maintains very high precision across all the settings. However, MAGeCK tends to be too conservative in identifying true hits and may show a lack of power. Note that when the off-target rate is high (20%) with 2 sgRNAs per gene, MAGeCK has a recall rate of less than 10%, where it cannot identify any true hits at all in some simulated datasets. In screening experiments, after the genome-wide screening, a secondary screening will typically be used to validate hits from the first round (Miles et al., 2016). This highlights the importance of the recall rate: those false positives are likely to be removed in the secondary screening, while those false negatives can be crucial genes that will be missed permanently. Nearly the same pattern can be observed for negative screens, as shown in Figure A.5 (Appendix). Thus, PBNPA provides the most accurate estimation of adjusted $p$ values among the four algorithms and also offers optimal recall rates.

1.4.1. Handling replicates

The comparisons we have discussed above are based on simulated data with no replicates. For low-quality screens, replicates are typically used to increase the power of identification. To handle screens with replicates, we propose to use Fisher’s method to combine $p$ values, as mentioned in the Methods section, followed by FDR adjustment. We simulated replicate datasets with parameters of the DM distribution set as $\gamma_{ij}$, which has higher over-dispersion than the DM distribution with $\gamma_{ij}$ and so may represent data of low quality. We evaluated 3 simulated replicates independently. Among the 150 positively selected genes, the analysis of individual replicates gives the following results (i.e., number of true hits identified/number of genes identified by PBNPA) with FDR controlled at 5%: 6/7, 9/11, and 8/9, respectively. After combining $p$ values for the first two replicates, the result is 72/86. After combining $p$ values for all three replicates, the result is 96/111. It is evident that PBNPA shows dramatically improved performance when even a small number of replicates are present.
Figure 1.5. Simulation evaluation of positive selection performance based on recall, precision and $F_1$ for different combinations of sgRNA number per gene (2~6) and off target ratio. Each bar represents the average of 50 simulated datasets and the standard error is indicated on the bar.
1.5. Real data applications

1.5.1. Consistency between replicates

Although the performance of various algorithms usually does not differ greatly in simulation studies, they tend to give quite different inferences on real data. This can be due to the fact that a simulation is not an exact reproduction of the complex data generation process in the real world. This phenomenon is also observed in algorithms analyzing CRISPR data (Diaz et al., 2015). From the simulation study, we have found that PBNPA and MAGeCK are handy to use and give better overall performance than the other two algorithms. Thus, we used datasets from two recently published articles to evaluate the consistency between these two algorithms as well as the consistency of the same algorithm on different replicates from the same experiment, since a good algorithm should give highly similar results on replicates of the same experiment. The KBM7 dataset is from a study with two replicates and 10 sgRNAs per gene, which aims to identify essential genes in the human genome to reveal genes that are oncogenic drivers or lineage specifiers (Wang et al., 2015). As shown in the upper panel of Figure 1.6, the identified hits are highly overlapped between the two algorithms for the same replicate, as well as between the two replicates with the same algorithm. This indicates both algorithms perform well on this dataset with high consistency. The Toxoplasma dataset is from a study with four replicates, which aims to identify essential genes of parasites for infection of human fibroblasts (Sidik et al., 2016). The library was designed to target more than 8000 protein coding genes in T. gondii with 10 sgRNAs per gene. For this dataset, the number of consistently identified genes for PBNPA is significantly higher than that identified by MAGeCK among the 4 replicates, as is shown in the middle and bottom panels of Figure 1.6. For PBNPA, there are 19 genes consistently identified in all four replicates and 80 genes consistently identified in at least three replicates. However, for MAGeCK there is no gene identified in all four replicates and only 11 genes consistently identified in at least three replicates. This is strong evidence that PBNPA has superior consistency and better FDR control than MAGeCK.
Figure 1.6. Comparing consistency of MAGeCK and PBNPA on replicates using real data. Upper panel: overlap of PBNPA and MAGeCK results on replicates 1 and 2 of the KBM7 dataset. Middle panel: overlap of PBNPA results on the four replicates of Toxoplasma. Bottom panel: overlap of MAGeCK results on the four replicates of Toxoplasma.
1.5.2. FDR control

Control of FDR is also studied by comparing control vs control or treatment vs treatment read counts between replicates, as no genes should be identified in this comparison. For the KBM7 dataset, we analyzed controls vs controls or treatment vs treatment with the two algorithms and found that PBNPA has fewer falsely identified genes compared with MAGeCK, as shown in Table 1.1. For the Toxoplasma dataset, the results also indicated PBNPA had fewer falsely identified genes than MAGeCK which is showed in Table 1.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>KBM7</th>
<th>Toxoplasma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>direction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Ctrl1 vs</td>
<td>Trt1 vs</td>
</tr>
<tr>
<td></td>
<td>ctrl2</td>
<td>trt2</td>
</tr>
<tr>
<td>Positive</td>
<td>MAGeCK</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PBNPA</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Negative</td>
<td>MAGeCK</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>PBNPA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1.1. Comparison of FDR control between MAGeCK and PBNPA

1.6. Discussion

The similarities and differences in performance of the two algorithms, MAGeCK and PBNPA, on the two real datasets can be explained below. In the KBM7 dataset, each gene is targeted by 10 sgRNAs. From our simulation study, 10 sgRNAs per gene should be sufficient to give reliable inference on the hits. Thus, these two algorithms give highly similar results. For the Toxoplasma dataset, although there are 10 sgRNAs designed for each gene, the algorithm used to design sgRNAs is optimized for human genes not for Toxoplasma, which, we conjecture, would deteriorate the efficiency of sgRNAs in the screen. In addition, the screening pipeline for Toxoplasma differs from that for cultured human cells, which may induce unknown variability in the data. Based on the above rationale, we conclude that PBNPA is more robust to data variability than MAGeCK. We also note that the other two
methods (RSA and ScreenBEAM) did not perform well on these real data, which agrees with our findings from simulation. In particular, RSA showed poor performance in controlling FDR; for example, in the KBM7 dataset, when we compared ctrl1 vs. ctrl2, RSA claimed more than 90% of the genes are significant when controlling FDR at 5% for positive selection. This is also consistent with an observation in the MAGeCK paper (Li et al., 2014) that RSA has a high FDR.

While researchers typically use gene-specific null distributions in their permutation procedures, we employed a common null probability distribution for all genes in PBNPA. We find that this gives similar or even slightly better performance than using gene-specific null distributions. However, building a common null distribution for all genes substantially saves computation time over building gene specific null distributions. For example: if there are 10,000 genes and we permute 10 times, we can get a common null distribution for all genes based on 10000×10=100,000 replicates; but we need to permute 100,000 times if we want an individual null distribution for each gene based on the same number of replicates. Here, using a common null distribution saves 10,000 times as much computation time as using gene-specific null distributions.

Although our algorithm is designed to analyze CRISPR data, it can also be applied to analyze genetic screens implemented with siRNAs or shRNAs and drug screens, which all generate data with structures similar to those in CRISPR screens. The idea of doing permutation twice, with significant genes from the first round removed to get a more accurate null distribution, could be used by other studies where p values are mainly generated from a permutation process. We note that there are supervised methods of analyzing CRISPR data, which need previous knowledge to estimate the background noise in the platform and variability in the data (Hart and Moffat, 2016). Such methods are suitable in situations when reliable previous screening results are available.

To the best of our knowledge, our paper is the first study to compare the performance of several algorithms with simulated datasets. With the known ground truth, we showed the overall superiority of our PBNPA algorithm compared to several existing methods in
analyzing CRISPR data, which is also verified by the real data studies. The behaviors of each algorithm are revealed from simulation studies, which could help researchers select the most appropriate algorithm to analyze CRISPR data.

Although there are many existing algorithms available for analyzing CRISPR data, researchers are particularly interested in new algorithms that can give consistent and reliable results with a small number of sgRNAs per gene and a low sequencing depth and that are not sensitive to platforms, which will facilitate genome-scale screens while lowering the cost. Our PBNPA algorithm is a step toward achieving this goal.

1.7. R package

We created an R package to implement PBNPA. This package is available at at CRAN: https://cran.r-project.org/web/packages/PBNPA/index.html.

The main function in the package named 'PBNPA'. This function uses the raw read count data for CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) screens and conducts statistical analysis for permutation based non-parametric analysis of CRISPR screen data. This function can also be used to analyze data from other types of functional genomics screens such as siRNA screen or shRNA screen. Drug screens or microarray expression data, if they have structures similar to what this algorithm is designed for, can also be analyzed with this function as the algorithm has no specific distributional assumptions for the data and \( p \)-values are calculated from a permutation based procedure. It can handle data with multiple replicates. After executing the function, a list of 5 elements will be returned. The first element is \( pos.gene \), which is the index of genes identified as hits for positive screen by controlling FDR at the selected level; the second element is \( pos.number \), which is the number of genes identified as hits for positive screening; The third element is \( neg.gene \), which is the index of genes identified as hits for negative screen by controlling FDR at the selected level; the fourth element is \( neg.number \), which is the number of genes identified as hits for negative screening; the fifth element is a dataframe which contains un-
adjusted $p$-values and FDR adjusted $p$-values for all the genes (for both negative selection and positive selection).
2.1. Introduction

Formalin-fixed paraffin-embedded (FFPE) tissue samples are usually collected for diagnostic purposes in clinical routines (Lüder Ripoli et al., 2016). Unlike freshly frozen (FF) tissue samples that must be frozen instantly after collection and then stored in freezers, FFPE samples can be stored at room temperature and kept for a long time. Due to the ease of handling and inexpensive storage (Perlmutter et al., 2004), numerous FFPE tissue samples have been deposited into tissue banks and pathology laboratories around the world, and are readily available (Lüder Ripoli et al., 2016; Reis et al., 2011). Such samples are often accompanied by well documented patient information, disease status and long-term clinical follow up information. Further, there exist vast archives of specimens from which only FFPE, but no FF, samples can be obtained (e.g., specimens of a deceased patient). Thus, the ubiquity of FFPE samples has made them a highly valuable resource in biomedical studies. In particular, FFPE samples have great potential for biomarker discovery, which can be critical for disease diagnosis, prognosis and treatment plan selection (Ludwig and Weinstein, 2005; Rosenfeld et al., 2008; Xie et al., 2011).

Despite advantages of FFPE samples, the formalin fixation process breaks RNA into small pieces with an average size of $\sim 200$nt and irreversible methylene crosslinks between RNAs and proteins may form that affect enzyme based downstream reactions (Masuda et al., 1999). The low quality of RNA from FFPE samples hinders reproducibility and sensitivity of assays for quantitatively measuring gene expression levels via microarray experiments or real time
polymerase chain reaction (qPCR) which involves enzyme-mediated reverse transcription from mRNA to cDNA (Von Ahlfen et al., 2007). Thus, in order to exploit the vast collection of FFPE samples, robust assays are needed to enable and improve expression profiling in these samples.

In recent years, several methods/platforms have been developed for gene expression profiling in FFPE samples either at the genome-wide scale or for a subset of genes. April et al. developed a whole genome cDNA-mediated annealing, extension, selection, and ligation (WG-DASL) assay to perform gene expression profiling in FFPE samples (April et al., 2009). Iddawela et al. reported that WG-DASL assays could reliably probe gene expression levels in breast cancer FFPE samples (Iddawela et al., 2016). Abdueva et al. showed that Affymetrix microarrays could be used to probe gene expression signatures and perform differential expression analysis with FFPE samples and obtained results comparable to those from unfixed tissues (Abdueva et al., 2010). Thompson et al. developed the HTG EdgeSeq chemistry platform that uses RNA extraction-free nuclease protection assay (qNPA), followed by the quantification of RNA molecules by next generation sequencing techniques such as RNA-seq, to profile microRNA and RNA in FFPE samples (Thompson et al., 2014).

Unlike whole genome expression profiling above, Paluch et al. developed targeted RNAseq that can selectively examine the abundance of immune related genes on archival FFPE samples (Paluch et al., 2017). Usually, platforms for measuring expression levels of a subset of genes only are called medium-throughput platforms. Compared to the high-throughput (genome-wide) platforms, they often have better technical reproducibility and are more readily to use in clinical settings. For medium-throughput platforms, besides the probes for detecting genes of interest, there are usually probes designed for internal control, for example, negative controls, positive controls and housekeeping genes. Negative controls target no known sequence and should have zero count ideally; positive controls added to the reaction system have known amounts of RNA targets; and housekeeping genes maintain basic cell functions, with expression levels that minimally fluctuate across different individuals compared with other genes (Waggott et al., 2012). These internal controls can provide infor-
mation for adjusting for unwanted biological and technical effects that can mask the signal of interest.

Among the medium-throughput platforms, the NanoString nCounter is the most popular (Geiss et al., 2008). It is highly multiplexed – it can effectively detect up to 800 genes in a single tube in one run, which bridges the gap between genome-wide expression profiling by microarray or RNAseq and targeted profiling by qPCR (Kulkarni, 2011). More importantly, the nCounter platform is a Clinical Laboratory Improvement Amendments (CLIA) certifiable assay (for Medicare & Medicaid Services et al., 2005), which could be translated into clinical settings.

Due to its importance in medium-throughput profiling, several analysis methods including NanoStringNorm, NAPPA and NanoStringDiff have been developed for the NanoString nCounter platform to normalize and extract gene expression levels from different samples. These algorithms are mainly focused on removing noise from each of the following three sources with the use of one specific type of internal controls: (1) lane-by-lane noise, which results from variation in experimental conditions (such as humidity, temperature, etc.) between reaction systems, is estimated and removed using information from positive controls; (2) background noise, introduced by non-specific binding of the probes, is estimated and removed using negative controls; and (3) variation in sample loading amounts or difference in RNA degradation levels is evaluated using housekeeping genes (Waggott et al., 2012; Wang et al., 2016; Harbron and Wappett, 2015).

To be specific, NanoStringNorm is an R package that implements a normalization protocol recommended by the manufacturer’s guideline (Waggott et al., 2012). First, the lane-by-lane variation is removed by scaling the samples with a factor that makes summary statistics of positive control counts (e.g., mean, median, or geometric mean) equal across samples. Then background correction is performed by subtracting the read count with a statistic representing the background noise, for example, the mean or maximum count of negative controls. Finally, the loading variation is adjusted by a factor calculated from housekeeping genes in the same way as in the first step. It is obvious that NanoStringNorm performs normalization
in an ad hoc way without any rigorous statistical model involved. NAPPA is perhaps the most commonly used algorithm by researchers to normalize NanoString data (Harbron and Wappett, 2015), to the best of our knowledge. This algorithm adjusts the background noise with a truncated Poisson distribution and corrects the loading variation by fitting a sigmoidal curve while normalizing the lane-by-lane variation similarly as in NanoStringNorm. NanoStringDiff is originally designed for identifying differentially expressed genes based on the NanoString nCounter platform, but can be easily adapted for the purpose of normalization (Wang et al., 2016). NanoStringDiff fits a generalized linear model to the data, from which three factors are extracted from positive controls, negative controls and housekeeping genes to adjust for lane-by-lane variation, background noise and variation in the amount of input sample, respectively.

Although the three methods are designed for or can be used to normalize NanoString nCounter data, no meticulous research has been conducted to study the characteristics of this type of data from FFPE samples; and no simulation studies were carried out to evaluate their performance in normalizing such FFPE data. In addition, information provided by different types of internal controls is intermingled. For example, although positive controls are designed to measure the noise from varying experimental conditions, read counts from negative controls can also provide useful information about this type of noise. The current normalization methods ignore this fact and cannot make the best use of data. In addition, all current algorithms use housekeeping genes by assuming that their expression levels are constant between different samples or individuals. But this may be fallacious – biologists generally define housekeeping genes as those that do not vary much between different tissues of an individual, but they have not evaluated the stability of their expression levels from different individuals (Eisenberg and Levanon, 2013). Thus, advanced statistical modeling based on an integrated understanding of the nCounter system without restrictive model assumptions is needed to boost its application in clinical and academic research.
We begin by exploring key features of the NanoString nCounter data from FFPE samples in Section 2.2. In Section 2.2.3, we construct an integrated system of random-coefficient hierarchical regression models for modeling read counts from the different types of probes in the nCounter system. Section 2.3 describes our computational strategy based on a Bayesian approach. We label the proposed method by RCRnorm, where “RCR” stands for random-coefficient hierarchical regression and “norm” stands for normalization. Section 2.4 presents a formal simulation study, conducted to evaluate the performance of RCRnorm in comparison with the three existing methods (i.e., NanoStringNorm, NAPPA and NanoStringDiff), as well as examine its robustness to deviations from key model assumptions. Section 2.5 provides real data applications to illustrate the proposed Bayesian approach. Section 2.6 concludes the paper with a brief summary and some in-depth discussion.

2.2. Motivating example

2.2.1. Data description

The data that motivate our research are from a published study (Xie et al., 2017), which aims to validate a 12-gene signature for predicting adjuvant chemotherapy (ACT) response in lung cancer. A gene signature is a subset of genes, selected from all human genes (more than 20,000), which can be used for diagnosis or prognosis of diseases such as cancer (Ziober et al., 2006; Chen et al., 2007). Typically, a gene signature is identified via variable/model selection techniques, with each gene’s expression measurement corresponding to a variable.

The 12-gene signature was developed from FF samples to predict, among lung cancer patients, who would benefit from ACT so that patients that are unlikely to benefit from ACT can avoid adverse effects of unnecessary treatment (Tang et al., 2013). As mentioned in the introduction, FFPE samples are widespread. FF samples, however, are not readily available for clinical applications, due to reasons including (i) easy contamination by pathogenic germs, (ii) rapid deterioration in room temperature, and (iii) much higher storage cost for frozen specimens than room temperature specimens (Stefan et al., 2010). Thus, it is important
to validate the performance of the signature on FFPE samples so that a clinical applicable
assay can be developed based on the nCounter platform (Xie et al., 2017).

The dataset used by Xie et al. (Xie et al., 2017) contains gene expression levels measured
by the nCounter platform on paired FF and FFPE samples from 30 patients. The goal
in their study is to verify that each gene’s expression levels of the 30 patients from FFPE
samples are well correlated with those from paired FF samples so that the statistical model
based on the 12-gene signature derived from FF samples can be applied to FFPE samples
as well. Although this signature only contains 12 genes, 87 genes in total were measured in
the dataset.

Table B.1 in appendix shows the data structure derived by combining raw data files for
different patient samples, where each row represents a probe, and each column except for
the first two represents a sample. The 1st column labeled “CodeClass” indicates the probe
type: negative controls, positive controls, housekeeping and regular genes. The 2nd column
contains unique probe names. Generally, there are six positive controls (i.e., $P = 6$) in
the code set, but the number of negative controls $N$ and the number of housekeeping genes
$H$ can vary. The name of each negative or positive control contains a pair of parentheses,
within which there is a number indicating the concentration amount of RNA added to the
system that is targeted by that control. For the six positive controls, the RNA amount is
0.125, 0.5, 2, 8, 32, 128 fM, respectively, while for all negative controls, it is zero since there
is no known RNA transcript that can be targeted by the probes. All the other columns in
Table B.1 contain (transformed) read counts from individual samples. As will be detailed
in Section 2.2.3, each (transformed) count is denoted by $Y$, with a superscript representing
the code-class affiliation, the 1st subscript denoting the patient ID and the 2nd subscript
denoting the probe ID in that code class.

In the study (Xie et al., 2017), the (paired) data involve two tables in the form of
Table B.1, one for FF samples and the other for FFPE samples from the same set of patients.
There are 8 negative controls, 7 housekeeping genes and 87 regular genes besides 6 positive
controls in the data.
Data generated by the nCounter system have to be normalized, to account for sample preparation variation, sample content variation, and background noise, etc., before they can be used to quantify gene expression and conduct any downstream statistical analysis. Here, the availability of data from FFPE samples would allow us to explore major characteristics of such data and examine key assumptions/hypotheses about the mean structure of the data, when developing a new normalization method that aims to improve existing ones. Meanwhile, the availability of data from paired FF samples would enable us to quantitatively assess and compare the performance of any normalization methods developed for the nCounter system. Due to the lack of ground truth, it is generally difficult to compare the performance of different normalization methods on real data. Nevertheless, the data from FF samples, once available, can be used to provide a surrogate of the truth. This is because FF tissues are known to maintain RNA very well (much lower degradation of RNA and no methylene crosslink between RNA and proteins) and thus are considered as a gold standard for most molecular assays (Solassol et al., 2011).

2.2.2. Exploratory analysis

To ensure that data resulting from an nCounter gene expression experiment is of adequate quality to be used in subsequent analysis, it is necessary to apply quality control procedures according to the NanoString guidelines. Among the 30 patients’ FFPE samples with 87 regular genes, two patients and four genes were removed for their compromised data quality because they have mean read counts lower than the maximum count of negative controls. An interesting fact is that the two samples discarded are the oldest among the 30 FFPE samples and were collected before the year of 2000. This supports the notion that storage time is a key factor that influences RNA quality from FFPE samples (Von Ahlfen et al., 2007).
Figure B.1 in appendix plots raw read counts of six positive control probes vs. patient index for FFPE samples. It shows that on the un-transformed scale, the high count probes have high variance. This is a general property of count data generated from sampling distributions, whose variance typically increases with the mean. Thus, we apply the commonly used log transformation to the raw counts; and to avoid $-\infty$ arising from zero counts, we add 1 to the observed counts before applying the logarithm.

The empirical distributions of the log10 transformed gene read count of FFPE vs FF samples are showed in Figure 2.1(a) and (b), respectively, in which each (density) curve corresponds to a patient sample and is plotted using log read counts of housekeeping and regular genes. It is obvious that the locations of the distributions of FFPE samples vary more dramatically than those of FF samples. This indicates the existence of heterogeneity in RNA degradation and fragmentation levels among the 28 FFPE tissue samples, contributing to individual sample effects in transcript abundance. This should be modeled, whenever possible, to enable comparison of gene expression levels between patients after removal of such technical artifacts.

Figure 2.1(c) plots log read counts of six positive control probes vs. patient index for FFPE samples. Compared with Figure B.1, we can see that the log transformation greatly stabilizes the count variance. Another noticeable observation is that the zig-zag patterns for the six probes are so similar, strongly indicating the existence of the lane effects.

Given a sample $i$, one would expect that the log transformed read count (say $Y_{ij}$) of any probe $j$ has a monotonically increasing relationship with the corresponding RNA amount (say $R_{ij}$). Using positive controls whose RNA amounts are known and fixed over all $i$ (i.e., $R_{ij} \equiv R_j$ and $R_j$s are known), we can compute the correlation between $R_{ij}$ and $Y_{ij}$ and that between $X_{ij}$ and $Y_{ij}$ for each patient $i$, where $X_{ij} \equiv \log R_{ij}$. Figure 2.1(d) shows two boxplots based on FFPE samples, one for the 28 correlations using log RNA amounts and the other for those using original RNA amounts. We can see that the correlations using log RNA amounts are much higher, with values very close to 1. Thus, a linear relationship between the log RNA amount $X_{ij}$ and log read count $Y_{ij}$ seems to capture the underlying
Figure 2.1. Exploratory analysis of lung cancer data from xie et al. (Xie et al., 2017). Panels (a) and (b) show empirical distributions of log read counts based on housekeeping and regular genes for the 28 FFPE and FF samples, respectively. For FFPE samples, panel (c) plots log read counts of six positive controls (with different known RNA concentration amounts) vs. patient index; (d) compares the boxplot of correlations between log RNA amount and log read count with the boxplot of correlations between RNA amount and log read count; (e) shows empirical densities of patient-wise intercepts and slopes, and (f) overlays the 28 patient-wise fitted lines of log read count vs log RNA amount, all estimated using data from positive controls; and panel (g) shows boxplots of residuals for the eight negative and six positive controls from fitting the linear trend (2.1) per patient, where each boxplot contains residuals from 28 patients for a control probe.
pattern well. More precisely, for each sample \( i \), this can be described by

\[
E(Y_{ij}|X_{ij} = x, a_i, b_i) = a_i + b_i x, \tag{2.1}
\]

where \( a_i \) and \( b_i \) are sample-specific regression coefficients.

Figure 2.1(e) shows the empirical densities of \( a_i \)'s and \( b_i \)'s; and 2.1(f) shows the linear trend (2.1) for each patient, all estimated using FFPE data from positive controls. The straight lines in Figure 2.1(f) are similar but apparently do not overlap. This suggests that the simplifying assumption \( a_i \equiv a \) (or \( b_i \equiv b \)) is not appropriate; but \( a_i \)'s (or \( b_i \)'s) share some commonality and so may come from the same distribution. From Figure 2.1(e), we can see that the two distributions are well apart with different spreads; and the Shapiro-Wilk test (Shapiro and Wilk, 1965) suggests no gross departure from normality at the significance level 0.05 for either distribution. Thus, it is plausible to assume that \( a_i \)'s and \( b_i \)'s are random and follow two separate normal distributions.

For every housekeeping or regular gene, the RNA amount \( R_{ij} \) reflects gene \( j \)'s expression abundance in sample \( i \), whose value is unknown. But for negative controls, \( R_{ij} \equiv 0 \) so that \( X_{ij} = -\infty \), which is ill defined. To solve this issue, we add a small positive number \( \delta \) so that \( X_{ij} = \log \delta \) instead and (2.1) holds for negative controls as well. Both \( \delta \) and \( R_{ij} \)'s are estimable. The intuition is that with the information from positive controls, we can pin down \( (a_i, b_i) \) for each sample so that with observed counts from negative controls, we can estimate \( \delta \), and with observed counts from housekeeping or regular genes, we can estimate \( R_{ij} \)'s.

We use \( \sum_{i=1}^{\sum_{j \in J^-}} (Y_{ij} - \hat{a}_i)/\hat{b}_i \) to obtain a rough estimate of \( \log \delta \) for FFPE samples, where \( J^- \) denotes the index set of negative controls, and \( \hat{a}_i \) and \( \hat{b}_i \) are estimated using data from positive controls as before. We then compute the residuals, i.e. deviations from the linear pattern (2.1), for each positive and negative control, and their boxplots are shown in Figure 2.1(g). Two interesting observations can be made here, which will be useful for the model construction in Section 2.2.3. First, negative controls tend to have much larger
deviations than positive controls; and their distributions tend to have much larger variability (hence wider spreads). Second, for each individual probe, the residuals are not randomly distributed around zero: all the boxplots for positive controls are entirely above/below zero, and most boxplots for negative controls have 75% residuals or more above/below zero, indicating residuals are clustered by probes.

2.2.3. Proposed data model based on RCR

Let \( i \) index (FFPE) patient samples, \( p \) index positive controls, \( n \) index negative controls, \( h \) index housekeeping genes, and \( r \) index regular genes, for \( i = 1, \ldots, I \), \( p = 1, \ldots, P \), \( n = 1, \ldots, N \), \( h = 1, \ldots, H \), and \( r = 1, \ldots, R \), where \( I \) is the number of patients, \( P \), \( N \), \( H \) and \( R \) are the (prespecified) number of positive controls, negative controls, housekeeping genes and regular genes in the NanoString nCounter platform, respectively.

Motivated by the analysis in Section 2.2, we set up a system of (hierarchical) linear regression models with random coefficients for the four different types of probes, in which the general linear relationship between the observed log read count and log RNA amount (either known or unknown) is assumed regardless of the probe type; and except for the observed log read counts, all the random components of the system are assumed to be independent. We begin with the model for the positive control class, given below:

\[
Y_{ip}^+ = a_i + b_i X_p^+ + d_p^+ + e_{ip}^+.
\]

where \( Y_{ip}^+ \) is the logarithm of read count plus 1 of the \( p \)th positive control from the \( i \)th sample, \( X_p^+ \) represents the logarithm of the known RNA input amount (unit: fM) in the reaction system, and the superscript ‘+’ indicates the membership of the positive control class. The \( a_i \) and \( b_i \) are the sample-specific random intercept and slope which may reflect the lane-by-lane variation. According to Figure 2.1(e)-(f), we may assume \( a_i \)s and \( b_i \)s be independent and identically distributed normal variables, respectively: \( a_i \overset{iid}{\sim} N(\mu_a, \sigma_a^2) \) and \( b_i \overset{iid}{\sim} N(\mu_b, \sigma_b^2) \). Further, \( d_p^+ \) represents the probe-specific systematic deviation from the linear
pattern (2.1) (see Figure 2.1(g)) and we assume \( d_p^+ \sim N(0, \sigma_d^2) \). Finally, \( e_{ip}^+ \sim N(0, \sigma_e^2) \) is the random error term, which reflects the remaining variability of the log observed count after taking into account the linear trend and the probe-specific deviation.

For the negative control class, the model is given by

\[
Y_{in}^- = a_i + b_i c + d_n^- + e_{in}^-,
\]

(2.3)

where \( Y_{in}^- \) is the logarithm of read count plus 1 of the \( n \)th negative control from the \( i \)th sample, \( c \equiv \log \delta \) is an unknown constant, the superscript '-' indicates the membership of the negative control class, and the other terms are defined similarly as in (2.2). As shown in Figure 2.1(g), the distributions of deviations (from the main linear pattern) for positive controls are very different from those for negative controls: from the centers (i.e., middle horizontal bars) of the boxplots, we can see \( d_n^- \)s vary more than \( d_p^+ \)s; and from the widths of the boxplots, we can see \( e_{in}^- \)s vary much more than \( e_{ip}^+ \)s. Thus, we have to assume \( d_n^- \sim N(0, \sigma_{d-}^2) \) and \( e_{in}^- \sim N(0, \sigma_{e-}^2) \), where the data suggest that \( \sigma_{d-}^2 > \sigma_d^2 \) and \( \sigma_{e-}^2 > \sigma_e^2 \).

For the housekeeping gene class, the model is given by

\[
Y_{ih}^* = a_i + b_i X_{ih}^* + d_h^* + e_{ih}^*,
\]

(2.4)

where \( X_{ih}^* \) is the unknown log RNA amount of the \( h \)th housekeeping gene from sample \( i \), the superscript '*' indicates the membership of the housekeeping gene class, and the other terms are defined similarly as before. Unlike positive or negative controls, \( X_{ih}^* \) in (2.4) is random by nature rather than being constant, which can be decomposed into a random term \( \kappa_{ih}^* \) and a fixed term \( \phi_i \), i.e., \( X_{ih}^* = \phi_i + \kappa_{ih}^* \). Here, \( \phi_i \) is a constant that reflects the individual effect of sample \( i \) in transcript abundance (e.g., patient-to-patient variation, variation in RNA degradation and fragmentation levels of FFPE tissues, variation in the amount of input sample material, etc.), satisfying \( \sum_{i=1}^I \phi_i = 0 \); and \( \kappa_{ih}^* \sim N(\lambda_h^*, \sigma_{\kappa^*}^2) \) reflecting the remaining expression abundance after adjusting for the sample effect. Note that \( E(\bar{X}_{ih}^*) = \lambda_h^* \), where \( \bar{X}_{ih}^* = \sum_{i=1}^I X_{ih}^*/I \) and \( \lambda_h^* \) is the gene-specific mean of the log RNA amount. Here, individual
sample effects $\phi_i$s and gene effects $\lambda_h^*$s are both modeled as fixed effects instead of random effects. This is because for a specific sample, we are interested in recovering $\kappa_{ih}^*$ from $X_{ih}^*$, rather than inferring the marginal distributions of $\phi_i$s and $\lambda_h^*$s.

For the regular gene class, the RNA amounts in different samples are unknown, too. So the model is set to be the same as that for the housekeeping gene class, but with a difference probe index $r$ and no superscript (for notational brevity):

$$Y_{ir} = a_i + b_i X_{ir} + d_r + e_{ir}, \quad (2.5)$$

where $X_{ir} = \phi_i + \kappa_{ir}$ is the unknown log RNA amount of the $r$th regular gene from sample $i$, and the definitions of $\kappa_{ir}$, $d_r$ and $e_{ir}$ are self-evident. Correspondingly, we assume $\kappa_{ir} \sim N(\lambda_r, \sigma^2_{\kappa})$. Note that two separate variances, $\sigma^2_{\kappa}$ and $\sigma^2_{\kappa}$, are needed for the housekeeping and regular genes, respectively. This is because expression levels of housekeeping genes are known to be more stable across samples, and so one would expect $\sigma^2_{\kappa} < \sigma^2_{\kappa}$.

In the reaction system of the nCounter platform, negative controls have no known target and all detected binding signals should be from non-specific binding while positive controls, housekeeping and regular genes all have known targets, and so their working mechanisms may be similar. Thus, we assume $d^*_p, d^*_h, d_r \sim N(0, \sigma^2_d)$ and $e^*_ip, e^*_ih, e_{ir} \sim N(0, \sigma^2_e)$. We comment that for the housekeeping and regular genes, (2.4) and (2.5) are both hierarchical: the bottom layer involves a linear regression model with random coefficients, and the second layer (for the unknown log RNA amount) involves a two-way ANOVA model, where one factor represents the sample-specific effect $\phi_i$ and the other factor represents the gene-specific effect that are related to $\lambda_h^*$ or $\lambda_r$. In addition, for all the four classes, since gene-specific deviations from the main linear trend (2.1) are allowed through $d^*_p, d^*_n, d^*_h$ and $d_r$, the log read counts of the same gene from different samples (e.g., $Y_{ir}$ and $Y_{i'r}$) are correlated; meanwhile, the log read counts of the different genes from the same patient (e.g., $Y_{ir}$ and $Y_{i'r'}$) are correlated, too, as they share the same random intercept $a_i$ and slope $b_i$. 

35
2.3. Bayesian approach

2.3.1. Full probability model

Based on the system of equations (2.2)-(2.5), the parameters of our data model include \( \mu_a, \sigma^2_a, \mu_b, \sigma^2_b, \sigma^2_d, \sigma^2_e, \sigma^2_{d-}, \sigma^2_{e-}, \{\lambda_i\}_{i=1}^{I-1}, \{\lambda^*_h\}_{h=1}^{H}, \sigma^2_{\kappa^*}, \{\lambda_r\}_{r=1}^{R}, \sigma^2_{\kappa}, \) and \( c, \) among which \( \mu_a, \mu_b, c, \phi_is, \lambda_r, \) and \( \lambda^*_h \) are location parameters, and all the others are variance parameters.

We assume all these parameters are a priori independent. Let \( Y \) denote all the log read count data observed; and let \( \Theta \) denote the collection of all latent random variables involved, including \( \{a_i\}_{i=1}^{I}, \{b_i\}_{i=1}^{I}, \{d^+_p\}_{p=1}^{P}, \{d^-_n\}_{n=1}^{N}, \{d^+_h\}_{h=1}^{H}, \{d_r\}_{r=1}^{R}, \{\kappa^*_h\}_{h=1}^{H} \) and \( \{\kappa_{ir}\}_{r=1}^{R} \}_{i=1}^{I} \), and all model parameters. Further, we use \( N(x|\mu,\sigma^2) \) to denote a normal distribution with mean \( \mu \) and variance \( \sigma^2 \), and \( \pi(\cdot) \) to denote a general prior distribution.

Then the full probability model is given by

\[
p(Y, \Theta) \propto \prod_{i=1}^{I} \left( \prod_{p=1}^{P} N(Y_{ip}^+|a_i + b_i X_p^+ + d^+_p, \sigma^2_e) \cdot \prod_{n=1}^{N} N(Y_{in}^-|a_i + b_i c + d^-_n, \sigma^2_{e-}) \right)
\]

\[
\cdot \prod_{h=1}^{H} N(Y_{ih}|a_i + b_i(\phi_i + \kappa^*_h) + d^+_h, \sigma^2_e) \cdot \prod_{r=1}^{R} N(Y_{ir}|a_i + b_i(\phi_i + \kappa_{ir}) + d_r, \sigma^2_e)
\]

\[
\cdot \prod_{h=1}^{H} N(\kappa^*_h|\lambda^*_h, \sigma^2_{\kappa^*}) \cdot \prod_{r=1}^{R} N(\kappa_{ir}|\lambda_r, \sigma^2_{\kappa}) \cdot N(a_i|\mu_a, \sigma^2_a) \cdot N(b_i|\mu_b, \sigma^2_b)
\]

\[
\cdot \prod_{p=1}^{P} N(d^+_p|0, \sigma^2_d) \cdot \prod_{n=1}^{N} N(d^-_n|0, \sigma^2_{d-}) \cdot \prod_{h=1}^{H} N(d^+_h|0, \sigma^2_d) \cdot \prod_{r=1}^{R} N(d_r|0, \sigma^2_d)
\]

\[
\cdot \pi(\mu_a) \cdot \pi(\mu_b) \cdot \pi(c) \cdot \prod_{i=1}^{I} \pi(\phi_i) \cdot \prod_{h=1}^{H} \pi(\lambda^*_h) \cdot \prod_{r=1}^{R} \pi(\lambda_r)
\]

\[
\cdot \pi(\sigma^2_a) \cdot \pi(\sigma^2_b) \cdot \pi(\sigma^2_d) \cdot \pi(\sigma^2_e) \cdot \pi(\sigma^2_{d-}) \cdot \pi(\sigma^2_{e-}) \cdot \pi(\sigma^2_{\kappa}) \cdot \pi(\sigma^2_{\kappa^*})
\]

where the first four lines represent the joint likelihood of the observed data and latent random variables, the fifth line represents prior distributions of the location parameters, and the last line represents prior distributions of the variance parameters.
2.3.2. Prior specification

For each variance parameter involved, we specify an inverse gamma prior distribution $IG(u, v)$, where $u$ and $v$ are small positive numbers to make the prior very vague and diffuse (e.g., $u = v = 0.01$). The purpose of doing so is to let the data speak for itself when sampling the variances from the joint posterior distribution.

For $\mu_a$ and $\mu_b$ (mean of random intercepts $a_i$s and mean of random slopes $b_i$s), we consider normal priors: $\mu_a \sim N(\hat{\mu}_a, m \times se(\hat{\mu}_a))$ and $\mu_b \sim N(\hat{\mu}_b, m \times se(\hat{\mu}_b))$. Here, $m$ is a prespecified constant (e.g., 3, 5) to make the prior much more diffuse than what data suggest; $\hat{\mu}_a$ and $\hat{\mu}_b$ are crude estimates of $\mu_a$ and $\mu_b$; and $se(\hat{\mu}_a)$ and $se(\hat{\mu}_b)$ are their standard errors. We simply set $\hat{\mu}_a = \frac{\sum_{i=1}^{I} \hat{a}_i}{I}$ and $\hat{\mu}_b = \frac{\sum_{i=1}^{I} \hat{b}_i}{I}$, where $\hat{a}_i$ and $\hat{b}_i$ are the (least square) estimated intercept and slope, respectively, from fitting $Y_{ip}^+$ vs. $X_{ip}^+$ for each patient $i$; and the standard errors can be estimated using jackknife resampling that removes two patient samples at a time (Efron and Stein, 1981).

For any other location parameter (say $\theta$), we use a noninformative uniform distribution, $\theta \sim \text{Uniform}(L_\theta, U_\theta)$, which should provide a sufficiently wide coverage for all plausible values of $\theta$ suggested by data. For the added small value $\delta$ associated with negative controls, we consider the range $(10^{-6}, 10^{-1})$ so that $c \sim \text{Uniform}(-6, -1)$ is specified a priori. For the gene-specific mean of the log RNA amount $\lambda_r$, the lower and upper bounds can be specified by $\text{mean}(\hat{X}_r) \pm m \times sd(\hat{X}_r)$, where $\hat{X}_r = (\hat{X}_{1r}, \cdots, \hat{X}_{Ir})$, and $\hat{X}_{ir}$ is a crude estimate of the log RNA amount of regular gene $r$ from sample $i$, e.g. $\hat{X}_{ir} = (Y_{ir} - \hat{a}_i)/\hat{b}_i$. The bounds for $\lambda^*_r$ can be specified similarly using $\hat{X}^*_ih = (Y^*_{ih} - \hat{a}_i)/\hat{b}_i$ for each housekeeping gene. Note that an alternative method to specify a conservative prior range for any of $\lambda_r$s and $\lambda^*_r$s is to use the maximum and minimum statistics, especially when we anticipate that the posterior distribution can be skewed. For example, the lower and upper bounds of $\lambda_r$ can be specified by $\min\{\hat{X}_r\} - \Delta_r$ and be $\max\{\hat{X}_r\} + \Delta_r$ where $\Delta_r$ is a prespecified constant that leaves some extra safe room for either bound (e.g., setting $\Delta_r = sd(\hat{X}_r)$).

Finally, for the sample effect $\phi_i$, the lower and upper bounds can be specified by $\hat{\phi}_i \pm m \times se(\hat{\phi}_i)$. Here, the rough estimate $\hat{\phi}_i$ and its standard error can be easily obtained
using regular genes by running a standard two-way ANOVA model on $\hat{X} \equiv (\hat{X}_r)_{r=1}^R$ with the constraint that the sum of sample-specific effects and the sum of gene-specific effects are zero. Alternatively, they can be estimated nonparametrically: $\hat{\phi}_i = \hat{X}_i - \hat{X}_\cdot$, where $\hat{X}_i = \frac{1}{R} \sum_{r=1}^R \hat{X}_{ir}$ and $\hat{X}_\cdot = \frac{1}{IR} \sum_{i=1}^I \sum_{r=1}^R \hat{X}_{ir}$; and $\text{se}(\hat{\phi}_i)$ can be roughly estimated using jackknife resampling that removes two housekeeping genes at a time.

2.3.3. Posterior computation and Bayesian inference

We use Markov Chain Monte Carlo (MCMC) to draw random samples from the joint posterior distribution $p(\Theta|Y)$, which is proportional to $p(Y, \Theta)$. Standard diagnostic techniques (Gelman et al., 2014) are used to detect the convergence. One advantage of the proposed method is that the posterior conditionals, as detailed in appendix B.2, are all known distributions, from each of which direct sampling can be done. This property allows us to design an efficient Gibbs sampler, in which all the involved quantities are drawn sequentially and generated readily without using any built-in sampling algorithm (such as Metropolis–Hastings and Acceptance/Rejection algorithms) that can greatly slow down the computation.

For the purpose of gene expression normalization, we are mainly interested in estimating $\kappa_{ir}$s for regular genes. For $i = 1, \cdots, I$ and $r = 1, \cdots, R$, let $\kappa_{ir}^{(t)}$ be the posterior draw of $\kappa_{ir}$ in the $t$th iteration of MCMC after the burn-in period, where $t = 1, \cdots, T$, and $T$ is the total number of iterations. Then we can estimate $\kappa_{ir}$ by $\tilde{\kappa}_{ir} = \frac{\sum_{t=1}^T \kappa_{ir}^{(t)}}{T}$. Similarly, we can obtain a Bayesian estimate of $\kappa_{ih}^*$ for each housekeeping gene.

2.4. Simulation

2.4.1. Settings

We conducted three simulation studies to examine the performance of RCRnorm, and to compare it with the three existing methods that have been proposed to normalize NanoS-
In the first study, five settings, labeled I-1 to I-5, were simulated based on the data model proposed in Section 2.2.3. In our basic setting I-1, data were generated using parameter values estimated from the FFPE samples in Section 2.5.1 for the lung cancer application. Settings I-2 – I-5 were modified from I-1 to mimic different real world scenarios.

1. The probed genes have larger variability in their expression levels. To simulate this situation, $\sigma_\kappa$ and $\sigma_{\kappa^*}$ were increased to 3 times that of the basic setting in I-2. These two parameters control the signal strength, where a larger value indicates more genes with strong signals and so it is easier to recover underlying expression levels.

2. The samples have larger lane-by-lane variation. This scenario mimics a poor control of experimental conditions across different samples or lanes, simulated in I-3 by increasing $\sigma_a$ and $\sigma_b$ to 3 times that of the basic setting.

3. The probe library is poorly designed so that probes have larger variability in their affinity to different gene targets. This scenario was simulated in I-4 by increasing $\sigma_d$ and $\sigma_{d^*}$ to 3 times that of the basic setting.

4. Effects of random errors (unexplained variability) were examined in I-5 by increasing $\sigma_e$ and $\sigma_{e^*}$ to 3 times that of the basic setting.

In practice, the linear trend assumption in (2.2)-(2.5) is often robust, but the normality assumption for probe-specific effects and random errors may not always hold. In our second study, we examined the robustness of RCRnorm when the normality assumption is violated. We simulated another five settings by modifying the basic setting, labeled II-1 to II-5, using a standard Student’s t distribution with three degrees of freedom ($t_3$), which represents a thick-tailed distribution, and a Gamma distribution with shape 2 and rate 1 ($G_{2,1}$), which represents a right-skewed distribution. In setting II-1, $t_3$ was used to generate the probe effects \( \{d_{p}^{+}\}_{p=1}^{P}, \{d_{n}^{-}\}_{n=1}^{N}, \{d_{h}^{*}\}_{h=1}^{H}, \{d_{r}^{*}\}_{r=1}^{R} \); and in II-2, $t_3$ was used to generate the random errors \( \{e_{ip}^{+}\}_{p=1}^{P}, \{e_{in}^{-}\}_{n=1}^{N}, \{e_{ih}^{*}\}_{h=1}^{H}, \{e_{ir}^{*}\}_{r=1}^{R} \). In II-3 and II-4, $G_{2,1}$ was used to generate the
probe effects and random errors, respectively. In II-5, $t_3$ was used to generate the probe effects and $G_{2,1}$ to generate the random errors. Data generated from $t_3$ were then rescaled to have the same variance as in the basic setting I-1; and data generated from $G_{2,1}$ were shifted and rescaled to have mean 0 and the same variance as in I-1. Except for the changes above, everything remains the same as in I-1.

Like the existing three methods, RCRnorm assumes a common sample effect $\phi_i$ for all genes in a given sample $i$ to account for between-sample variations resulted from loading or RNA degradation of different samples. However, RNA degradation rates are different among genes as they are determined by a myriad of factors. The gene-wise RNA degradation from either internal pathways or environmental conditions is technically difficult to measure and cannot be separated from true gene expression levels with current data using any of the four methods. To understand how this uncertainty influences their performance, we designed Study III, where $\delta_{vr}$ and $\delta_{vh}^*$ were generated from $N(0, \sigma_\delta^2)$ and added to $\phi_i$, and everything else is again the same as in I-1. In settings III-1 to III-2, we set $\sigma_\delta$ to 0.1 and 0.4, respectively, to study its potential effect.

Under each setting, 50 datasets were independently simulated, each with 28 patient samples, 6 positive controls, 8 negative controls, 7 housekeeping and 83 regular genes, which are exactly the same as in the real lung cancer data. Since all the read counts were generated in the log10 scale, they were exponentiated and rounded to the nearest integers so that the simulated data can be analyzed by all the four algorithms. For RCRnorm, 8000 iterations were simulated in each MCMC run and the first 5000 were used for burn-in. The existing algorithms were applied using their default settings.

2.4.2. Results

To evaluate the performance on normalization under known truth, we computed gene-wise Spearman correlations for 83 regular genes between normalized data and true expression levels within each simulated dataset, and reported their mean, standard deviation (SD), 25th, 50th and 75th percentiles as the summary statistics for every method. Then under
each setting, boxplots of these summary statistics based on 50 replicates were generated and used to compare the four methods. Note that Spearman correlation was used instead of Pearson correlation for a fair comparison. This is because RCRnorm works on the log10 scale, NAPPA on the log2 scale, and the other algorithms on the original scale; and so Spearman correlation that ignores data values and relies on ranks only should be used.

Figure 2.2 shows boxplots of mean and SD, and Figure B.2 in appendix shows boxplots of the 25th, 50th, and 75th percentiles, of gene-wise correlations for each of the five settings in Simulation I. In the basic setting I-1, RCRnorm is a clear winner; and the existing algorithms have somewhat similar performance, among which NanoStringDiff seems to be a bit worse than the other two because of its generally smaller mean, percentiles and larger SD as well as the existence of outliers. This seems to be true in all the other settings except for I-2, where they follow the order RCRnorm > NAPPA > NanoStringNorm > NanoStringDiff instead. In Setting I-2, the increased signal strength improves the performance of the existing algorithms significantly. But it does not affect RCRnorm much as it already performs well in I-1. From settings I-3 to I-5, the increased variability, regardless of the source, worsens the performance of every algorithm: compared to I-1, the mean and percentiles decrease but the SD increases in general. The increase in probe-level variation (i.e., \( \sigma_d \) and \( \sigma_{d-} \)) in Setting I-4 has the largest negative impact on the performance, followed by the increase in variability of random noise (i.e. \( \sigma_e \) and \( \sigma_{e-} \)) in Setting I-5, and last by the increase in lane-by-lane variation (i.e., \( \sigma_a \) and \( \sigma_b \)) in Setting I-3. Among all, NanoStringDiff is the most sensitive to such changes and has extremely bad outliers, meaning that it fails to work sometimes. A further investigation reveals that when running NanoStringDiff, the generalized linear model did not converge for some datasets. By contrast, RCRnorm is the least affected by such changes, and it maintains strong performance in all settings. Besides being well apart from the other boxplots in all settings except for I-2, its boxplots show the smallest inter quartile ranges, meaning that RCRnorm gives very consistent results over different replicate datasets.

Figure 2.3, along with Figure B.3 in appendix, shows results for the five settings in Simulation II. With these distributional disturbances added to the model system, the performance
Figure 2.2. Simulation study I for mimicking various real-world scenarios: boxplots for mean and SD of gene-wise Spearman correlations between normalized data and true expression levels based on 50 replicates for each of the five settings I1-I5. Compared to the basic setting I-1 (parameter values estimated from the FFPE samples in the lung cancer application), gene expression variability is increased in I-2, lane-by-lane variation is increased in I-3, probe-level variation is increased in I-4, and variability of random noise is increased in I-5. Note that NSnorm stands for NanoStringNorm and NSdiff for NanoStringDiff.
of every algorithm becomes worse, compared with I-1, as indicated by the generally wider boxes and more occurrence of outliers for all the summary statistics. Typically, the changes are not large except for NanoStringDiff in Setting II-2 (i.e., heavy-tailed random errors), where extremely bad outliers occur again. RCRnorm seems to be quite robust to these moderate violations of the normality assumption and still outperforms the other methods.

Figure 2.4, along with Figure B.4 in appendix, shows results for the three settings in Simulation III. With these experimentally unmeasurable gene-specific sample effects added, the performance of all the algorithms becomes worse as expected, when compared with I-1. Although the performance deteriorates with increasing $\sigma_\delta$, RCRnorm maintains the best performance among the four.

### 2.5. Real data applications

#### 2.5.1. Lung cancer data

We use the NanoString nCounter data from FFPE samples described in Section 2.2 to illustrate the proposed RCRnorm first. We ran our MCMC algorithm for 15,000 iterations in total. The convergence for all the model parameters was detected after 7,000 iterations, and we discarded the first 10,000 for burn-in. We then thinned the chain to reduce the autocorrelation among posterior draws by saving every tenth draw only, and so 500 posterior samples were kept. Figure 2.5 shows the posterior densities of global parameters including $\mu_a$, $\mu_b$, $c$, $\sigma_{\kappa^*}$, $\sigma_\kappa$, $\sigma_d$, $\sigma_{d--}$, $\sigma_e$, and $\sigma_{e--}$; and Table 2.1 presents a summary for Bayesian estimates of these parameters including the posterior mean, median, standard error (SE), and a 95% credible interval (CI) using the 2.5th and 97.5th percentiles of (thinned) posterior samples. Here, the posterior mean is used to estimate each location parameter; but the posterior median is used to estimate each variance parameter since the corresponding posterior density is skewed to the right.

Several intriguing observations can be made from the above figure and table. As we know, housekeeping genes are involved in the maintenance of basic cellular function, and
Figure 2.3. Simulation study II for robustness checking: boxplots for mean and SD of gene-wise Spearman correlations between normalized data and true expression levels based on 50 replicates for each of the five settings II1-II5. In II1, a thick-tail distribution $t_3$ was used to simulate probe effects; in II-2, $t_3$ was used to simulate random errors; in II-3, a right-skewed distribution $G_{2,1}$ was used to simulate probe effects; in II-4, $G_{2,1}$ was used to simulate random errors; and in II-5, $t_3$ was used to generate probe effects and $G_{2,1}$ to generate random errors. Except for the changes above, everything remains the same as in the basic setting I-1 including all parameter values (so $t_3$ and $G_{2,1}$ need to be rescaled or shifted). Note that NSnorm stands for NanoStringNorm and NSdiff for NanoStringDiff.
Figure 2.4. Simulation study III for mimicking different RNA degradation levels between genes: boxplots for mean and SD of gene-wise Spearman correlations between normalized data and true expression levels based on 50 replicates for each of the two settings III1-III2, where $\delta_{ir}$ and $\delta_{th}$ were generated from a normal distribution with mean 0 and SD 0.1 and 0.4, respectively. Note that NSnorm stands for NanoStringNorm and NSdiff for NanoStringDiff.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>median</th>
<th>SD</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_a$</td>
<td>2.462</td>
<td>2.462</td>
<td>0.014</td>
<td>(2.436, 2.490)</td>
</tr>
<tr>
<td>$\mu_b$</td>
<td>0.927</td>
<td>0.927</td>
<td>0.003</td>
<td>(0.922, 0.932)</td>
</tr>
<tr>
<td>$c$</td>
<td>-1.877</td>
<td>-1.872</td>
<td>0.125</td>
<td>(-2.126, -1.635)</td>
</tr>
<tr>
<td>$\sigma_{\kappa^*}$</td>
<td>0.138</td>
<td>0.136</td>
<td>0.011</td>
<td>(0.121, 0.163)</td>
</tr>
<tr>
<td>$\sigma_{\kappa}$</td>
<td>0.361</td>
<td>0.361</td>
<td>0.007</td>
<td>(0.349, 0.375)</td>
</tr>
<tr>
<td>$\sigma_d$</td>
<td>0.333</td>
<td>0.308</td>
<td>0.104</td>
<td>(0.198, 0.595)</td>
</tr>
<tr>
<td>$\sigma_d$</td>
<td>0.129</td>
<td>0.120</td>
<td>0.045</td>
<td>(0.074, 0.242)</td>
</tr>
<tr>
<td>$\sigma_{e^-}$</td>
<td>0.211</td>
<td>0.211</td>
<td>0.011</td>
<td>(0.189, 0.233)</td>
</tr>
<tr>
<td>$\sigma_e$</td>
<td>0.035</td>
<td>0.035</td>
<td>0.003</td>
<td>(0.031, 0.041)</td>
</tr>
</tbody>
</table>

Table 2.1. Lung cancer data: posterior summary statistics of global parameters from applying RCRnorm to FFPE samples
Figure 2.5. Lung cancer data: posterior densities of global parameters from applying RCRnorm to FFPE samples
so are expected to be uniformly expressed with low variability in all cells and experimental conditions. Our analysis using RCRnorm confirms that, compared to other genes, expression levels of housekeeping genes indeed vary much less. Clearly, the Bayesian estimate of $\sigma_{k_\star}$ (0.136, the SD of expression levels for housekeeping genes) is much smaller than that of $\sigma_\kappa$ (0.361, the SD of expression levels for regular genes); and Figure 2.5(b) shows their posterior density curves are well separated with the correct order. Further, our exploratory analysis in Section 2.2.2 strongly indicates $\sigma_d < \sigma_{d-}$ and $\sigma_e < \sigma_{e-}$. That is, the SD of the probe-specific deviation (from the linear trend) for negative controls is larger than that for the other types of probes; so does the SD of the random errors. The Bayesian estimates in Table 2.1 confirm the underlying features again (0.120<0.308 and 0.035<0.211) and the posterior densities in Figure 2.5(e) and (f) support them as well. Note that the data model of RCRnorm does not impose such order constraints at all, but the end results from RCRnorm capture these characteristics accurately.

Next, we compare the performance of RCRnorm in normalizing FFPE data with the existing algorithms. As FF samples generally have much better quality than FFPE samples, we used normalized FF data as a gold standard, where each method was applied to normalize both FFPE and FF data, and Pearson correlations between normalized FFPE and FF data were computed to quantify its performance. The summary statistics (mean, SD, 25%, 50% and 75% quantile) of 83 gene-wise correlations are presented in the left panel of Table 2.2, with the best value bolded in each column. Compared to the original data, all algorithms significantly improve the gene-wise correlations; and RCRnorm has the best performance in terms of higher mean and percentiles as well as smaller variability. Note that the existing algorithms have somewhat similar performance, though NAPPA seems to be slightly better than the other two. However, RCRnorm can further offer a sizable gain over their already improved performance.

Although gene-wise correlations are the focus of the original study (Xie et al., 2017) for validating the gene signature, we also report 28 patient-wise correlations in the right panel of Table 2.2. Here, only RCRnorm and NAPPA are able to adjust patient-wise correlations.
### Table 2.2. Lung cancer data: summary statistics of Pearson correlation coefficients between normalized FFPE and FF samples using different algorithms

<table>
<thead>
<tr>
<th></th>
<th>Gene-wise Pearson correlation</th>
<th>Patient-wise Pearson correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>SD</td>
</tr>
<tr>
<td>Original data</td>
<td>0.291</td>
<td>0.278</td>
</tr>
<tr>
<td>RCRnorm</td>
<td>0.550</td>
<td>0.197</td>
</tr>
<tr>
<td>NAPPA</td>
<td>0.488</td>
<td>0.194</td>
</tr>
<tr>
<td>NanoStringDiff</td>
<td>0.487</td>
<td>0.207</td>
</tr>
<tr>
<td>NanoStringNorm</td>
<td>0.489</td>
<td>0.199</td>
</tr>
</tbody>
</table>

This is because NanoStringDiff and NanoStringNorm linearly transform a patient’s data with scale factors calculated from internal controls and thus do not change the patient-wise correlations. By contrast, both RCRnorm and NAPPA achieve some improvement from normalization, though it is not as big as seen in the gene-wise case. This is perhaps because the patient-wise correlations of the original data are already high and so there is not much room left for improvement. Again, RCRnorm seems to be better than NAPPA with higher mean, percentiles and about the same variability.

It is interesting to observe that for the unnormalized data, the patient-wise correlations (mean: 0.806, SD: 0.129) are much higher than the gene-wise correlations (mean: 0.291, SD: 0.278). The high patient-wise correlations indicate the superb performance of NanoString nCounter on expression profiling with FFPE samples. On the other hand, the much lower patient-wise correlations highlight the importance of removing sample-specific effects for downstream statistical analysis.

In this application, the gene-wise correlations achieved by all the four algorithms are generally lower than what we have seen in our simulation. This can be explained by the following reasons. Firstly, due to experimental and technical limitations, gene-wise RNA degradation levels cannot be measured and removed by any of the four algorithms. As shown in Figure 2.4, such variation could cause a large drop in performance. Secondly,
normalized FF expression levels were used to calculate the correlations in the application while true expression levels were used in the simulation.

2.5.2. Colorectal cancer data

Our second application involves a colorectal cancer study (Omolo et al., 2016) that compared five different platforms to identify which platform could faithfully translate the RAS pathway gene signature identified from FF samples into FFPE samples. RAS pathway activation is a risk factor for the failure of EGFR combination therapy in colorectal cancer patients. Thus, it is clinically important to identify a platform that could obtain reliable information from FFPE samples. Among the 5 platforms compared, NanoString nCounter was found to be the best platform to recover gene expression information from FFPE samples.

We applied RCRnorm to the NanoString nCounter data from FFPE samples in this study, where, again, 15,000 iterations were used with the first 10,000 being burn in. The dataset contains 54 samples, with 6 positive controls (with the same input amounts as before), 6 negative controls, 11 housekeeping and 18 regular genes. Thus, unlike the lung cancer data, it has more samples than probes.

Figure B.5 plots posterior densities and Table B.2 reports posterior summary statistics of global parameters in appendix. Compared with Table 2.1 in the lung cancer study, the estimates of $\mu_a$, $\mu_b$ and $c$ are all close, although they are completely from two independent and distinct studies. Again, the Bayesian estimates of $\sigma_{k^*}$ and $\sigma_\kappa$ (0.190 vs. 0.278) confirm that expression levels of the housekeeping genes vary less than those of the regular genes. Interestingly, $\sigma_{k^*}$ in this study is larger than that in the lung cancer study (estimate: 0.190 > 0.136; 95% CI: (0.176, 0.215) completely above (0.121, 0.163)), although more housekeeping genes were used here (11 vs. 7). This seems to suggest that increasing the number of housekeeping genes used does not necessarily reduce their variability to the minimal level. Our results further confirm that $\sigma_d \ll \sigma_{d^-}$ (0.069 vs. 0.257) and $\sigma_e \ll \sigma_{e^-}$ (0.042 vs. 0.273). Recall that these characteristics were revealed by exploring the lung cancer data. Nevertheless, they may generally hold for NanoString nCounter data of any kind.
In this study, paired FF NanoString data are not available, and so performance comparison among the four methods cannot be done using correlations between normalized FFPE and FF data. According to the manual of NanoStringNorm (Waggott et al., 2012), housekeeping genes are typically selected to be genes with high means and low standard deviations. Thus, the two-sample t test was used to compare the coefficients of variation (CV = SD/mean) between the housekeeping and regular genes. A good normalization algorithm should have a clear separation of CVs between these two types of genes. To make the CVs from the four algorithms comparable, we transformed the normalized data from the four algorithms into the same scale. In Table 2.3, RCRnorm not only has the smallest two-sided p value, but also confirms the existence of a significant difference between the two gene types at a significance level 0.05. None of the existing methods was able to do so. This suggests the superior performance of RCRnorm in normalizing NanoString data.

<table>
<thead>
<tr>
<th></th>
<th>RCRnorm</th>
<th>NAPPA</th>
<th>NanoStringDiff</th>
<th>NanoStringNorm</th>
</tr>
</thead>
<tbody>
<tr>
<td>p value</td>
<td>0.036</td>
<td>0.146</td>
<td>0.170</td>
<td>0.096</td>
</tr>
</tbody>
</table>

Table 2.3. Colorectal cancer data: two-sided p values from the t test that compares the CVs between housekeeping and regular genes

2.6. Discussion

Motivated by a lung cancer study in predicting adjuvant chemotherapy (ACT) response, we have developed a novel (Bayesian) method, RCRnorm, to normalize NanoString nCounter data. Through simulation studies, we have shown that RCRnorm compares very favorably with the existing methods, especially for situations with an elevated level of heterogeneity from various sources. In the lung cancer application, RCRnorm performs the best, and greatly improves gene-wise correlations between paired FF and FFPE samples. Thus, it offers an important move toward applying gene signatures identified from genome-wide expression profiling of FF samples into wide clinical use with FFPE samples. Meanwhile, it provides improved patient-wise correlations while NanoStringDiff and NanoStringNorm cannot.
The competitive performance of RCRnorm can be largely explained by two unique features it owns. First of all, RCRnorm relies on an integrated system of hierarchical linear regression models with random coefficients, which effectively captures mean structures underlying the data shared among different types of probes, to maximally remove systematic sample-specific biases in gene expression profiling. Unlike RCRnorm, the previous methods use the different types of internal controls in an isolated and somewhat heuristic manner. Thus, they do not take full advantage of the rich information provided by the nCounter system. Secondly, the existing methods adjust sample loading effects with information extracted from housekeeping genes. This is based on the assumption that expression levels of housekeeping genes are stable across samples, implying that their biological variability in gene expression is zero. In practice, their biological variability, although smaller than other genes in general, is not zero. Some housekeeping genes have been reported to have significant fluctuations (Gubern et al., 2009). Thus, this simplifying assumption may lead to compromised performance of the existing methods on normalization. By contrast, RCRnorm does not need the assumption as two separate variance terms, $\sigma_{\kappa}^2$ and $\sigma_{\kappa^*}^2$, are used to model the biological variability of housekeeping and regular genes, respectively. Further, by estimating and comparing $\sigma_{\kappa}^2$ and $\sigma_{\kappa^*}^2$, RCRnorm can provide an alternative way to examine the validity of housekeeping genes used in an nCounter system from an analytical perspective.

In addition, RCRnorm offers much better interpretability than the other methods. It is based on a rigorous model system, whose parameters can be intuitively interpreted. With estimates obtained from the Bayesian approach, researchers can gain a deep understanding about the dataset under study. Moreover, in the integrated system, $X$ represents the log RNA content, whose value is attached with a unit log fM. So for housekeeping and regular genes, the quantity of interest $\kappa$, which is an additive term that makes up $X$, should have the same unit. Thus, the normalized expression produced by RCRnorm also has the unit log fM so that their values are directly comparable to the input amounts of positive controls in the system. This may help us concretely understand how high a gene is expressed.
RCRnorm employs a Bayesian framework to handle its computational needs, and so a non-informative prior setup, as detailed in Section 2.3.2, is adopted. In situations when meaningful prior knowledge is available, prior distributions can be chosen to incorporate such knowledge for improved results. We have developed an efficient Gibbs sampler for posterior computation and inference, where all the steps can be done by direct sampling from known distributions. Based on our numerical experience, the algorithm is computationally stable and converges without manual tuning in all our settings. In the lung cancer application, it took \( \sim 1000 \) seconds to run 15,000 iterations using R on a 2.8 GHz Intel Core i7 processor. Note that based on the proposed model system, frequentist approaches to estimation such as maximum likelihood and nonparametric methods may be used to facilitate computation. Nevertheless, RCRnorm offers the advantage of quantifying the estimation uncertainty easily as it is Bayesian in nature. The existing algorithms, which are all frequentist, cannot even provide confidence intervals for key parameters or normalized expression.

RCRnorm is a method designed based on characteristics of nCounter data observed from FFPE samples. However, the proposed model system does not impose restrictive assumptions that can limit its application to other nCounter data such as those from FF samples. It can also be used with other platforms that use internal controls (with minor adaptions). Further, replicates from the same patient, when available, can be naturally incorporated into the system to enable better estimation of model parameters and normalized expression. This is because RCRnorm, again as a Bayesian method, has the capability to pool information from various sources such as probes, patients, replicates. The other methods can only treat the replicates as independent samples and then calculate the mean or median of normalized expression levels to combine replicates.

Although current technologies cannot reliably quantify RNA degradation levels in FFPE samples, clinical information for these samples such as age of FFPE samples (not the patient age) and RIN (RNA integrity number; a measure used to evaluate RNA quality) (Von Ahlfen et al., 2007), if available, can be integrated to model the degradation levels, to potentially improve the performance on normalization.
Finally, we mention that applying different normalization methods can alter results of downstream analysis. Thus, integrated strategies that add additional structures into the constructed model system can be developed for a routine analysis task, which automatically removes the uncertainty related to normalized expression.

2.7. R package

The proposed method RCRnorm has been implemented by an open source R package that is downloadable from CRAN: https://cran.r-project.org/web/packages/RCRnorm/.

The main function in the package is named ‘RCRnorm’. This function uses the raw read count data from NanoString platform with data for each probe type stored as an element in a list. Users can choose the number of burn in cycles and total number of iterations for the Gibbs sampler. Starting point for the Gibbs sampler can be set to be random or roughly estimated from the data. For fast convergence, starting point is set to be roughly estimated from the data by default. The Gibbs sampler is implemented with the full conditional densities specified as in appendix B.2.
Appendix A

PBNPA: A PERMUTATION-BASED NON-PARAMETRIC ANALYSIS OF CRISPR SCREEN DATA

A.1. Figures

Figure A.1 shows the result of simulation evaluation of positive selection performance using datasets with an increased over-dispersion level while fixing the number of sgRNAs per gene. Figure A.2 shows the result of simulation evaluation of positive selection performance using datasets with an increased over-dispersion level while fixing the proportion of off target sgRNAs. Figure A.3 shows the result of simulation evaluation of negative selection performance using datasets with an increased over-dispersion level while fixing the number of sgRNAs per gene. Figure A.4 shows the result of simulation evaluation of negative selection performance using datasets with an increased over-dispersion level while fixing the proportion of off target sgRNAs.

A.2. R package code for PBNPA

```r
# Used to analyze single dataset (real data used by mageck paper).
get.median = function(x, y, func)
{
  return(tapply(y, x, func)) ## mean vs median
}
# Get permutated p.values. x is an element, y is a vector contains all permutated values.
get.pos.permu.p = function(x, y)
{
  pos.p = sum(y > x, na.rm = T)/length(y)
```

54
Figure A.1. Simulation evaluation of positive selection performance using datasets with an increased over-dispersion level. ROC curves and AUCs are shown for different algorithms with an increasing off target proportion while fixing the number of sgRNAs per gene at 3. Each curve represents the average of ROC curves for 50 simulated datasets and hereafter.
Figure A.2. Simulation evaluation of positive selection performance using datasets with an increased over-dispersion level. ROC curves and AUCs are shown for different algorithms with an increasing number of sgRNAs per gene while fixing the off target proportion at 10%.
Figure A.3. Simulation evaluation of negative selection performance using datasets with an increased over-dispersion level. ROC curves and AUCs are shown for different algorithms with an increasing off target proportion while fixing the number of sgRNAs per gene at 3.
Figure A.4. Simulation evaluation of negative selection performance using datasets with an increased over-dispersion level. ROC curves and AUCs are shown for different algorithms with an increasing number of sgRNAs per gene while fixing the off target proportion at 10%.
Figure A.5. Simulation evaluation of negative selection performance based on recall, precision and F$_1$ for different combinations of sgRNA number per gene (2~6) and off target ratio. Each bar represents the average of 50 simulated datasets and standard error is indicated on the bar.
get.neg.permu.p = function(x, y)
{
  neg.p = sum(y < x, na.rm = T)/length(y)
  return(neg.p)
}

#zs.gene: calculated stat for each gene according to the function
permu.pvalue = function(dat, sim.no = sim.no, zs.gene, func, seed = 7292016)
{
  zs = log(dat[,4]/dat[,3])
  set.seed(seed)
  index.mat = matrix(rep(dat$Gene, sim.no), ncol = sim.no)
  permu.index.mat = apply(index.mat, 2, sample) #each column is a permutation of the
  original gene index.
  result.mat = apply(permu.index.mat, 2, get.median, y = zs, func)
  pos.p = sapply(zs.gene, get.pos.permu.p, as.vector(result.mat))
  neg.p = sapply(zs.gene, get.neg.permu.p, as.vector(result.mat))
  return(data.frame(pos.p, neg.p))
}

#dataset.id is the id number for simulated dataset name; off.ratio is the off target propor-

fold.crispr = function(dat, sim.no = 10, func = "median", alpha.threshold = .2)
{
  dat = dat[order(dat$Gene), ] # so that data set is in increasing order of gene
  dat[is.na(dat)] = 0
  datt = dat
dat[, 3] = datt[, 3] * mean(c(sum(datt[, 3]), sum(datt[, 4]))) / sum(datt[, 3])
dat[, 4] = datt[, 4] * mean(c(sum(datt[, 3]), sum(datt[, 4]))) / sum(datt[, 4])
zs.gene = tapply(log(dat[, 4]/dat[, 3]), dat$Gene, func)  ### mean vs median
initial.p.value = permu.pvalue(dat, sim.no = sim.no, zs.gene = zs.gene, func = func)
initial.adj.pos.pvalue = initial.p.value$pos.p
initial.adj.neg.pvalue = initial.p.value$neg.p
initial.result = data.frame(Gene = sort(unique(dat$Gene)), initial.adj.pos.pvalue, initial.adj.neg.pvalue)
  #summarize genes selected by the program
initial.pos.gene = initial.result$Gene[initial.adj.pos.pvalue < alpha.threshold]
initial.neg.gene = initial.result$Gene[initial.adj.neg.pvalue < alpha.threshold]
update.dat = dat[!is.element(dat$Gene, c(initial.pos.gene, initial.neg.gene)),]
p.value = permu.pvalue(update.dat, sim.no = sim.no, zs.gene = zs.gene, func = func)
final.result = data.frame(Gene = sort(unique(dat$Gene)), pos.pvalue = p.value$pos.p, neg.pvalue = p.value$neg.p)
return(final.result)
}
PBNPA = function(dat, sim.no = 10, alpha.threshold = .2, fdr = .05)
{
nrep = length(dat)
combine.pos = list()
combine.neg = list()
for (i in 1:nrep)
{
  result = fold.crispr(dat[[i]], sim.no = sim.no, alpha.threshold = alpha.threshold)
  combine.pos[[i]] = result$pos.pvalue
  combine.neg[[i]] = result$neg.pvalue
}

combined.pos = metaRNASeq::fishercomb(combine.pos, BHth = fdr)
combined.neg = metaRNASeq::fishercomb(combine.neg, BHth = fdr)
final.result = data.frame(Gene = sort(unique(dat[[1]]$Gene)),
pos.pvalue = combined.pos$rawpval, pos.fdr = combined.pos$adjpval,
neg.pvalue = combined.neg$rawpval, neg.fdr = combined.neg$adjpval)
  pos.gene = final.result$Gene[combined.pos$DEindices] #genes that are selected as positive genes.
  neg.gene = final.result$Gene[combined.neg$DEindices] #genes that are selected as negative genes.
pos.no = length(pos.gene)
  neg.no = length(neg.gene)
return.value = list(pos.gene = pos.gene, pos.no = pos.no, neg.gene = neg.gene, neg.no = neg.no, final.result = final.result)
return(return.value)
Appendix B

RCRNORM: AN INTEGRATED SYSTEM OF RANDOM-COEFFICIENT
HIERARCHICAL REGRESSION MODELS FOR NORMALIZING NANOSTRING
NCOUNTER DATA FROM FFPE SAMPLES

B.1. Tables and figures

B.1.1. Additional information about lung cancer data

<table>
<thead>
<tr>
<th>CodeClass</th>
<th>Name</th>
<th>Patient_1</th>
<th>...</th>
<th>Patient_i</th>
<th>...</th>
<th>Patient_I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative Control</td>
<td>Neg_A(0)</td>
<td>$Y^-_{11}$</td>
<td>...</td>
<td>$Y^-_{i}$</td>
<td>...</td>
<td>$Y^-_{I1}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Negative Control</td>
<td>Neg_H(0)</td>
<td>$Y^-_{18}$</td>
<td>...</td>
<td>$Y^-_{i8}$</td>
<td>...</td>
<td>$Y^-_{I8}$</td>
</tr>
<tr>
<td>Positive Control</td>
<td>Pos_A(128)</td>
<td>$Y^+_{11}$</td>
<td>...</td>
<td>$Y^+_{i1}$</td>
<td>...</td>
<td>$Y^+_{I1}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Positive Control</td>
<td>Pos_F(0.125)</td>
<td>$Y^+_{16}$</td>
<td>...</td>
<td>$Y^+_{i6}$</td>
<td>...</td>
<td>$Y^+_{I6}$</td>
</tr>
<tr>
<td>Housekeeping Gene</td>
<td>Gene*_1</td>
<td>$Y^*_{11}$</td>
<td>...</td>
<td>$Y^*_{i1}$</td>
<td>...</td>
<td>$Y^*_{I1}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Housekeeping Gene</td>
<td>Gene*_7</td>
<td>$Y^*_{17}$</td>
<td>...</td>
<td>$Y^*_{i7}$</td>
<td>...</td>
<td>$Y^*_{I7}$</td>
</tr>
<tr>
<td>Regular Gene</td>
<td>Gene_1</td>
<td>$Y_{11}$</td>
<td>...</td>
<td>$Y_{i1}$</td>
<td>...</td>
<td>$Y_{I1}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Regular Gene</td>
<td>Gene_87</td>
<td>$Y_{1,87}$</td>
<td>...</td>
<td>$Y_{i,87}$</td>
<td>...</td>
<td>$Y_{I,87}$</td>
</tr>
</tbody>
</table>

Table B.1. Lung cancer data: data structure from the NanoString nCounter platform
Figure B.1. Lung cancer data: read counts of 6 positive controls from FFPE samples of the 28 patients. The legend shows that the amount of RNA targeted by each of the six probes.

B.1.2. Additional simulation results

Figure B.2, B.3 and B.4 showed the above mentioned simulation results of different percentiles.

B.1.3. Results for colorectal cancer data

Table B.2 and figure B.5 showed the additional results for colorectal cancer data.

B.2. Full conditionals

Let $\theta|\cdots$ denote $\theta$ given the observed data and all the other parameters and latent variables (i.e., $\theta|Y, \Theta_{-\theta}$). The full conditional posterior distributions based on the priors specified in Section 2.3 are given below.

1. For $i = 1, \ldots, I$,

$$a_i|\cdots \sim N\left(\frac{A_1}{\sigma^2_{e^{-}}} + \frac{A_2 + A_3 + A_4}{\sigma^2_{e^{-}}} + \frac{\mu_a}{\sigma^2_{a}}, \frac{1}{\sigma^2_{a}}\right)$$
Figure B.2. Simulation study I for mimicking various real-world scenarios: boxplots for 25th, 50th and 75th percentiles of gene-wise Spearman correlations between normalized data and true expression levels based on 50 replicates for each of the five settings I1-I5. Compare to the basic setting I-1 (parameter values estimated from the FFPE samples in the lung cancer application), gene expression variability is increased in I-2, lane-by-lane variation is increased in I-3, probe-level variation is increased in I-4, and variability of random noise is increased in I-5. Note that NSnorm stands for NanoStringNorm and NSdiff for NanoStringDiff.
Figure B.3. Simulation study II for robustness checking: boxplots for 25th, 50th and 75th percentiles of gene-wise Spearman correlations between normalized data and true expression levels based on 50 replicates for each of the five settings II1-II5. In II1, a thick-tail distribution $t_3$ was used to simulate probe effects; in II-2, $t_3$ was used to simulate random errors; in II-3, a right-skewed distribution $G_{2,1}$ was used to simulate probe effects; in II-4, $G_{2,1}$ was used to simulate random errors; and in II-5, $t_3$ was used to generate probe effects and $G_{2,1}$ to generate random errors. Except for the changes above, everything remains the same as in the basic setting I-1 including all parameter values (so $t_3$ and $G_{2,1}$ need to be rescaled or shifted). Note that NSnorm stands for NanoStringNorm and NSdiff for NanoStringDiff.
Figure B.4. Simulation study III for mimicking different RNA degradation levels between genes: boxplots for 25th, 50th and 75th percentiles of gene-wise Spearman correlations between normalized data and true expression levels based on 50 replicates for each of the two settings III1-III2, where $\delta_{ir}$ and $\delta_{ih}^\ast$ were generated from a normal distribution with mean 0 and SD 0.1 and 0.4, respectively. Note that NSnorm stands for NanoStringNorm and NSdiff for NanoStringDiff.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>median</th>
<th>SD</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_a$</td>
<td>2.281</td>
<td>2.281</td>
<td>0.004</td>
<td>(2.274, 2.290)</td>
</tr>
<tr>
<td>$\mu_b$</td>
<td>0.946</td>
<td>0.946</td>
<td>0.002</td>
<td>(0.942, 0.949)</td>
</tr>
<tr>
<td>$c$</td>
<td>-1.750</td>
<td>-1.720</td>
<td>0.177</td>
<td>(-2.238, -1.499)</td>
</tr>
<tr>
<td>$\sigma_{\kappa_x}$</td>
<td>0.191</td>
<td>0.190</td>
<td>0.010</td>
<td>(0.176, 0.215)</td>
</tr>
<tr>
<td>$\sigma_\kappa$</td>
<td>0.279</td>
<td>0.278</td>
<td>0.010</td>
<td>(0.264, 0.303)</td>
</tr>
<tr>
<td>$\sigma_{d^-}$</td>
<td>0.289</td>
<td>0.257</td>
<td>0.139</td>
<td>(0.127, 0.708)</td>
</tr>
<tr>
<td>$\sigma_d$</td>
<td>0.075</td>
<td>0.069</td>
<td>0.026</td>
<td>(0.041, 0.144)</td>
</tr>
<tr>
<td>$\sigma_{e^-}$</td>
<td>0.273</td>
<td>0.273</td>
<td>0.011</td>
<td>(0.250, 0.295)</td>
</tr>
<tr>
<td>$\sigma_e$</td>
<td>0.042</td>
<td>0.042</td>
<td>0.002</td>
<td>(0.039, 0.047)</td>
</tr>
</tbody>
</table>

Table B.2. Colorectal cancer data: posterior summary statistics of global parameters from applying RCRnorm to FFPE samples
Figure B.5. Colorectal cancer data: posterior densities of global parameters from applying RCRnorm to FFPE samples
where \( A_1 = \sum_{n=1}^{N} (Y_{in}^- - b_i c - d_n^-) \), \( A_2 = \sum_{p=1}^{P} (Y_{ip}^+ - b_i d_p^+) \), \( A_3 = \sum_{h=1}^{H} (Y_{ih}^* - b_i (\phi_i + \kappa_{ih}^*)) \) and \( A_4 = \sum_{r=1}^{R} (Y_{ir} - b_i (\phi_i + \kappa_{ir})) \).

2. For \( i = 1, \ldots, I \),

\[
\begin{align*}
 \phi_i | \cdots & \sim N \left( \frac{\sum_{h=1}^{H} (Y_{ih}^* - a_i - b_i \kappa_{ih}^*) + \sum_{r=1}^{R} (Y_{ir} - a_i - b_i \kappa_{ir})}{(H + R) \cdot b_i}, \frac{\sigma^2}{(H + R) \cdot b_i^2} \right) \cdot I_{(L_{\phi_i}, U_{\phi_i})}(\phi_i)
\end{align*}
\]

which is truncated normal on the prior range \((L_{\phi_i}, U_{\phi_i})\). After we draw \( \phi_1, \ldots, \phi_{I-1} \), we set \( \phi_I = - \sum_{1}^{I-1} \phi_i \).

3. For \( i = 1, \ldots, I - 1 \),

\[
\begin{align*}
 \kappa_{ih}^* | \cdots & \sim N \left( \frac{b_i (Y_{ih}^* - a_i - b_i \phi_i)}{\sigma_e^2} + \frac{\lambda_{k_i}}{\sigma_{k_i}^2}, \frac{1}{\sigma_e^2} \right) \cdot I_{(\lambda_{k_i}, \infty)}(\kappa_{ih}^*)
\end{align*}
\]

4. For \( i = 1, \ldots, I \) and \( h = 1, \ldots, H \),

\[
\begin{align*}
 \kappa_{ih} | \cdots & \sim N \left( \frac{b_i (Y_{ih}^* - a_i - b_i \phi_i)}{\sigma_e^2} + \frac{\lambda_{k_i}}{\sigma_{k_i}^2}, \frac{1}{\sigma_e^2} \right) \cdot I_{(\lambda_{k_i}, \infty)}(\kappa_{ih})
\end{align*}
\]

5. For \( i = 1, \ldots, I \) and \( r = 1, \ldots, R \),

\[
\begin{align*}
 \kappa_{ir} | \cdots & \sim N \left( \frac{b_i (Y_{ir} - a_i - b_i \phi_i)}{\sigma_e^2} + \frac{\lambda_{k_i}}{\sigma_{k_i}^2}, \frac{1}{\sigma_e^2} \right) \cdot I_{(\lambda_{k_i}, \infty)}(\kappa_{ir})
\end{align*}
\]
6. For \( h = 1, \ldots, H \),

\[
\lambda_h^* | \ldots \sim N \left( \frac{\sum_{i=1}^I \kappa_{ih}^*}{I}, \frac{\sigma_{K_h}^2}{I} \right) \cdot I_{(L_{\lambda_h^*}, U_{\lambda_h^*})}(\lambda_h^*)
\]

which is truncated normal on the prior range \((L_{\lambda_h^*}, U_{\lambda_h^*})\).

7. For \( r = 1, \ldots, R \),

\[
\lambda_r | \ldots \sim N \left( \frac{\sum_{i=1}^I \kappa_{ip}^*}{I}, \frac{\sigma_{\lambda_r}^2}{I} \right) \cdot I_{(L_{\lambda_r}, U_{\lambda_r})}(\lambda_r)
\]

which is truncated normal on the prior range \((L_{\lambda_r}, U_{\lambda_r})\).

8. For \( n = 1, \ldots, N \),

\[
d_n^- | \ldots \sim N \left( \frac{\sum_{i=1}^I (Y_{in} - a_i - b_i c_i)}{\sigma_{e_n}^2}, \frac{1}{\frac{I}{\sigma_{e_n}^2} + \frac{1}{\sigma_{d_n}^2}} \right)
\]

9. For \( p = 1, \ldots, P \),

\[
d_p^+ | \ldots \sim N \left( \frac{\sum_{i=1}^I (Y_{ip} - a_i - b_i X_{ip})}{\sigma_{e_p}^2}, \frac{1}{\frac{I}{\sigma_{e_p}^2} + \frac{1}{\sigma_{d_p}^2}} \right)
\]

10. For \( h = 1, \ldots, H \),

\[
d_h^* | \ldots \sim N \left( \frac{\sum_{i=1}^I (Y_{ih}^* - a_i - b_i (\phi_i + \kappa_{ih}^*))}{\sigma_{e_h}^2}, \frac{1}{\frac{I}{\sigma_{e_h}^2} + \frac{1}{\sigma_{d_h}^2}} \right)
\]

11. For \( R = 1, \ldots, R \),

\[
d_r | \ldots \sim N \left( \frac{\sum_{i=1}^I (Y_{ir} - a_i - b_i (\phi_i + \kappa_{ir}^*))}{\sigma_{e_r}^2}, \frac{1}{\frac{I}{\sigma_{e_r}^2} + \frac{1}{\sigma_{d_r}^2}} \right)
\]
12. For global parameters $c, \mu_a, \mu_b,$

$$c|... \sim N \left( \frac{\sum_{i=1}^{I} \sum_{n=1}^{N} b_i \cdot (Y_{in}^- - a_i - d_{n}^-)}{N \sum_{i=1}^{I} b_i^2}, \frac{\sigma_{c-}^2}{N \sum_{i=1}^{I} b_i^2} \right) \cdot I_{(-6, -1)}(c)$$

which is truncated normal on the prior range $(-6, -1);$

$$\mu_a|... \sim N \left( \frac{\sum_{i=1}^{I} a_i + \frac{\hat{\mu}_a}{(m \times SE(\mu_a))^2}}{\frac{1}{\sigma_a^2} + \frac{1}{(m \times SE(\mu_a))^2}}, \frac{1}{\sigma_a^2} + \frac{1}{(m \times SE(\mu_a))^2} \right)$$

which is truncated normal on the prior range $(L_{\mu a}, U_{\mu a});$

$$\mu_b|... \sim N \left( \frac{\sum_{i=1}^{I} b_i + \frac{\hat{\mu}_b}{(m \times SE(\mu_b))^2}}{\frac{1}{\sigma_b^2} + \frac{1}{(m \times SE(\mu_b))^2}}, \frac{1}{\sigma_b^2} + \frac{1}{(m \times SE(\mu_b))^2} \right)$$

which is truncated normal on on the prior range $(L_{\mu b}, U_{\mu b}).$

13. For all the variance parameters,

$$\sigma_a^2|... \sim IG \left( u + \frac{I}{2}, v + \frac{\sum_{i=1}^{I} (a_i - \mu_a)^2}{2} \right).$$

$$\sigma_b^2|... \sim IG \left( u + \frac{I}{2}, v + \frac{\sum_{i=1}^{I} (b_i - \mu_b)^2}{2} \right).$$

$$\sigma_d^-|... \sim IG \left( u + \frac{N}{2}, v + \frac{\sum_{n=1}^{N} (d_{n}^-)^2}{2} \right).$$

$$\sigma_d^2|... \sim IG \left( u + \frac{P + H + R}{2}, v + \frac{\sum_{p=1}^{P} (d_p^+)^2 + \sum_{h=1}^{H} (d_h^+)^2 + \sum_{r=1}^{R} (d_r)^2}{2} \right).$$

$$\sigma_{c-}^2|... \sim IG \left( u + \frac{IN}{2}, v + \frac{\sum_{i=1}^{I} \sum_{n=1}^{N} (Y_{in}^- - a_i - b_i c - d_{n}^-)^2}{2} \right).$$

$$\sigma_{c+}^2|... \sim IG \left( u + \frac{I(P + H + R)}{2}, v + \frac{\sum_{i=1}^{I} D_i}{2} \right)$$
where $D_i = \sum_{p=1}^{P} (Y_{ip}^+ - a_i - b_iX_{ip}^+ - d_p^+) + \sum_{h=1}^{H} (Y_{ih}^+ - a_i - b_i(\phi_i + \kappa_{ih}^*))^2 + \sum_{r=1}^{R} (Y_{ir} - a_i - b_i(\phi_i + \kappa_{ir}))^2$.

$$\sigma_{\kappa s}^2 \cdots \sim IG \left( u + \frac{IH}{2}, v + \sum_{i=1}^{I} \sum_{h=1}^{H} \frac{(\kappa_{ih} - \lambda_{h}^*)^2}{2} \right).$$

$$\sigma_{\kappa r}^2 \cdots \sim IG \left( u + \frac{IR}{2}, v + \sum_{i=1}^{I} \sum_{r=1}^{R} \frac{(\kappa_{ir} - \lambda_{r})^2}{2} \right).$$

In our preliminary experiments, we find that updating $\sigma_d^2$ only using data from positive controls would greatly facilitate the convergence of the algorithm while achieving almost the same results. Thus, in our R package named RCRnorm, $\sigma_d^2$ is updated using

$$\sigma_d^2 \cdots \sim IG \left( u + \frac{P}{2}, v + \sum_{p=1}^{P} \frac{(d_p^+)^2}{2} \right).$$

### B.3. R package code for RCRnorm

```r
# get coefficient of linear regression from positive ctrl
fitWithPosCtrl = function(y, x)
{
  mod1 = stats::lm(y ~ x)
  coefs = stats::coef(mod1)
  unname(coefs)
}

# get prior range of uniform distribution
get_range = function(x, mm = 5)
{
  c(mean(x)-mm*stats::sd(x), mean(x)+mm*stats::sd(x))
}

# get residual from positive ctrl fitted with simple linear regression
```
get_residual = function(log_dat, RNA_conc, coefs)
  #in matrix format
  {
    log_dat - sweep(sweep(RNA_conc, 2, coefs[2, ], '*'), 2, coefs[1, ], '+')
  }
  #main function to implement RCRnorm
  RCRnorm = function(dat, pos_conc = log10(c(128, 32, 8, 2, 0.5, 0.125)), iter = 8000,
                     warmup = 5000, random_init = F, all_dat = T, seed = 1, mm = 3, m_ab = 9)
  {
    ptm <- proc.time()
    set.seed(seed*3723)
    #log10 transform the original count data
    pos_dat = log10(dat$pos_dat + 1)
    neg_dat = log10(dat$neg_dat + 1)
    hk_dat = log10(dat$hk_dat + 1)
    reg_dat = log10(dat$reg_dat + 1)
    #inverse gamma parameter
    u = v = .01
    #number of each class of genes.
    n_hk = dim(hk_dat)[1]
    n_reg = dim(reg_dat)[1]
    n_neg = dim(neg_dat)[1]
    n_pos = dim(pos_dat)[1]
    #number of patients or samples
    n_patient = dim(pos_dat)[2]
    #number of MCMC iteration used to calculate posterior after convergence
    iter_keep = iter - warmup
    #calculate the coefficient for each patient; note: with positive controls.
# all_coef: First row: a+5se; second row: b+5se; third row: a-5se; fourth row: b-5se.

```r
all_coef = apply(pos_dat, 2, fitWithPosCtrl, pos_conc)
mu_a_itm = mean(all_coef[1,])
mu_b_itm = mean(all_coef[2,])
```

# Jacknife to estimate mean and variance of mu_a and mu_b
```
mu_a = numeric()
mu_b = numeric()
for (i in 1:500)
{
  mu_a[i] = mean(sample(all_coef[1,], n_patient - 2))
  mu_b[i] = mean(sample(all_coef[2,], n_patient - 2))
}
mu_a_mu = mean(mu_a)
cat(mu_a_mu, '\n')
mu_b_mu = mean(mu_b)
cat(mu_b_mu, '\n')
sigma2_mu_a = m_ab * stats::var(mu_a)
cat(sigma2_mu_a, '\n')
sigma2_mu_b = m_ab * stats::var(mu_b)
cat(sigma2_mu_b, '\n')
hk_RNA = sweep(sweep(hk_dat, 2, all_coef[1,], '-'), 2, all_coef[2,], '/')
reg_RNA = sweep(sweep(reg_dat, 2, all_coef[1,], '-'), 2, all_coef[2,], '/')
```

## estimate genes’ mean expression level range
```
lambda_hk_range = apply(hk_RNA, 1, get_range, mm = mm)
lambda_reg_range = apply(reg_RNA, 1, get_range, mm = mm)
```

# estimate patient effect range by two way ANOVA with patient’s regular gene expression level.
```
gene = factor(rep(1:n_reg, n_patient))
```
patient = factor(rep(1:n_patient, each = n_reg))
mod = stats::lm(unlist(reg_RNA) ~ patient + gene, contrasts = list(patient = 'contr.sum',
gene = 'contr.sum'))
phi = numeric(n_patient)
phi[1:(n_patient - 1)] = summary(mod)$coefficients[2:n_patient, 1]
phi[n_patient] = -sum(phi)
phi_L = phi - mm * summary(mod)$coefficients[2, 2]
phi_U = phi + mm * summary(mod)$coefficients[2, 2]
#initialize all the parameters
aa = matrix(NA, ncol = n_patient, nrow = iter_keep)
bb = matrix(NA, ncol = n_patient, nrow = iter_keep)
cc = numeric(iter_keep)
phi_return = matrix(NA, ncol = n_patient, nrow = iter_keep)  # patient effect
kappa_hk = matrix(NA, ncol = n_patient * n_hk, nrow = iter_keep)
kappa_reg = matrix(NA, ncol = n_patient * n_reg, nrow = iter_keep)
lambda_hk = matrix(NA, ncol = n_hk, nrow = iter_keep)
lambda_reg = matrix(NA, ncol = n_reg, nrow = iter_keep)
d_neg = matrix(NA, ncol = n_neg, nrow = iter_keep)
d_pos = matrix(NA, ncol = n_pos, nrow = iter_keep)
d_hk = matrix(NA, ncol = n_hk, nrow = iter_keep)
d_reg = matrix(NA, ncol = n_reg, nrow = iter_keep)
mu_a = numeric(iter_keep)
mu_b = numeric(iter_keep)
sigma2e_neg = numeric(iter_keep)
sigma2e_phr = numeric(iter_keep)
sigma2a = numeric(iter_keep)
sigma2b = numeric(iter_keep)
sigma2kappa_hk = numeric(iter_keep)
if (random_init == T)
{
  sigma2a_itm = stats::runif(1, 0, .01)
  sigma2b_itm = stats::runif(1, 0, .01)
  a_itm = stats::rnorm(n_patient, 2.5, .1)
  b_itm = stats::rnorm(n_patient, .9, .1)
  cc_itm = stats::runif(1, -6, -1)
  phi_itm = stats::rnorm(n_patient, 0, 2)
  phi_itm[n_patient] = -sum(phi_itm[1:(n_patient - 1)])
  kappa_hk_itm = stats::rnorm(n_hk * n_patient, 0, 1)
  sigma2kappa_hk_itm = stats::runif(1, 0, 1)
  kappa_reg_itm = stats::rnorm(n_reg * n_patient, 0, 1)
  sigma2kappa_reg_itm = stats::runif(1, 0, 1)
  lambda_hk_itm = stats::rnorm(n_hk, 0, 1)
  lambda_reg_itm = stats::rnorm(n_reg, 0, 1)
  d_neg_itm = stats::rnorm(n_neg, 0, .01)
  sigma2d_neg_itm = stats::runif(1, 0, .1)
  d_pos_itm = stats::rnorm(n_pos, 0, .01)
  sigma2d_phr_itm = stats::runif(1, 0, .1)
  d_hk_itm = stats::rnorm(n_hk, 0, .01)
  d_reg_itm = stats::rnorm(n_reg, 0, .01)
  sigma2e_neg_itm = stats::runif(1, 0, .1)
  sigma2e_phr_itm = stats::runif(1, 0, .1)
}

# get initial values; itm: intermediate
if (random_init == F)
{
    sigma2a_itm = stats::var(all_coef[1,])
    sigma2b_itm = stats::var(all_coef[2,])
    a_itm = all_coef[1,]
    b_itm = all_coef[2,]
    cc_itm = mean(unlist(sweep(sweep(neg_dat, 2, all_coef[1,], '-'), 2, all_coef[2,], '/')))
    phi_itm = phi lambda_hk_itm = apply(lambda_hk_range, 2, mean)
    lambda_reg_itm = apply(lambda_reg_range, 2, mean)
    estimate_kappa = sweep(rbind(hk_RNA, reg_RNA), 2, phi_itm, '-')
    estimate_kappa_var = sweep(sweep(rbind(hk_RNA, reg_RNA), 2, phi_itm, '-'), 1, c(lambda_hk_itm, lambda_reg_itm), '-')
    kappa_hk_itm = as.vector(unlist(estimate_kappa[1:n_hk,]))
    sigma2kappa_hk_itm = stats::var(as.vector(unlist(estimate_kappa_var[1:n_hk,])))
    kappa_reg_itm = as.vector(unlist(estimate_kappa[(1+n_hk):(n_hk+n_reg),]))
    sigma2kappa_reg_itm = stats::var(as.vector(unlist(estimate_kappa_var[(1+n_hk):(n_hk+n_reg)],)))
    pos_RNA = matrix(rep(pos_conc, n_patient), ncol = n_patient)
    neg_RNA = matrix(rep(cc_itm, n_neg * n_patient), ncol = n_patient)
    d_neg_itm = apply(get_residual(neg_dat, neg_RNA, all_coef), 1, mean)
    sigma2d_neg_itm = stats::var(d_neg_itm)
    d_pos_itm = apply(get_residual(pos_dat, pos_RNA, all_coef), 1, mean)
    sigma2d_phr_itm = stats::var(d_pos_itm)
    d_hk_itm = rep(0, n_hk)
    d_reg_itm = rep(0, n_reg)
    sigma2e_neg_itm = stats::var(unlist(sweep(get_residual(neg_dat, neg_RNA, all_coef), 1, d_neg_itm, '-')))}
\[ \text{sigma2e\_phr\_itm} = \text{stats::var(unlist(sweep(get\_residual(pos\_dat, pos\_RNA, all\_coef),} \\
1, d\_pos\_itm, \text{\textquotesingle-\textquotesingle}))} \]

for (i in 1:iter)
{
    \[ \text{A2} = \text{colSums(sweep(pos\_dat - pos\_conc \%o\% b\_itm, 1, d\_pos\_itm, \text{\textquotesingle-\textquotesingle})} \]
    \[ \text{B2} = \text{colSums(sweep(sweep(pos\_dat, 2, a\_itm, \text{\textquotesingle-\textquotesingle}), 1, d\_pos\_itm, \text{\textquotesingle-\textquotesingle}), 1, pos\_conc, \text{\textquotesingle*\textquotesingle})} \]
    if (all\_dat == F)
    {
        \[ \text{a\_itm} = \text{stats::rnorm(rep(1, n\_patient), ((A2)/sigma2e\_phr\_itm} \\
            + \text{mu\_a\_itm}/sigma2a\_itm)/ ((n\_pos)/sigma2e\_phr\_itm + 1/sigma2a\_itm),
            \sqrt{1/((n\_pos)/sigma2e\_phr\_itm + 1/sigma2a\_itm)})) \]
        \[ \text{b\_itm} = \text{stats::rnorm(rep(1, n\_patient), ((B2)/sigma2e\_phr\_itm} \\
            + \text{mu\_b\_itm}/sigma2b\_itm)/ ((\text{sum}(pos\_conc^2))/sigma2e\_phr\_itm \\
            + 1/sigma2b\_itm), \sqrt{1/((\text{sum}(pos\_conc^2))/sigma2e\_phr\_itm} \\
            + 1/sigma2b\_itm))) \]
    } else
    {
        \[ \text{A1} = \text{colSums(sweep(sweep(neg\_dat, 2, b\_itm*cc\_itm, \text{\textquotesingle-\textquotesingle}), 1, d\_neg\_itm, \text{\textquotesingle-\textquotesingle})} \]
        \[ \text{A3} = \text{colSums(sweep(hk\_dat - matrix(rep(b\_itm, each = n\_hk) * (rep(phi\_itm, each = n\_hk) + kappa\_hk\_itm), nrow = n\_hk), 1, d\_hk\_itm, \text{\textquotesingle-\textquotesingle})} \]
        \[ \text{A4} = \text{colSums(sweep(reg\_dat - matrix(rep(b\_itm, each = n\_reg) * (rep(phi\_itm, each = n\_reg) + kappa\_reg\_itm), nrow = n\_reg), 1, d\_reg\_itm, \text{\textquotesingle-\textquotesingle})} \]
        \[ \text{a\_itm} = \text{stats::rnorm(rep(1, n\_patient), ((A1/sigma2e\_neg\_itm} + (A2+A3+A4)/ \\
            sigma2e\_phr\_itm + mu\_a\_itm/sigma2a\_itm))/ (n\_neg/ \\
            sigma2e\_neg\_itm + (n\_pos+n\_hk+n\_reg)/sigma2e\_phr\_itm} \]
    
78
\[ \sigma^2_{\text{e}_\text{neg}_\text{itm}} = \sigma^2_{\text{a}_\text{itm}} + \frac{1}{\sigma^2_{\text{e}_\text{itm}}} \]

\[ B_1 = \text{cc}_\text{itm} \times \text{colSums(sweep(sweep(neg_dat, 2, a_{itm}, \cdot\cdot\cdot), 1, d_{neg}_\text{itm}, \cdot\cdot\cdot))} \]

\[ B_3 = \text{colSums(matrix(rep(\phi_{itm}, each = n_{hk}) + \kappa_{hk}_\text{itm}, nrow = n_{hk})\times\text{sweep(sweep(hk_dat, 2, a_{itm}, \cdot\cdot\cdot), 1, d_{hk}_\text{itm}, \cdot\cdot\cdot))} \]

\[ B_4 = \text{colSums(matrix(rep(\phi_{itm}, each = n_{reg}) + \kappa_{reg}_\text{itm}, nrow = n_{reg})\times\text{sweep(sweep(reg_dat, 2, a_{itm}, \cdot\cdot\cdot), 1, d_{reg}_\text{itm}, \cdot\cdot\cdot))} \]

\[ b_{itm} = \text{stats::rnorm(rep(1, n_{patient}), ((B_1/\sigma^2_{\text{e}_\text{neg}_\text{itm}}) + (B_2 + B_3 + B_4)/\sigma^2_{\text{e}_\text{phr}_\text{itm}} + \mu_{b_{itm}}/\sigma^2_{b_{itm}}) / (n_{neg} \times (\text{cc}_\text{itm}^2)/\sigma^2_{\text{e}_\text{neg}_\text{itm}} + (\text{sum(pos_conc}^2) + \text{colSums(matrix((rep(\phi_{itm}, each = n_{hk}) + \kappa_{hk}_\text{itm})^2, nrow = n_{hk}) + \text{colSums(matrix((rep(\phi_{itm}, each = n_{reg}) + \kappa_{reg}_\text{itm})^2, nrow = n_{reg}))))/\sigma^2_{\text{phr}_\text{itm}} + 1/\sigma^2_{b_{itm}}), \text{sqrt}(1/(n_{neg} \times (\text{cc}_\text{itm}^2)/\sigma^2_{\text{e}_\text{neg}_\text{itm}} + (\text{sum(pos_conc}^2) + \text{colSums(matrix((rep(\phi_{itm}, each = n_{hk}) + \kappa_{hk}_\text{itm})^2, nrow = n_{hk})) + \text{colSums(matrix((rep(\phi_{itm}, each = n_{reg}) + \kappa_{reg}_\text{itm})^2, nrow = n_{reg}))))/\sigma^2_{\text{phr}_\text{itm}} + 1/\sigma^2_{b_{itm}}))}) \]

\[ cc_{\text{itm}} = \text{truncnorm::rtruncnorm(1, -6, -1, \text{sum(sweep(sweep(sweep(neg_dat, 2, a_{itm}, \cdot\cdot\cdot), 1, d_{neg}_\text{itm}, \cdot\cdot\cdot), 2, b_{itm}, \cdot\cdot\cdot)) / (n_{neg} \times \text{sum(b_{itm}^2)}), \text{sqrt(\sigma^2_{\text{e}_\text{neg}_\text{itm}}/(n_{neg} \times \text{sum(b_{itm}^2))}))} = \text{numeric(n_{patient})} \]

\[ \phi_{itm}[1:(n_{patient} - 1)] = \text{truncnorm::rtruncnorm(rep(1, n_{patient}), phi_L, phi_U, (\text{colSums(hk_dat-matrix(rep(a_{itm}, each = n_{hk}) + rep(b_{itm}, each = n_{hk}) \times \kappa_{hk}_\text{itm} + rep(d_{hk}_\text{itm}, n_{patient}), nrow = n_{hk}) + \text{colSums(reg_dat-matrix(rep(a_{itm}, each = n_{reg}) + rep(b_{itm}, each = n_{reg}) \times \cdot\cdot\cdot))}))} \]

79
kappa_reg_itm = stats::rnorm(rep(1, n_patient * n_reg), ((rep(b_itm, each = n_reg)*(unlist(reg_dat) - rep(a_itm, each = n_reg) - rep(b_itm*phi_itm, each = n_reg) - rep(d_reg_itm, n_patient)))/ sigma2e_phr_itm + rep(lambda_reg_itm, n_patient)/sigma2kappa_reg_itm)/ rep((b_itm^2)/sigma2e_phr_itm + 1/sigma2kappa_reg_itm, each = n_reg), sqrt(rep(1/(b_itm^2)/sigma2e_phr_itm + 1/sigma2kappa_reg_itm), each = n_reg))

lambda_reg_itm = truncnorm::rtruncnorm(rep(1, n_reg), lambda_reg_range[1,], lambda_reg_range[2,], rowMeans(matrix(kappa_reg_itm, nrow = n_reg)), sqrt(sigma2kappa_reg_itm/n_patient))

d_neg_itm = stats::rnorm(n_neg, rowSums(sweep(neg_dat, 2, a_itm + cc_itm*b_itm, '-'))/sigma2e_neg_itm)/(n_patient/sigma2e_neg_itm + 1/sigma2d_neg_itm), sqrt(1/(n_patient/sigma2e_neg_itm + 1/sigma2d_neg_itm)))

d_pos_itm = stats::rnorm(n_pos, rowSums(sweep(pos_dat-pos_conc %o% b_itm, 2, a_itm, '-'))/
\[
\frac{\sigma_{2e\_phr\_itm}}{(n\_patient/\sigma_{2e\_phr\_itm} + 1/\sigma_{2d\_phr\_itm})}, \\
sqrt{1/(n\_patient/\sigma_{2e\_phr\_itm} + 1/\sigma_{2d\_phr\_itm})}
\]

d_{hk\_itm} = stats::rnorm(n\_hk, rowSums(sweep(hk\_dat\_matrix(rep(b\_itm, \\
each = n\_hk)*(rep(\phi\_itm, each = n\_hk)+kappa\_hk\_itm),nrow = n\_hk), \\
2, a\_itm, \'-')/\sigma_{2e\_phr\_itm})/(n\_patient/\sigma_{2e\_phr\_itm} + \\
1/\sigma_{2d\_phr\_itm}), sqrt(1/(n\_patient/\sigma_{2e\_phr\_itm} + 1/\sigma_{2d\_phr\_itm})))

d_{reg\_itm} = stats::rnorm(n\_reg, rowSums(sweep(reg\_dat\_matrix(rep(b\_itm, \\
each = n\_reg)*(rep(\phi\_itm, each = n\_reg)+kappa\_reg\_itm),nrow = n\_reg), \\
2, a\_itm, \'-')/\sigma_{2e\_phr\_itm})/(n\_patient/\sigma_{2e\_phr\_itm} + 1/\sigma_{2d\_phr\_itm}), \\
sqrt(1/(n\_patient/\sigma_{2e\_phr\_itm} + 1/\sigma_{2d\_phr\_itm})))

mu_{a\_itm} = stats::rnorm(1, (sum(a\_itm)/\sigma_{2a\_itm} + \\
mu_{a\_mu}/\sigma_{2\mu\_a})/(n\_patient/\sigma_{2a\_itm} + 1/\sigma_{2\mu\_a}), \\
sqrt(1/(n\_patient/\sigma_{2a\_itm} + 1/\sigma_{2\mu\_a})))

mu_{b\_itm} = stats::rnorm(1, (sum(b\_itm)/\sigma_{2b\_itm} + \\
mu_{b\_mu}/\sigma_{2\mu\_b})/(n\_patient/\sigma_{2b\_itm} + 1/\sigma_{2\mu\_b}), \\
sqrt(1/(n\_patient/\sigma_{2b\_itm} + 1/\sigma_{2\mu\_b})))

\sigma_{2e\_neg\_itm} = 1/stats::rgamma(1, u + n\_patient*n\_neg/2, v+sum(sweep(sweep( \\
neg\_dat, 2, a\_itm + b\_itm*cc\_itm, \'-') , 1, d\_neg\_itm, \'-')^2)/2)

\sigma_{2e\_phr\_itm} = 1/stats::rgamma(1, u + n\_patient*(n\_pos+n\_hk+n\_reg)/2, \\
v + (sum(sweep(sweep(pos\_dat - pos\_conc \%\% b\_itm, 2, a\_itm, \\
\', 1, d\_pos\_itm, \'-')^2) + sum(sweep(hk\_dat - matrix(rep(a\_itm, \\
each = n\_hk) + rep(b\_itm, each = n\_hk)*(rep(\phi\_itm, each = n\_hk)+kappa\_hk\_itm), \\
nrow = n\_hk)) , 1, d\_hk\_itm, \'-')^2) + sum(sweep(reg\_dat - matrix(rep(a\_itm, each = \\
n\_reg) + rep(b\_itm, \\
each = n\_reg)*(rep(\phi\_itm, each = n\_reg)+kappa\_reg\_itm), \\
nrow = n\_reg)) , 1, d\_reg\_itm, \'-')^2))/2)

\sigma_{2a\_itm} = 1/stats::rgamma(1, u + n\_patient/2, \\
v + sum((a\_itm - mu_{a\_itm})^2)/2)
\[
\sigma_{b_{itm}}^2 = \frac{1}{\text{stats::rgamma}(1, u + n_{patient}/2, v + \sum((b_{itm} - \mu_{b_{itm}})^2)/2)}
\]

\[
\sigma_{\kappa_{hk_{itm}}}^2 = \frac{1}{\text{stats::rgamma}(1, u + n_{patient}*n_{hk}/2, v + \sum((\kappa_{hk_{itm}} - \text{rep}(\lambda_{hk_{itm}}, n_{patient}))^2)/2)}
\]

\[
\sigma_{\kappa_{reg_{itm}}}^2 = \frac{1}{\text{stats::rgamma}(1, u + n_{patient}*n_{reg}/2, v + \sum((\kappa_{reg_{itm}} - \text{rep}(\lambda_{reg_{itm}}, n_{patient}))^2)/2)}
\]

\[
\text{cat(paste(round(\sigma_{\kappa_{reg_{itm}}}^2, 3),"\n"))}
\]

\[
\sigma_{d_{neg_{itm}}}^2 = \frac{1}{\text{stats::rgamma}(1, u + n_{neg}/2, v + \sum(d_{neg_{itm}}^2)/2)}
\]

\[
\sigma_{d_{phr_{itm}}}^2 = \frac{1}{\text{stats::rgamma}(1, u + n_{pos}/2, v + \sum(d_{pos_{itm}}^2)/2)}
\]

if(i > warmup)
{
  j = i - warmup

  aa[j,] = a_{itm}

  bb[j,] = b_{itm}

  cc[j] = cc_{itm}

  phi_{return}[j,] = phi_{itm}

  kappa_{hk}[j,] = kappa_{hk_{itm}}

  kappa_{reg}[j,] = kappa_{reg_{itm}}

  lambda_{hk}[j,] = lambda_{hk_{itm}}

  lambda_{reg}[j,] = lambda_{reg_{itm}}

  d_{neg}[j,] = d_{neg_{itm}}

  d_{pos}[j,] = d_{pos_{itm}}

  d_{hk}[j,] = d_{hk_{itm}}

  d_{reg}[j,] = d_{reg_{itm}}

  mu_{a}[j] = mu_{a_{itm}}

  mu_{b}[j] = mu_{b_{itm}}

  sigma_{e_{neg}[j]} = sigma_{e_{neg_{itm}}}

  sigma_{e_{phr}[j]} = sigma_{e_{phr_{itm}}}

  sigma_{d}[j] = sigma_{d_{itm}}

82
sigma2b[j] = sigma2b_itm
sigma2kappa_hk[j] = sigma2kappa_hk_itm
sigma2kappa_reg[j] = sigma2kappa_reg_itm
sigma2d_neg[j] = sigma2d_neg_itm
sigma2d_phr[j] = sigma2d_phr_itm

# get the mcmc samples of key parameters.
mcmc.samples = list(aa = aa, bb = bb, d_pos = d_pos, d_neg = d_neg,
cc = cc, mu_a = mu_a, mu_b = mu_b, phi = phi_return, kappa_reg
= kappa_reg, d_hk = d_hk, sigma2a = sigma2a, sigma2b = sigma2b,
sigma2kappa_reg = sigma2kappa_reg, sigma2kappa_hk = sigma2kappa_hk,
sigma2e_neg = sigma2e_neg, sigma2e_phr = sigma2e_phr, sigma2d_neg
= sigma2d_neg, sigma2d_phr = sigma2d_phr)

# calculate summary statistics for the parameters.
kappa_reg = matrix(colMeans(kappa_reg), nrow = n_reg)
kappa_hk = matrix(colMeans(kappa_hk), nrow = n_hk)
cc = mean(cc)
lambda = c(colMeans(lambda_hk), colMeans(lambda_reg))
mu_a = mean(mu_a)
mu_b = mean(mu_b)
phi = colMeans(phi_return)
sigma2a = stats::median(sigma2a)
sigma2b = stats::median(sigma2b)
sigma2kappa_hk = stats::median(sigma2kappa_hk)
sigma2kappa_reg = stats::median(sigma2kappa_reg)
sigma2e_neg = stats::median(sigma2e_neg)
sigma2e_phr = stats::median(sigma2e_phr)
sigma2d_neg = stats::median(sigma2d_neg)
sigma2d_phr = stats::median(sigma2d_phr)
print(proc.time() - ptm)

return(list(mu_a = mu_a, mu_b = mu_b, cc = cc, lambda = lambda, phi = phi,
kappa_hk = kappa_hk, kappa_reg = kappa_reg, sigma2a = sigma2a, sigma2b = sigma2b,
sigma2kappa_hk = sigma2kappa_hk, sigma2kappa_reg = sigma2kappa_reg, sigma2d_neg
= sigma2d_neg, sigma2d_phr = sigma2d_phr, sigma2e_neg = sigma2e_neg, sigma2e_phr
= sigma2e_phr))
}


off-target and on-target scoring algorithms and integration into the guide rna selection tool crispor. Genome Biol, 17.


immune transcripts in ffpe samples using targeted rna sequencing. Oncotarget, 8(2):3197.


