2016

IMPROVED MODELS FOR DIFFERENTIAL ANALYSIS FOR GENOMIC DATA

Hong Wang
University of Kentucky, ukytiger@gmail.com
Digital Object Identifier: https://doi.org/10.13023/ETD.2016.397

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Hong Wang, Student

Dr. Arnold J. Stromberg, Major Professor

Dr. Constance Wood, Director of Graduate Studies

IMPROVED MODELS FOR DIFFERENTIAL ANALYSIS FOR GENOMIC DATA

DISSERTATION

A dissertation submitted in partial fulfillment of

the requirements for the degree of Doctor of

Philosophy in the College of Arts and Sciences

at the University of Kentucky

By

Hong Wang

Lexington, Kentucky

Co-Directors: Dr. Arnold J. Stromberg, Professor of Statistics

and Dr. Chi Wang, Professor of Biostatistics

Lexington, Kentucky

2016

ABSTRACT OF DISSERTATION

IMPROVED MODELS FOR DIFFERENTIAL ANALYSIS FOR GENOMIC DATA

This paper intend to develop novel statistical methods to improve genomic data analysis, especially for differential analysis. We considered two different data type: NanoString nCounter data and somatic mutation data. For NanoString nCounter data, we develop a novel differential expression detection method. The method considers a generalized linear model of the negative binomial family to characterize count data and allows for multi-factor design. Data normalization is incorporated in the model framework through data normalization parameters, which are estimated from control genes embedded in the nCounter system. For somatic mutation data, we develop beta-binomial model-based approaches to identify highly or lowly mutated genes and to compare somatic mutations between patient groups. An empirical Bayes shrinkage approach is used to improve estimation of model parameters in all projects.

KEYWORDS: Differential Analysis, Empirical Bayes Shrinkage Methods , NanoString nConter, Somatic Mutation, Negative Binomial Distribution, Beta Binomial Distribution, Generalized Linear Model

Author's signature: Hong Wang

Date: October 11, 2016

IMPROVED MODELS FOR DIFFERENTIAL ANALYSIS FOR GENOMIC DATA

By
Hong Wang

Co-Director of Dissertation: Dr. Arnold J. Stromberg

Co-Director of Dissertation: Dr. Chi Wang

Director of Graduate Studies: Dr. Constance Wood

Date: October 11, 2016

ACKNOWLEDGMENTS

I would like to thank all the people who have supported and made contributions to my graduate research in the past years. First of all, I want to express my sincere gratitude to my advisor Dr. Arnold Stromberg for his tremendous support throughout my graduate study. Dr. Stromberg has been very supportive on my dissertation and provided timely and instructive comments and evaluation at every stage of the dissertation process, allowing me to complete this project on schedule. He also supported me financially during this process so I have enough time to focus on my work.

Secondly, my deepest gratitude and appreciation go to my co-advisor, Dr. Chi Wang, for patience and support during my graduate research. He has been an excellent advisor, as well as a mentor and a great friend. He is an organized, caring and patient advisor, I have been greatly improved by studying with him. For example, he taught me how to do research project step by step, hand by hand including research question analysis, model building, code writing, simulation, data validation, results interpretation, problems solving and paper writing. Under his advising, I published my first R package NanoStringDiff on the Bioconductor and published my paper on Nucleic Acids Research recently.

Next, I will express my deep appreciation and gratitude to all my complete Dissertation Committee and outside examiner: Dr.William Griffith, Dr. Ruriko Yoshida, Dr. Simon Bonner, Dr. Katherine Thompson and Dr.Hunter Moseley. Each individual provided insights that guided and challenged my thinking, substantially improving the finished product.

In addition to the technical and instrumental assistance above, I received important assistance from all my friends and classmates. It is such a wonderful things to

have you all during my PhD period. Also I would appreciate all faculty and stall in the department of Statistics for their help and support.

Last but not least, I want to thank my family, my husband Wei and my kids Austin and Dennis for their love and understanding. Also, I want to appreciate my parents for their love, support and understanding. Thank you for patiently supporting my academic pursuit and taking care of my kids for me during my PhD period. Without them, I could not have accomplished what has been done. Now, I will start new chapter of my life, I want th thank them again for what they have done in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introductions

1.1 Introduction for differential expression analysis based on NanoString nCounter Data

The advanced medium-throughput NanoString nCounter technology has been increasingly used for mRNA or miRNA differential expression (DE) studies due to its advantages. The NanoString nCounter system provides a simple and cost-effective way to profile specific nucleic acid molecules in a complex mixture. The system uses target-specific color-coded barcodes that can hybridize directly to target molecules. The expression level of a target molecule is measured by counting the number of times the barcode for that molecule is detected by a digital analyzer. The system does not need amplification and is sensitive enough to detect low abundance molecules. It can simultaneously quantify up to 800 different interesting targets, making it ideal for miRNA profiling and targeted mRNA expression analysis.

Figure 1.1 is an example of NanoString nCounter data with two groups. NanoString nCounter system provide Positive controls, Negative controls, Housekeeping controls and Endogenous. Endogenous is an interesting class of target we want to investigate, which can be a gene, an exon, or any region of interest. We use the term gene hereafter to refer to gene, exon, or region for convenience. Positive controls can be used to adjust system variation. Negative controls can be used to account for background noise, since negative controls have no expected expression. So the

1	ID			0mM-2	0mM-3	30mM-1	30mM-3
2	*Code Class*	*Name*	*Accession*				
3	Positive	POS_A(128)	nmiR00813.1	10423	14214	14535	13209
4	Positive	POS_B(32)	nmiR00809.1	2262	3225	3388	2903
5	Positive	POS_C(8)	nmiR00811.1	510	742	741	603
6	Positive	POS_D(2)	nmiR00822.1	199	263	304	227
7	Positive	POS_E(0.5)	nmiR00801.1	42	74	78	48
8	Positive	POS_F(0.125)	nmiR00825.1	15	21	31	19
9	Negative	NEG_A(0)	nmiR00810.1	6	9	14	12
10	Negative	NEG_B(0)	nmiR00828.1	15	24	17	13
11	Negative	NEG_C(0)	nmiR00803.1	5	16	14	11
12	Negative	NEG_D(0)	nmiR00823.1	7	14	14	13
13	Negative	NEG_E(0)	nmiR00827.1	11	11	14	12
14	Negative	NEG_F(0)	nmiR00805.1	8	12	7	8
15	Housekeeping	ACTB|0	NM_001101.2	170	353	275	249
16	Housekeeping	B2M|0	NM_004048.2	2489	4767	3061	2606
17	Housekeeping	GAPDH|0	NM_002046.3	1057	1920	2654	2146
18	Endogenous1	hsa-let-7a-5p|0	MIMAT0000062	5282	9813	8704	7503
19	Endogenous1	hsa-let-7b-5p|0	MIMAT0000063	1671	2831	2525	2179
20	Endogenous1	hsa-let-7c|0	MIMAT0000064	155	299	313	252
21	Endogenous1	hsa-let-7d-5p|0	MIMAT0000065	25	42	40	37
22	Endogenous1	hsa-let-7e-5p|0	MIMAT0000066	107	209	276	264
23	Endogenous1	hsa-let-7f-5p|0	MIMAT0000067	59	128	102	97
24	Endogenous1	hsa-let-7g-5p|0	MIMAT0000414	362	674	584	575

Figure 1.1: An example of NanoString nCounter Data

observed expression levels of negative controls can be treated as background noise. Housekeeping controls, also called reference genes, can be used to adjusts for the variation in the amount of input sample material. The nCounter system suggests the use of housekeeping genes, whose expressions are stable across samples, to inform this factor. NanoString provides a variety of housekeeping genes for users to choose from. Typically, at least three housekeeping genes are included in the CodeSet.

The NanoString nCounter system provides more accurate quantifications of mRNA expressions than PCR-based methods and microarrays in formalin-fixed paraffin embedded (FFPE) samples, where RNA degradation is commonly observed (Reis et al.,

2011). FFPE is a standard protocol for the long-term storage of human clinical specimens, which provide valuable disease, diagnostic and treatment information for clinical research. However, the fixation and embedding process modifies and degrades RNA, presenting challenges for differential expression analysis using FFPE samples. The quality of results from common gene expression profiling methods such as PCR-based techniques and microarrays decrease significantly for FFPE samples. However, NanoString nCounter system can provide stable and accurate results for FFPE samples offer great potential to further clinical research.

One of the fundamental tasks for molecule expression studies is to identify differential expression (DE) for mRNAs or miRNAs across experimental conditions. Most current methods for DE detection in nCounter data, such as NanoStringNorm (Waggott et al., 2012) and NanoStriDE (Brumbaugh et al., 2011), are based on t-tests. Although popular, the t-test is most appropriate for analyzing continuous, preferably normally distributed data. However, data produced by nCounter Analyzer are counts. Therefore, it is more natural to use a discrete distribution to characterize the data. Brumbaugh *et al.* (Brumbaugh et al., 2011) suggested the use of DESeq (Anders and Huber, 2010), a tool developed for RNA-seq, to analyze nCounter data because the method uses a negative binomial model for count data from RNA-seq. However, nCounter data and RNA-seq data are different, especially in data normalization, as we discuss in the following paragraph. To our knowledge, there has not been a discrete statistical model specifically designed for nCounter data.

Data normalization is a crucial step for using nCounter to quantify gene expression. The nCounter platform provides positive controls, housekeeping genes, and

negative controls to quantify lane-specific variation, differences in sample input, and non-specific background, respectively. A common data normalization procedure includes dividing the raw data by size factors and subtracting background level (Waggott et al., 2012, Brumbaugh et al., 2011). This procedure, however, spoils the discrete nature of the data and makes them ineligible to be analyzed as counts. As an alternative approach, methods developed for RNA-seq data analysis, e.g. DESeq and edgeR (Robinson and Smyth, 2007, 2008, Robinson et al., 2010), treat the size factor as a scaling parameter in the negative binomial model for data normalization. However, since those methods were developed for RNA-seq, they do not utilize positive controls and housekeeping genes when calculating the size factor. They also do not adjust for background noise, which can lead to biased quantification of gene expression, especially when the expression level is relatively low. Therefore, the results of normalization based on them may be less than optimal.

In this paper, we present a novel DE detection method specifically designed for nCounter data, which fully takes into account the discrete nature of the data and the critical need for data normalization. Our method, named NanoStringDiff, utilizes a generalized linear model (GLM) of the negative binomial family to characterize count data and allows for multi-factor design. We incorporate size factors, calculated from positive controls, housekeeping genes, and background level, obtained from negative controls, in the model framework, so that all the normalization information provided by the nCounter Analyzer is fully utilized. As demonstrated by simulations and real data analysis, our method provides more accurate and powerful results in DE detection compared to existing methods.

1.2 Introduction for differential analysis based on somatic mutation Data

Cancer arises from a clone that has accumulated the requisite somatically acquired genetic mutations. In recent years, somatic mutation profiling has become available by using the next-generation sequencing, especially the whole-exome sequencing (WES), which allows the sequencing of the protein coding portion of the genome.The WES technology has already been shown to be an advanced technology to characterize complex genomic alterations, find novel mutations, and identify potential therapeutic targets (Alexandrov et al., 2013, Ellis et al., 2012, Liang et al., 2012, Stephens et al., 2013, Stransky et al., 2011, Wang et al., 2012). According to different effect and location, somatic mutation can be classified as silent and non-silent. Non-silent mutations almost always receive more attention because they result in a change in phenotype often due to a change in the amino acid sequence of a protein. Despite the huge success of WES studies that have identified thousands of novel somatic mutations, the epidemiological and clinical interpretations of the findings are still very limited. One basic yet challenging task is to identify genes that are highly mutated compared to the background mutation rate, which are likely to be cancer-associated. The other fundamental question is how to compare somatic mutation patterns between patients with varying characteristics such as geographic region, tumor stage, or response to therapy.

Also, mutational processes are complex and heterogeneous processes. Lawrence(Lawrence et al., 2013) pointed out that failing to account for heterogeneity in mutational processes can lead to incorrect results in detecting differential mutation patterns. There

are three main types of heterogeneity associated with mutational processes. First, heterogeneity across patients in the same group. Since mutation is a stochastic process affected by a lot of factors, patients even in the same treatment group do not share identical mutation rate. Second, heterogeneity in the mutational spectrum of the tumor. The rate of mutations at CpG dinucleotides is much higher than that at other sites. Moreover, the rate of transition mutations is higher than that of transversion mutations. Published literature classify mutations into several categories, each with different mutation rates (see Chapter 3 for details). Third, regional heterogeneity across the genome. Different genes have different mutation rates.

To identify highly mutated genes, Lawrence et al proposed a new algorithm MutSigCV (Lawrence et al., 2013), which accurately accounts for mutational process heterogeneity, improving the accuracy of identifying new cancer-associated genes. But MutSigCV only can detect mutated genes with higher non-silent mutation rate as compared to its background mutation rate. However, genes with a lower non-silent mutation rate as compared to the background are also of interest. Those genes may play critical roles in cell survival and development. So mutations in these genes are not permitted. Therefore, researchers need to be very careful not targetting those genes when developing targeted cancer therapies. We have proposed a beta-binomial model-based approach to detect cancer-associated genes while fully taking into account all the complexities that are ignored by current methods. Specifically, our method, accounts for various types of variations in mutation rate and adjusts for baseline covariates. We propose an empirical Bayes shrinkage approach to estimate the dispersion parameter in the beta-binomial model and a likelihood ratio test to

identify differentially mutated genes.

In terms of comparing somatic mutation patterns between different groups, it is well know that the most frequently used statistical method for differential analysis between two groups is the Fisher's exact test (Ellis et al., 2012). For each gene, the test compares the proportions of patients with non-silent mutations in that gene between the two groups. Despite its popularity, the Fisher's exact test has several drawbacks. First, all types of mutations are treated in the same way. However, it is well known that mutation rates vary in different types of mutations. Second, it does not consider biological variation across patients from the same group. Third, it does not adjust for background mutation rate, which is the rate for silent mutations that do not significantly alter the phenotype. Fourth, it does not adjust for demographic and clinical characteristics. Patients from the two groups may differ in age, gender, and other baseline characteristics, which may confound the association between gene mutation and group assignment. Fifth, it does not adjust for sequencing coverage. A sample that has more complete coverage of a gene region tends to identify more mutations in that gene than a sample that only has partial coverage of the gene. To conclude, the Fisher's exact test oversimplifies the comparison problem and may lead to less than optimal results.

By incorporating mutational heterogeneity into the analyses, we propose a beta-binomial model-based approach to compare somatic mutation patterns between two groups while fully taking into account the complexities that are ignored by the Fishers exact test. Specifically, our proposed method accounts for various sources of variation, normalizes data based on background mutation rate, and adjusts for baseline

characteristics. As demonstrated by simulations, our method provides more accurate and powerful results in differential mutation pattern detection compared to existing methods.

Our method has been applied to identify novel somatic mutation patterns that are unique to squamous cell lung cancer (SCLC) patients in Appalachian Kentucky by comparing mutations between Appalachian and non-Appalachian samples. Appalachian Kentucky is home to the highest incidence rate of lung cancer in the United States (Lag et al., 1975). The disproportionately high incidence is not explained by tobacco alone, and it is thought that other factors such as genetic predisposition may play a key role in this disparity. It is therefore critical to understand the molecular characteristics based on evaluation of somatic genomic alterations in this population.

Chapter 2 NanoStringDiff: A Novel Statistical Method for Differential Expression Analysis Based on NanoString nCounter Data

2.1 Introduction

The advanced medium-throughput NanoString nCounter technology has been increasingly used for mRNA or miRNA differential expression (DE) studies due to its advantages, including direct measurement of RNA expression levels without amplification, digital readout, and superior applicability to formalin-fixed paraffin embedded (FFPE) samples. However, the analysis of nCounter data is hampered because most methods developed are based on t-tests, which do not fit the count data generated by the NanoString nCounter system. Furthermore, data normalization procedures of current methods are either not suitable for counts or are not specific for NanoString nCounter data. Therefor we have developed a novel DE detection method designed to specifically handle NanoString nCounter data. The method, named NanoStringDiff, uses a generalized linear model of the negative binomial family to characterize count data and allows for multifactor design. Data normalization is incorporated into the model framework through data normalization parameters, which are estimated from positive controls, negative controls, and housekeeping genes embedded in the nCounter system. We applied an empirical Bayes shrinkage approach that estimates the dispersion parameter in the model and performs a likelihood ratio test to identify differentially expressed genes. Simulations and real data analysis demonstrate that

the proposed method performs better than existing methods.

2.2 Data description

The following four real nCounter datasets were used to generate simulation studies to evaluate the performance of our proposed method.

Horbinski data: Horbinski *et al.* studied human glioma cell lines expressing GFP or GFP with IDH1 mutation (R132H) (GSE80821). The cells were grown in vitro for six days. 800 miRNAs were profiled with three replicates in each group. The data for the mutant group were used in this paper.

Mori data: Mori *et al.*(Mori et al., 2014) studied the possible reasons responsible for the widespread miRNA repression observed in cancer, global microRNA expression in mouse liver normal tissues, and liver tumors induced by deletion of Nf2 (merlin) were profiled by nCounter Mouse miRNA Expression Assays (GSE52207). Expressions of 599 miRNAs were measured with two replicates in each group. The data for the normal group were used in this paper.

Busskamp data: Busskamp *et al.*(Busskamp et al., 2014) profiled miRNAs of iPS cells (PGP1) at 0, 1, 3, and 4 days post-doxycycline induction of murine NGN1 and NGN2 using the nCounter human miRNA assay kit v1 (GSE62145). Expression of 734 miRNAs were profiled for three replicates in each group. The data from day 3 were used in this paper.

Teruel-Montoya data: Teruel-Montoya *et al.*(Teruel-Montoya et al., 2014) used the nCounter human miRNA assay kit v1 and v2 to profile miRNAs in normal human platelets, T-cells, B-cells, granulocytes and erythrocytes from 5 healthy male donors

(GSE57679). The data for B-cells were used in this paper, where expression of 730 miRNAs were profiled for five replicates in the B-cell group.

In addition, as a real data analysis, we applied our method to identify differentially expressed miRNAs between the mutant and GFP control groups for the Horbinski data.

2.3 Normalization parameters

Data generated by the nCounter system have to be normalized prior to being used to quantify gene expression and compare expression rates between different experimental conditions. Data normalization includes adjustment for sample preparation variation, background noise, and sample content variation. We introduce three normalization parameters to quantify these variations and noise, respectively. These parameters can be directly informed from the internal controls of the nCounter system.

Positive control size factor (c_i): this size factor accounts for lane-by-lane variation. The nCounter Analyzer has six spike-in positive hybridization controls with different concentrations for each sample, which can be used to infer c_i.

Background noise parameter (θ_i): this parameter quantifies the non-specific background level. The nCounter Analyzer includes six to eight negatives control probes that have no target in the sample. The observed expression levels of those negative controls characterize θ_i.

Housekeeping size factor (d_i): this size factor adjusts for the variation in the amount of input sample material. The nCounter system suggests the use of housekeeping genes, whose expressions are stable across samples, to inform this factor.

NanoString provides a variety of housekeeping genes for users to choose from. Typically, at least three housekeeping genes are included in the CodeSet.

2.4 The data model

Denote the observed count from gene g in sample i by Y_{gi}, and the unobserved expression rate by λ_{gi}. We assume a Poisson model for Y_{gi} given λ_{gi}:

$$Y_{gi}|\lambda_{gi} \sim \text{Poisson}(c_i d_i \lambda_{gi} + \theta_i).$$

Our model incorporates the positive control size factor, c_i, and the housekeeping size factor, d_i, to adjust for the sample-by-sample difference due to experimental variations. It also includes the background noise parameter, θ_i, to adjust for non-specific background. Using the additive property of the Poisson distribution, we can decompose Y_{gi} into $Z_{gi} + B_{gi}$, where $Z_{gi}|\lambda_{gi} \sim \text{Poisson}(c_i d_i \lambda_{gi})$ denotes the count from the expression of gene g and $B_{gi} \sim \text{Poisson}(\theta_i)$ denotes the background noise.

Due to biological variation, expression rates among samples from the same treatment group are not identical. This results in the over-dispersion problem, where the observed variation is larger than expected by the Poisson model. This problem is well recognized in RNA-seq experiments, where the data are also in counts (Robinson and Smyth, 2007, Anders and Huber, 2010). A common approach to address the problem is to consider a Bayesian hierarchical model, where a Gamma distribution is used to characterize the variation in the underlying expression rate:

$$\lambda_{gi}|u_{gi}, \eta_g \sim \text{Gamma}(u_{gi}, \eta_g).$$

Here the Gamma distribution is parameterized with mean u_{gi} and log dispersion η_g, where η_g is the negative of logarithm of the shape parameter in the common parameterization. Let $v_{gi} = c_i d_i u_{gi}$, then based on the hierarchical model, the marginal distribution of Z_{gi} given v_{gi} and η_g is negative binomial with mean v_{gi} and variance $v_{gi} + v_{gi}^2 \exp(\eta_g)$. Therefore, the marginal distribution of Y_{gi} is the convolution of a negative binomial distribution and a Poisson distribution.

Consider a general, multifactor experiment. Let X be the design matrix, where the number of rows is the number of samples and the number of columns is the number of covariates. The mean parameter u_{gi} is specified based on a generalized linear model with logarithmic link function:

$$\log u_{gi} = X_i \beta_g^T,$$

where X_i represents the i^{th} row of the design matrix X, which is a vector of covariates that specifies the treatment conditions applied to sample i, and β_g is a vector of regression coefficients quantifying the covariates effects for gene g. The DE analysis for experimental factor j can be performed by evaluating the hypothesis $H_0 : \beta_{gj} = 0$, where β_{gj} is the j^{th} element of β_g.

Accurate estimation of the dispersion parameter plays an important role in DE detection and shrinkage estimators have been shown to be useful in typical RNA-seq experiments when the number of replicates is small (Anders and Huber, 2010, Hard-

castle and Kelly, 2010, Robinson et al., 2010, Wu et al., 2013). Because the nCounter data is most similar to the RNA-seq data in the aspects of data discreteness and very limited number of replicates, we borrow the shrinkage method developed for RNA-seq data to estimate the dispersion parameter. Specifically, we consider an empirical Bayes shrinkage method (Wu et al., 2013), which introduces a prior distribution for the dispersion parameter and borrows information from the ensemble of genes to estimate the dispersion parameter for a specific gene. For example, we want to estimate log dispersion parameter η_1 for Gene1 in the Figure 2.1, the traditional method is only using 4 samples from Gene1 to estimate η_1, such small sample size can't provide enough information to estimate stable and accurate dispersion parameter. Empirical Bayes shrinkage method assume a prior distribution for the dispersion parameter and estimate η_1 using posterior distribution which include information from prior distribution to improve accuracy of the estimation.

The key challenge associate with empirical Bayes shrinkage method is how to choose prior distribution for the dispersion parameters. Figure 2.2 depicts the empirical distribution of the maximum likelihood estimates of gene-specific log dispersion η_g for four real datasets. Solid curves are fitted normal densities. The four NanoString nCounter datasets we considered all have limited replicates in one group, which were not sufficient to obtain a reliable maximum likelihood estimate of the dispersion parameter. Therefore, data from two groups are pooled together to increase the sample size for estimating log dispersion use maximum likelihood method. Those histograms show that η_g can be approximately modeled by a normal distribution.

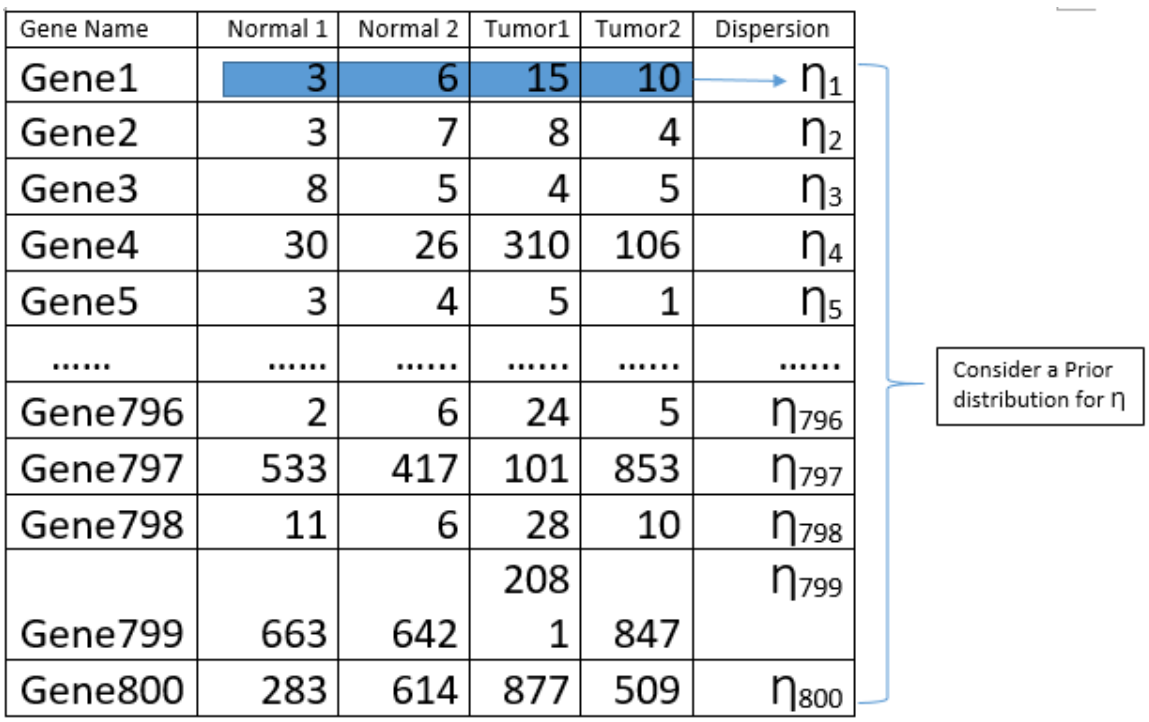

Figure 2.1: Example data set for empirical Bayes shrinkage method

Thus, we consider the following prior:

$$\eta_g \sim \text{Normal}(m_0, \tau^2),$$

where m_0 and τ^2 are hyper-parameters representing mean and variance for the normal distribution, respectively.

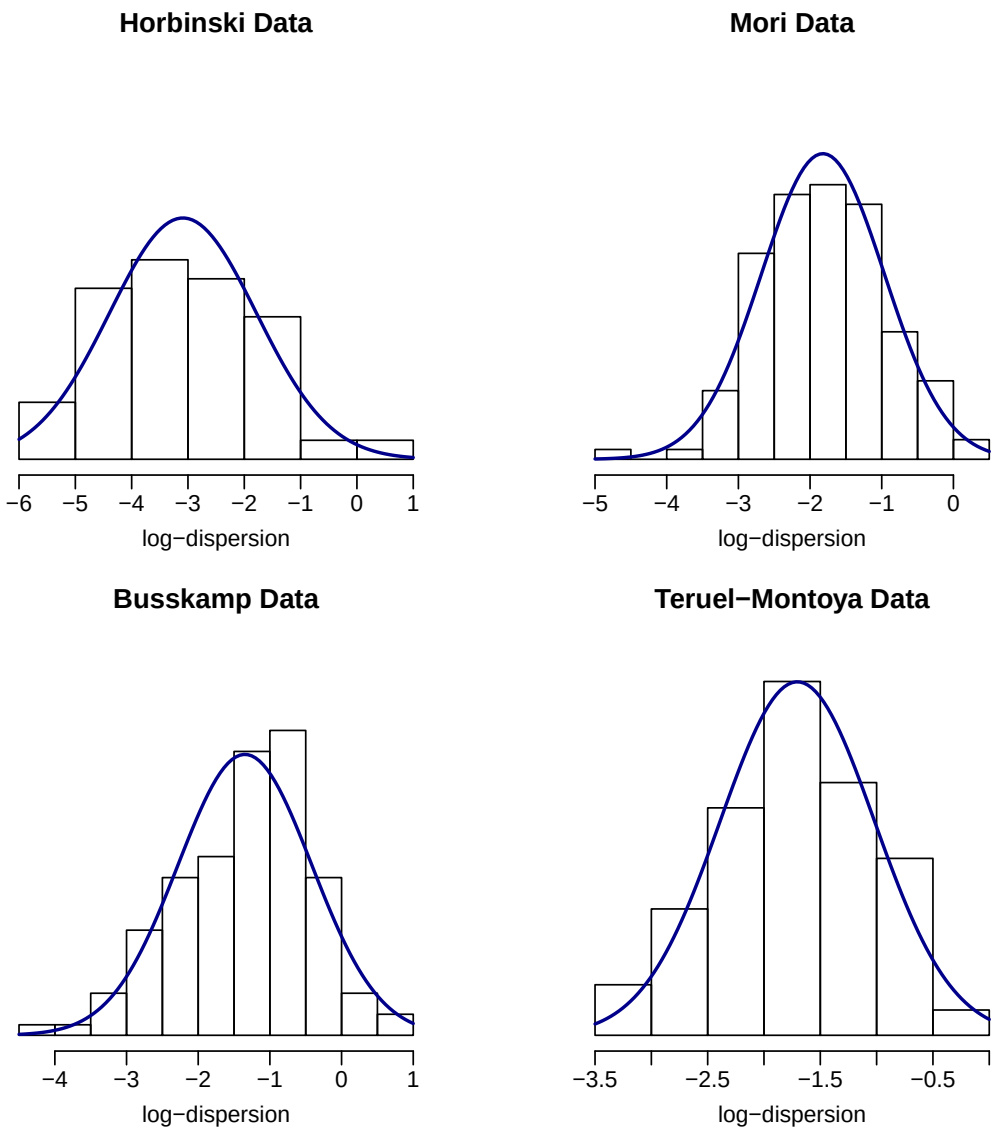

Figure 2.2: Empirical distribution of the log dispersion η_g

To sum up, the hierarchical model we consider is as follows:

$$\begin{aligned} Y_{gi}|\lambda_{gi} &\sim \text{Poisson}(c_i d_i \lambda_{gi} + \theta_i) \\ \lambda_{gi}|u_{gi}, \eta_g &\sim \text{Gamma}(u_{gi}, \eta_g) \\ \eta_g &\sim \text{Normal}(m_0, \tau^2) \\ \log u_{gi} &= X_i \beta_g^T. \end{aligned} \tag{2.1}$$

2.5 Parameter estimation and differential expression analysis

Estimating size factors and background noise

Appropriate estimation of the positive control size factor, background noise parameter and housekeeping size factor can effectively improve the accuracy of DE detection. The nCounter system suggests using the spike-in positive and negative control genes to estimate positive size factor and background noise. For each sample, nCounter provides six positive controls corresponding to six different concentrations in the 30 ul hybridzation: 128fM, 32fM, 8fM, 2fM, 0.5fM, and 0.125fM. It also provides six to eight negative controls, which can be seen as corresponding to 0fM, as no transcript is expected. We consider a Poisson model for those spike-in control genes. For each sample i, let M_{gi} denote the read count for spike-in control gene g, θ_i denote the sample-specific background noise, q_i denote the expression rate, and con_g denote the concentration for spike-in control gene g, we assume:

$$M_{gi} \sim \text{Poisson}(\theta_i + q_i \times con_g).$$

By fitting the Poisson model, we obtain the maximum likelihood estimates (MLEs) $\hat{\theta}_i$ and $\hat{q}_i$. Then the background noise parameter can be estimated by $\hat{\theta}_i$, and the positive size factor can be estimated by

$$\hat{c}_i = \frac{\hat{q}_i}{\Sigma_{i=1}^{n} \hat{q}_i / n},$$

where n is the number of samples.

The housekeeping size factor can be estimated from housekeeping genes. Because the expressions of housekeeping genes are also affected by platform source of variation and background noise, we standardize the observed read counts for housekeeping genes, H_{gi}, as follows:

$$HS_{gi} = \frac{H_{gi} - \hat{\theta}_i}{\hat{c}_i}.$$

Then we calculate the ratio of HS_{gi} for sample i relative to its average across all samples and use the median of the ratios for all housekeeping genes as the estimate of the housekeeping size factor. Mathematically,

$$\hat{d}_i = \underset{\{g:g\in\text{housekeeping genes}\}}{\text{median}} \frac{HS_{gi}}{\Sigma_{i=1}^{n} HS_{gi}/n}.$$

Estimating hyper-parameters for the prior distribution of the dispersion parameter

The hyper-parameters are empirically estimated using expression data for endogenous genes (i.e. the target genes). Specifically, for each endogenous gene, we get the MLE of the log dispersion parameter, denoted by $\hat{\eta}_g$. Because data contain background noise and endogenous genes with very low read counts cannot provide effective information, we only use $\hat{\eta}_g$ from endogenous genes with read counts larger than the the maximum value of negative controls to estimate the hyper-parameters. We use the median of $\hat{\eta}_g$ for those endogenous genes to estimate m_0, i.e. $\hat{m}_0 = \text{median}_g \hat{\eta}_g$. The estimation of τ^2 is more complex. As pointed out by Wu *et al.*(Wu et al., 2013), $var(\hat{\eta}_g) = \tau^2 + var(\hat{\eta}_g|\eta_g)$, where $var(\hat{\eta}_g|\eta_g)$ is the variation due to estimating η_g. Therefore,

the sample variance of $\hat{\eta}_g$ overestimates τ^2. Similar to Wu *et al.* Wu et al. (2013), we first use an *ad hoc* method to create some pseudo datasets with $\tau^2 = 0$ to estimate $var(\hat{\eta}_g|\eta_g)$, then subtract it from the sample variance of $\hat{\eta}_g$ to obtain an estimate of τ^2.

To be specific, we create a pseudo-dataset with $\tau^2 = 0$ by simulating Y'_{gi} from $NB(\hat{v}_{gi}, \hat{\eta}_0)$, where $\hat{\eta}_0 = \hat{m}_0$ is used as common dispersion for all genes. Then we calculate the sample log dispersion $\hat{\eta}'_g$ for each gene, we then assume $\hat{\eta}'_g$ is normally distributed and estimate $var(\hat{\eta}_g|\eta_g)$ as SE^2 using IQR method, that is

$$SE^2 = [IQR(\hat{\eta}'_g)/1.349]^2. \tag{2.2}$$

We generate a number of pseudo-datasets, and calculate the mean $\bar{SE}^2$ as the baseline. The last step of this procedure is estimate τ^2 as

$$\hat{\tau}^2 = max([IQR(\hat{\eta}_g)/1.349]^2 - \bar{SE}^2, t_0)$$

using IQR method. Where t_0 is the lower bound for $\hat{\tau}^2$. In practice, we use $t_0 = 0.01$

The details steps to derive Equation 2.2 using IQR method is as follows:

$$IQR(\hat{\eta}_g^{'}) = Q3 - Q1 = 2 * Z_{0.75} * SE$$

$$SE = IQR(\hat{\eta}_g^{'})/2 * Z_{0.75}$$

$$SE = IQR(\hat{\eta}_g^{'})/1.349$$

$$SE^2 = [IQR(\hat{\eta}_g^{'})/1.349]^2.$$

Estimating model coefficients and dispersion

Using the additive property of the Poisson distribution, we can decompose Y_{gi} into $Z_{gi} + B_{gi}$, where $Z_{gi}|\lambda_{gi} \sim \text{Poisson}(c_i d_i \lambda_{gi})$ denotes the count from the expression of gene g and $B_{gi} \sim \text{Poisson}(\theta_i)$ denotes the background noise. In order to address over dispersion problem, we use gamma distribution to describe unobserved expression rate λ_{gi}, that is $\lambda_{gi}|u_{gi}, \eta_g \sim \text{Gamma}(u_{gi}, \eta_g)$. Let $v_{gi} = c_i d_i u_{gi}$, then based on the hierarchical model, the marginal distribution of Z_{gi} given v_{gi} and η_g is negative binomial with mean v_{gi} and variance $v_{gi} + v_{gi}^2 \exp(\eta_g)$. Therefore, the marginal distribution of Y_{gi} is the convolution of a negative binomial distribution and a Poisson distribution.

Proposition 2.5.1. *Let $Z_{gi} \sim NB(v_{gi}, \eta_g)$ with probability mass function:*

$$P(Z_{gi} = j) = \frac{\gamma\{j+\exp(-\eta_g)\}}{j!\gamma\{\exp(-\eta_g)\}} \Big\{\frac{v_{gi}\exp(\eta_g)}{1+v_{gi}\exp(\eta_g)}\Big\}^j \Big\{\frac{1}{1+v_{gi}\exp(\eta_g)}\Big\}^{\exp(-\eta_g)},$$

and $B_{gi} \sim Possion(\theta_i)$ with probability mass function:

$$P(B_{gi} = Y_{gi} - j) = \frac{\theta_i^{Y_{gi}-j}\exp(-\theta_i)}{(Y_{gi}-j)!}.$$

Then, $Y_{gi} = Z_{gi} + B_{gi}$ is the convolution of a negative binomial distribution and a Poisson distribution with probability mass function:

$$\begin{aligned} p(Y_{gi}|\beta_g, \eta_g) &= \frac{\exp(-\theta_i)}{\gamma\{\exp(-\eta_g)\}}\{\frac{1}{1+v_{gi}\exp(\eta_g)}\}^{\exp(-\eta_g)} \\ &\quad \times \sum_{j=0}^{Y_{gi}} \frac{\gamma\{j+\exp(-\eta_g)\}\theta_i^{Y_{gi}-j}}{j!(Y_{gi}-j)!}\{\frac{v_{gi}\exp(\eta_g)}{1+v_{gi}\exp(\eta_g)}\}^j, \end{aligned}$$

Proof.

$$\begin{aligned} p(Y_{gi}|\beta_g, \eta_g) &= \sum_{j=0}^{Y_{gi}} P(Z_{gi}=j)P(B_{gi}=Y_{gi}-j) \\ &= \frac{\exp(-\theta_i)}{\gamma\{\exp(-\eta_g)\}}\{\frac{1}{1+v_{gi}\exp(\eta_g)}\}^{\exp(-\eta_g)} \\ &\quad \times \sum_{j=0}^{Y_{gi}} \frac{\gamma\{j+\exp(-\eta_g)\}\theta_i^{Y_{gi}-j}}{j!(Y_{gi}-j)!}\{\frac{v_{gi}\exp(\eta_g)}{1+v_{gi}\exp(\eta_g)}\}^j, \end{aligned}$$

□

We estimate β_g and η_g using an iterative procedure.

To estimate β_g, we consider its likelihood function for a given η_g:

$$L(\beta_g|\eta_g) \propto \Pi_i p(Y_g|\eta_g, \beta_g), \tag{2.3}$$

and obtain the MLE, $\hat{\beta}_g$.

To estimate η_g, we consider its posterior distribution given Y_{gi} and β_g. In order to derive posterior penalize log likelihood function for η_g, we assume a normal prior on the log dispersion η_g, e.g., $\eta_g \sim N(m_0, \tau^2)$, with density function:

$$p(\eta_g) = \frac{1}{\sqrt{2\pi\tau^2}} \exp\{-\frac{(\eta_g - m_0)^2}{2\tau^2}\}.$$

Then the conditional posterior distribution for η_g given all the observed counts and β_g is:

$$p(\eta_g | Y_{gi}, \beta_g, i = 1, ..., n) \propto p(\eta_g) \Pi_i p(Y_{gi} | \eta_g, \beta_g).$$

Therefore,

$$\begin{aligned}
& \log\{p(\eta_g | Y_{gi}, \beta_g, i = 1, ..., n)\} \\
\propto \; & \log\{p(\eta_g)\} + \sum_i \log\{p(Y_{gi} | \eta_g, \beta_g)\} \\
\propto \; & \sum_i \log \left[\sum_{j=0}^{Y_{gi}} \frac{\gamma\{j + \exp(-\eta_g)\} \theta_i^{Y_{gi} - j}}{j!(Y_{gi} - j)!} \{ \frac{v_{gi} \exp(\eta_g)}{1 + v_{gi} \exp(\eta_g)} \}^j \right] \\
- \; & n\psi\{\exp(-\eta_g)\} - \exp(-\eta_g) \sum_i \log\{1 + v_{gi} \exp(\eta_g)\} - \sum_i \theta_i \\
- \; & \frac{(\eta_g - m_0)^2}{2\tau^2} - \log \tau,
\end{aligned} \tag{2.4}$$

where $\psi(.)$ is the log gamma function. Equation (2.4) also can be viewed as a penalized log likelihood function with penalty $-\frac{(\eta_g - m_0)^2}{2\tau^2}$, penalizing values that deviate far from the common prior m_0. Estimates of size factors, background noise, and hyper-parameters are plugged into Equation (2.4) and treated as constants. For a given β_g, we obtain the estimate of η_g, denoted by $\tilde{\eta}_g$, by maximizing Equation (2.4).

We start with $\hat{\eta}_g$ as the initial value of η_g, plug it into Equation (2.3) to obtain $\hat{\beta}_g$. Then we plug $\hat{\beta}_g$ into Equation (2.4) to obtain $\tilde{\eta}_g$. By iteratively updateing $\hat{\beta}_g$ and

$\tilde{\eta}_g$ until convergence, we obtain estimates for β_g and η_g. Figure 2.3 show the work flow of this procedure.

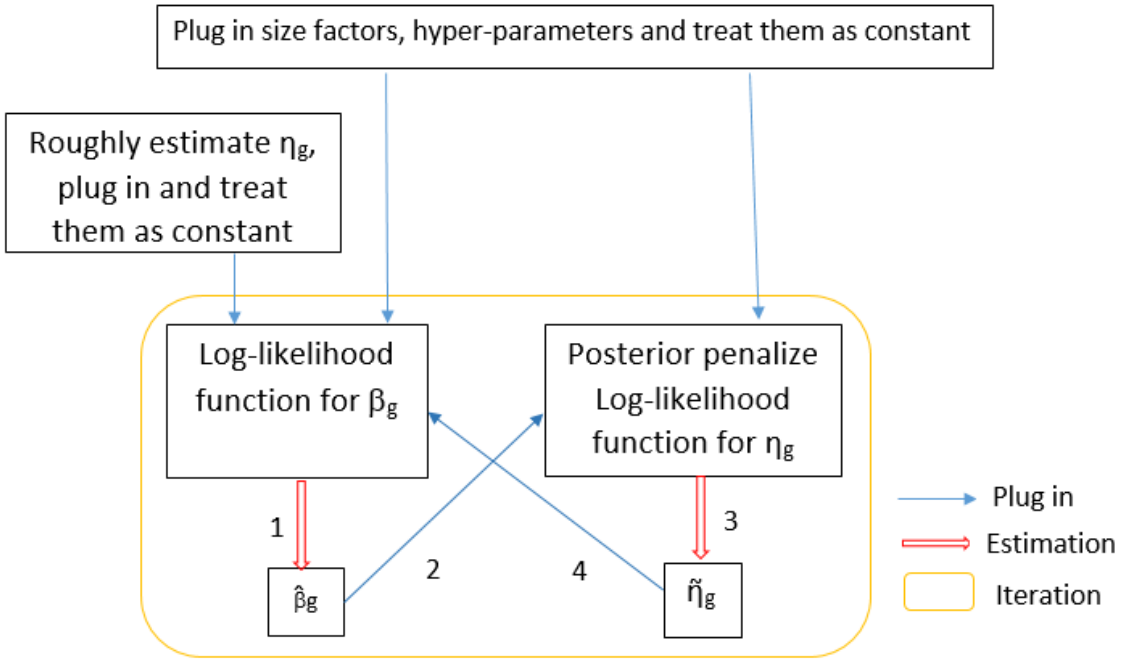

Figure 2.3: Work flow of estimating model coefficients and dispersion

Hypothesis testing and false discovery rate

We consider a likelihood ratio test for DE detection. For each gene, we compare the maximum of the log of the likelihood in Equation (2.3) under the null hypothesis versus that without any constraint. We extend our model using generalize linear model to handle more complex experimental situation,such as multi-group comparison and multi-factor experimental design. In general, user can define their specific hypothesis test based on their scientific question and experimental design. Consider simplest

setting, two group comparison without any covariate, the hypothesis test for gene g is straightforward:

$$H_0 \colon u_{g1} = u_{g2}$$

$$H_a \colon u_{g1} \neq u_{g2}.$$

For likelihood ratio multiple testing, the chi-square approximation is used to obtain a p-value. The Benjamini and Hochberg procedure(Benjamini and Hochberg, 1995) is used to calculation the false discovery rate (FDR), which provide a choice of a cutoff for statistical significance.

2.6 Q-PCR validation of miRNAs

Total RNA was extracted using TRizol reagent (Life Technologies, Grand Island, NY). For miRNA including internal control U6, the single-stranded cDNA from total RNA (20 ng) was synthesized using specific miRNA primers from the TaqMan MicroRNA Assays and reagents from the TaqMan MicroRNA Reverse Transcription (RT) Kit according to the manufacture's instruction. For β-actin and 18S internal controls, cDNA was prepared from the total RNA using the High Capacity cDNA Reverse Transcription Kit (Life Technologies). Expression of target genes was then assessed by Comparative Ct ($\Delta\Delta$Ct) using commercially available probes and TaqMan Universal PCR master mix and performed on a StepOnePlusTM 96-well instrument as described by the manufacturer (Life Technologies). The expression level of each miRNA targets

was normalized by β-actin, 18S, or U6 RNA and reported as a relative level to a specified control, as noted. The data were analyzed by two-sample t-tests.

2.7 Simulations

Simulation settings

We performed comprehensive simulation studies to evaluate the performance of our NanoStringDiff method and to compare with two other software packages that have been proposed for analyzing NanoString data (Waggott et al., 2012, Brumbaugh et al., 2011): NanoStringNorm (version 1.1.21) and DESeq2 (version 1.10.0). For NanoStringNorm, we called function NanoStringNorm with one recommended setting, that is using geometric mean to estimate positive size factor and housekeeping size factor, and using mean background value plus 2 standard deviation as background threshold. For DESeq2, we called the function DESeq with default settings. The DESeq method (Anders and Huber, 2010) was originally developed for RNA-seq data and had been suggested to analyze NanoString data by Brumbaugh *et al.* (Brumbaugh et al., 2011). Here, we considered its successor, DESeq2 (Love et al., 2014), in the methods comparison. To more generally assess the difference between NanoStringDiff and RNA-seq data analysis methods, we also compared our method to edgeR (version 3.12.0) (Robinson and Smyth, 2007, 2008, Robinson et al., 2010), which is another frequently used method for RNA-seq data analysis.

To evaluate DE detection under known truth, and to conduct the comparison under realistic scenarios encountered in nCounter experiments, data were generated

based on model (2.1) using parameters estimated from the four real datasets described above. We focused on the two-group comparison situation and simulated data based on four real nCounter datasets: Horbinski data, Mori data, Busskamp data, and Teruel-Montoya data. For the log dispersion parameter, η_g, we considered two different approaches to generate the data: (a) One approach is to use a normal distribution with distribution parameters calculated from real data. Explicitly, we obtained $\eta_g \sim N(-4.636, 1.177^2)$ from Horbinski data, $\eta_g \sim N(-5.03, 0.816^2)$ from Mori data, $\eta_g \sim N(-3.956, 0.923^2)$ from Busskamp data and $\eta_g \sim N(-2.418, 1.026^2)$ from Teruel-Montoya data. We denoted this approach as initial simulation setting. (b) The other approach is a distribution-free approach, where the log dispersion was randomly re-sampled from the log dispersions calculated using the real datasets. We considered this approach as secondary simulation setting.

For both simulation settings, expressions of 800 endogenous genes were generated from Poisson($c_i d_i \lambda_{gi} + \theta_i$), where model parameter values were set based on real data. Specifically, the normalization parameters, c_i, d_i, and θ_i, were calculated from the source real datasets. The mean parameter, λ_{gi}, was randomly re-sampled from the means calculated from the real datasets. We considered 150 endogenous genes as truly differentially expressed with the log fold change randomly generated from a mixture distribution $0.5N(1, -0.3) + 0.5N(1, 0.3)$. In our simulations, expressions of the positive and negative control genes were generated from Poisson($\theta_i + q_i \times con_g$) with θ_i and q_i calculated from the real data. Expressions of housekeeping genes were also generated from a Poisson distribution, with the Poisson mean parameter calculated from the real data. We considered 3, 5 or 8 replicates in each treatment

group and ran 100 simulations for each simulation scenario.

We first present the results under initial simulation setting for using a normal distribution to generate the dispersion parameter. The results under secondary simulation setting for using the other method to generate the dispersion parameter are provided in later section.

Simulation results under initial simulation setting

We first evaluated the estimation of size factors based on NanoStringDiff, and compared to DESeq2 and edgeR. Figure 2.4(a-c) plot the NanoString estimated positive size factor, housekeeping size factor, and sample specific background noise against their true values, respectively. NanoStringDiff provides accurate estimation of those parameters. For comparison with DESeq2 and edgeR, we define an overall size factor as the product of the positive size factor and the housekeeping size factor. The overall size factor is used in DESeq2 and edgeR, where it is estimated by using data from endogenous genes. Figure 2.4(d) plots the estimated overall size factor against the true value based on NanoStringDiff, DESeq2, and edgeR. The overall size factor estimated from NanoStringDiff has a much smaller variation compared to DESeq2 and edgeR. This is because NanoStringDiff fully utilizes the positive controls and housekeeping genes information provided by nCounter to estimate the size factor. In contrast, such information is not used by DESeq2 and edgeR.

We next evaluate the estimation of log dispersion parameters. Figure 2.5 shows the estimated versus true log dispersions for four real datasets. Our empirical Bayes shrinkage estimator tracks the real dispersion for genes with higher counts (> 20).

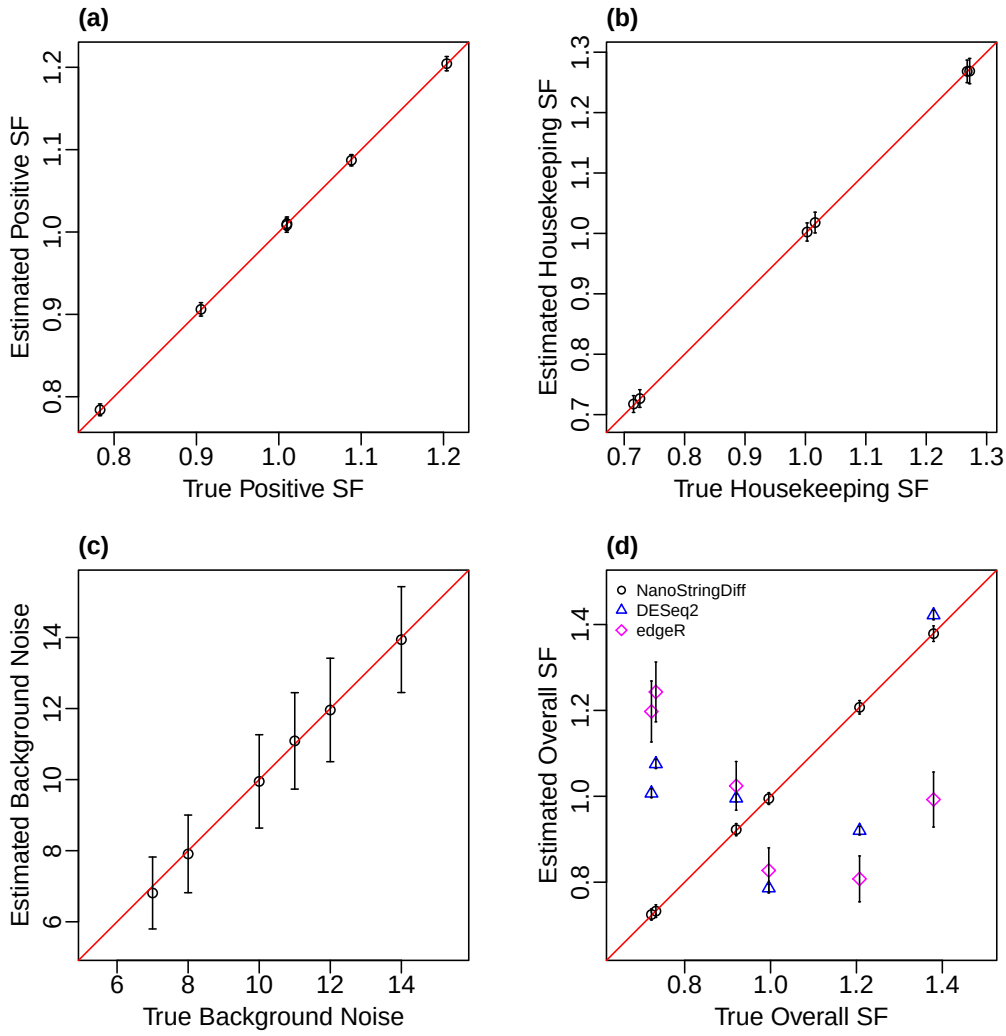

Figure 2.4: Estimation of normalization parameters. (a) Positive size factor estimated from NanoStingDiff against its true value;(b) Housekeeping size factor estimated from NanoStingDiff against its true value;(c) Background noise estimated from NanoStingDiff against its true value; (d) The overall size factor estimated from NanoStringDiff, DESeq2, and edgeR against its true value; For NanoStringDiff, the estimated overall size factor was the product of the estimated positive and housekeeping size factors; For DESeq2 and edgeR: the estimated overall size factor was directly calculated from the algorithm. Results were from a dataset simulated based on the Horbinski data with 3 replicates and averaged across the replicates.

For genes with lower counts (<= 20), the dispersion parameter estimates were shrunk more towards the average log dispersion, because there was little information about dispersion for those genes in the data. It also appeared that our algorithm worked

better when the true log dispersion was larger than −4. When the true log dispersion was very small, the estimator tended to overestimate the log dispersion, as shown from the Horbinski data and Mori data.

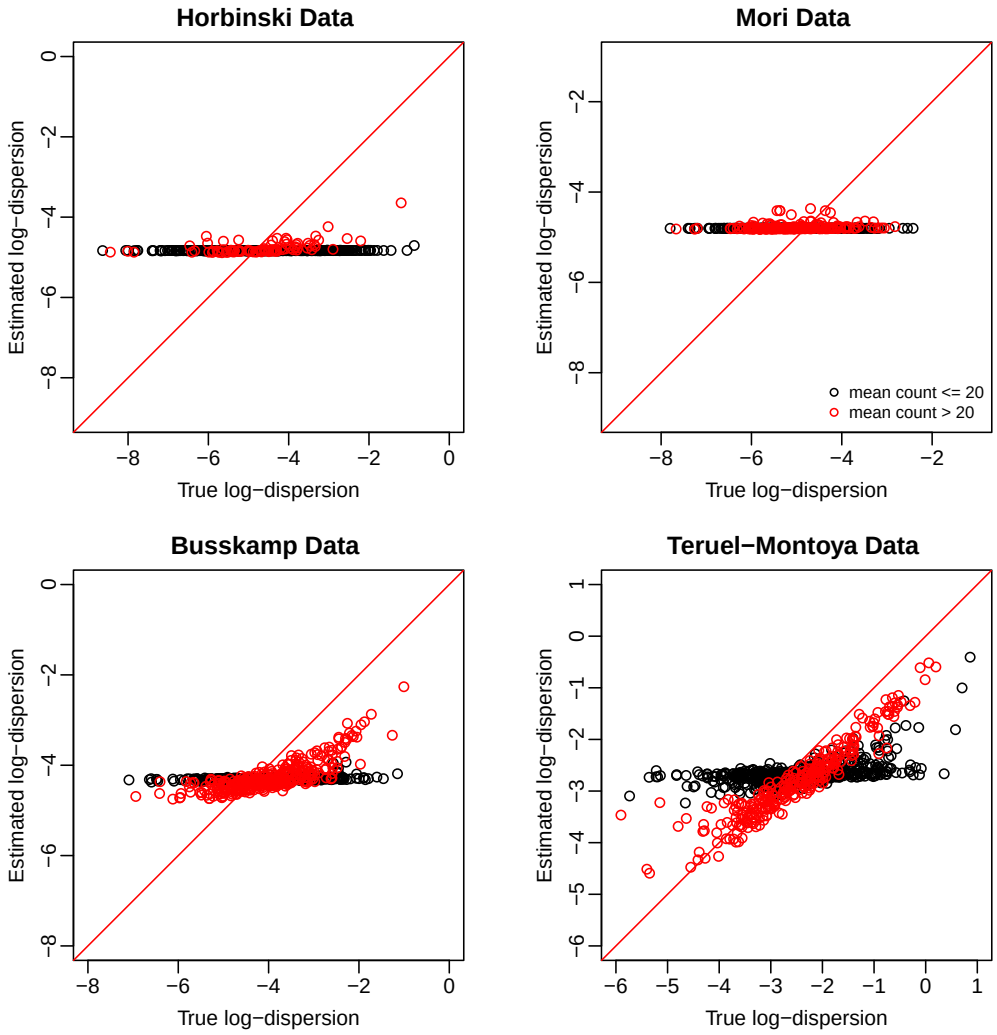

Figure 2.5: Estimation of log dispersion

We then assessed the control of type I error rate for the likelihood ratio test in NanoStringDiff. Figure 4.9 plots the reported type I error rate against the true type I error rate based on simulated data. For data simulated based on Horbinski data and Mori data, our method provided good control of the type I error rate: the reported

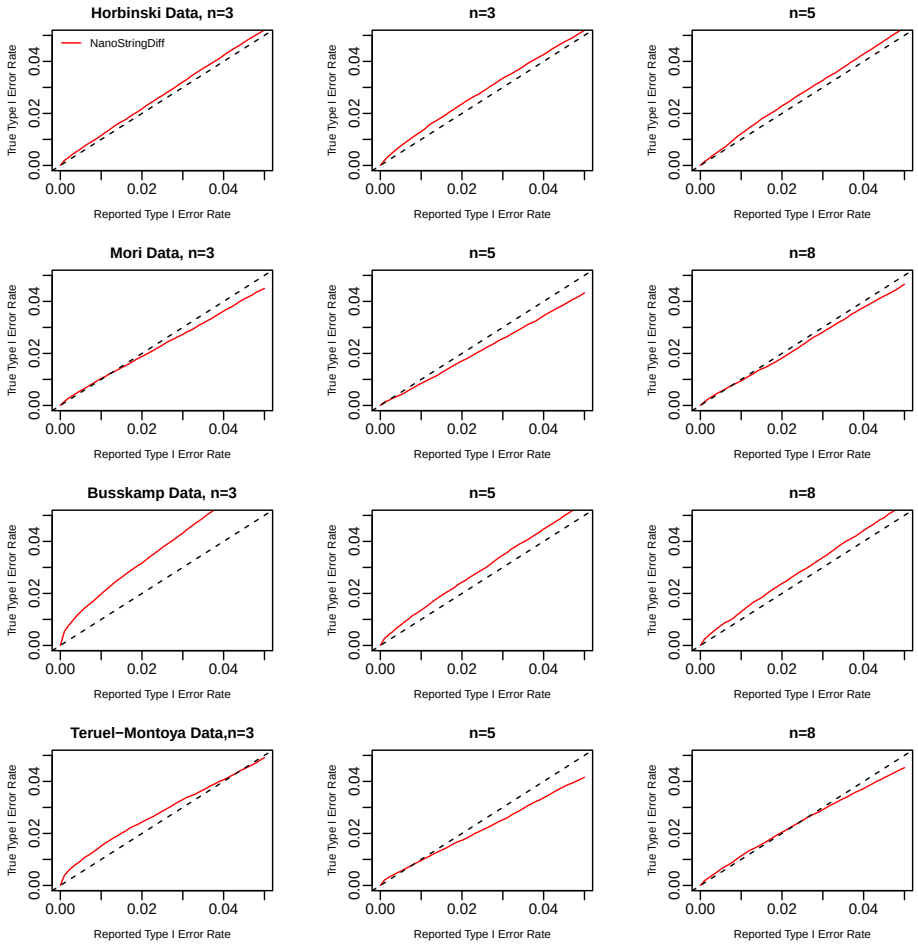

Figure 2.6: Type I error rate for data simulated based on four real datasets. For each given p-value threshold (reported type I error rate), the true type I error rate was calculated as the proportion of non-DE genes having p-values smaller than the threshold. Results were averaged over 100 simulations.

value was close to the true value. But for some other simulation scenarios, the type I error rate was inflated. To be specific, When the sample size was not very small ($n = 5$ or 8) or the biological variation was not large (for data simulated based on Horbinski data or Mori data), our method provided good control of the type I error rate, i.e. the reported value was close to the true value. However, when the sample size was very small ($n = 3$) and the biological variation was large (for data simulated

based on Busskamp data or Teruel-Montoya data), the reported type I error rate was smaller than the true value. This was due to the use of the chi-square approximation in calculating the p-value from the likelihood ratio test. Although the approximation performed well in most situations, it appeared to be inaccurate and could cause an inflated type I error rate when sample size was very small and the biological variation was large. As a result, the FDR was also inflated (Figure 2.10 Busskamp data and Teruel-Montoya data, n=3), so that our method was anti-conservative under such situation.

An important task of DE analysis is to rank genes based on their evidence of being differentially expressed. From this point of view, the ability to have as many true positives as possible in the top-ranked genes is a critical part of evaluating the performance of a method. We compared the receiver operating characteristic (ROC) curve, which shows true positive rate and false positive rate at various thresholds, among NanoStringDiff, NanoStringNorm, DESeq2, and edgeR (Figure 2.7 and Figure 2.8). Note that we only used genes having at least one average expression count after adjusting background noise because there was no classification power for genes with average count lower than background noise (area under the ROC curve (AUC) close to 0.5 for all three methods, data not shown). Separate ROC curves were generated for genes with average count, after adjusting background noise, between 1 and 100 (Figure 2.7), and higher than 100 (Figure 2.8). From Figure 2.7, we found that for genes with average count between 1 and 100, ROC curves from NanoStringDiff were higher than those for DESeq2 and NanoStringNorm, indicating better performance of NanoStringDiff in providing higher true positive rates at given false positive rates.

The ROC curves from NanoStringDiff and edgeR were close to each other. From Figure 2.7, we found that for genes with average count larger than 100, as expected, the ROC curves were higher for all methods exclude Teruel-Montoya data. The difference across methods were also much smaller due to the less impact of background noise adjustment at large read counts. For Teruel-Montoya data, the ROC curves provided by three other methods still not perform good even when average count great than 100, might be due to the large biological variation associated with human data.

In practice, DE genes are often declared based on a user-specified FDR threshold. Given the threshold, a powerful method is expected to identify as many true DE genes as possible. We compared the number of DE genes identified under a given FDR threshold (0.05, 0.1, or 0.2) among different methods. As shown in Figure 2.9, NanoStringDiff detected more true DE genes than NanoStringNorm, DESeq2 and edgeR in all simulation scenarios especial when sample size is small (n=3). The colorful bars represent DE genes identified by different methods, and the shaded area represents false discoveries, that is the number of non-DE genes within positive calls.

We also investigated the control of FDR for different methods. Figure 2.10 plots the reported FDR against the true FDR for each method. The reported FDR curves from NanoStringDiff were close to true FDR curves. In contrast, the true FDR from NanoStringNorm and DESeq2 were much smaller than the reported FDR. The true FDR from edgeR was also much smaller than the reported FDR in most cases. Therefore, those methods were over conservative, which partially explains the limited number of true DE genes they could identify for a given FDR threshold.

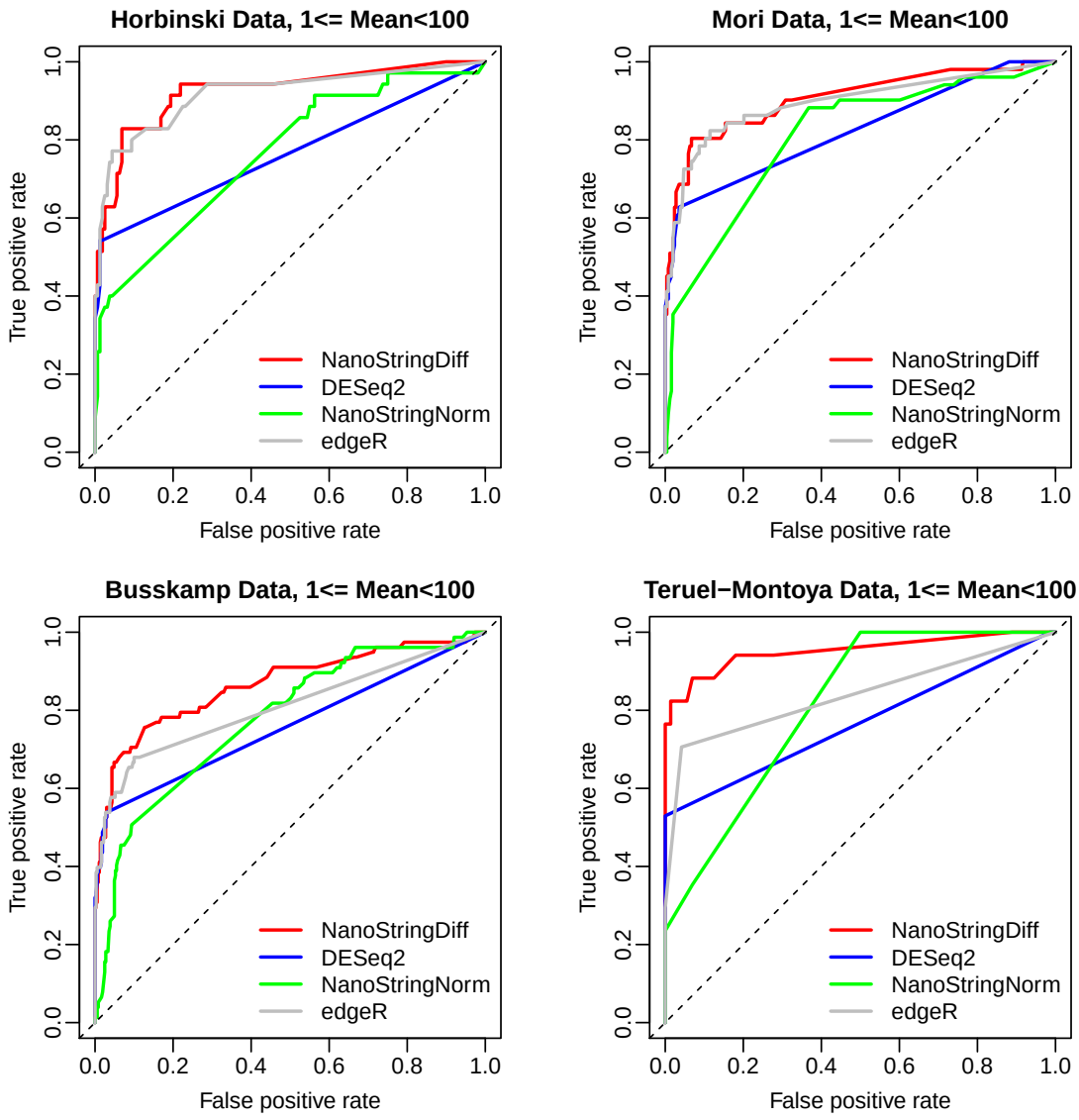

Figure 2.7: ROC curves comparing different methods. Curves were generated for genes with average read count after adjusting background noise between 1 and 100. Results were from data simulated based on four real data sets with 3 replicates.

For Busskamp data, NanoStringDiff provided good estimate of the FDR when the sample size was 5 or 8. The FDR was inflated when the sample size was 3, due to the inaccurate chi-square approximation in claculating p-values for the likelihood ratio test (Figure 4.9).

Fot Teruel-Montoya data, NanoStringDiff provided better estimate of the FDR

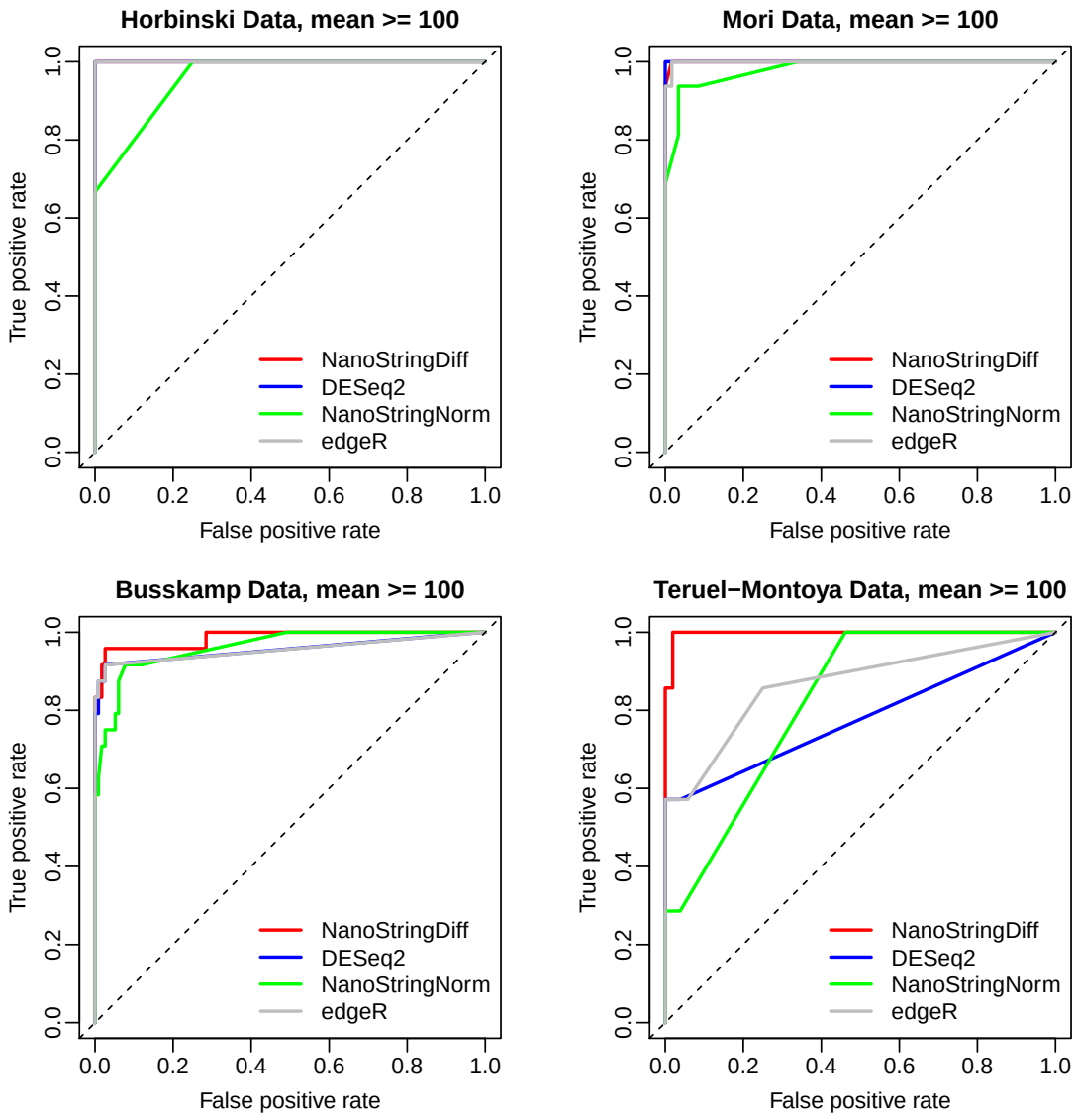

Figure 2.8: ROC curves comparing different methods. Curves were generated for genes with average read count after adjusting background noise higher than 100. Results were from data simulated based on four real data sets with 3 replicates.

compared to the other three methods. For the situation with 3 replicates, none of NanoStringDiff, NanoStringNorm and DESeq2 provided satifying FDR estimate. This is because the Teruel-Montoya data have large biological variations. Statistical inference is more difficult under such situation, especially when the sample size is very small. For NanoStringDiff, the small sample size led to inaccurate chi-square

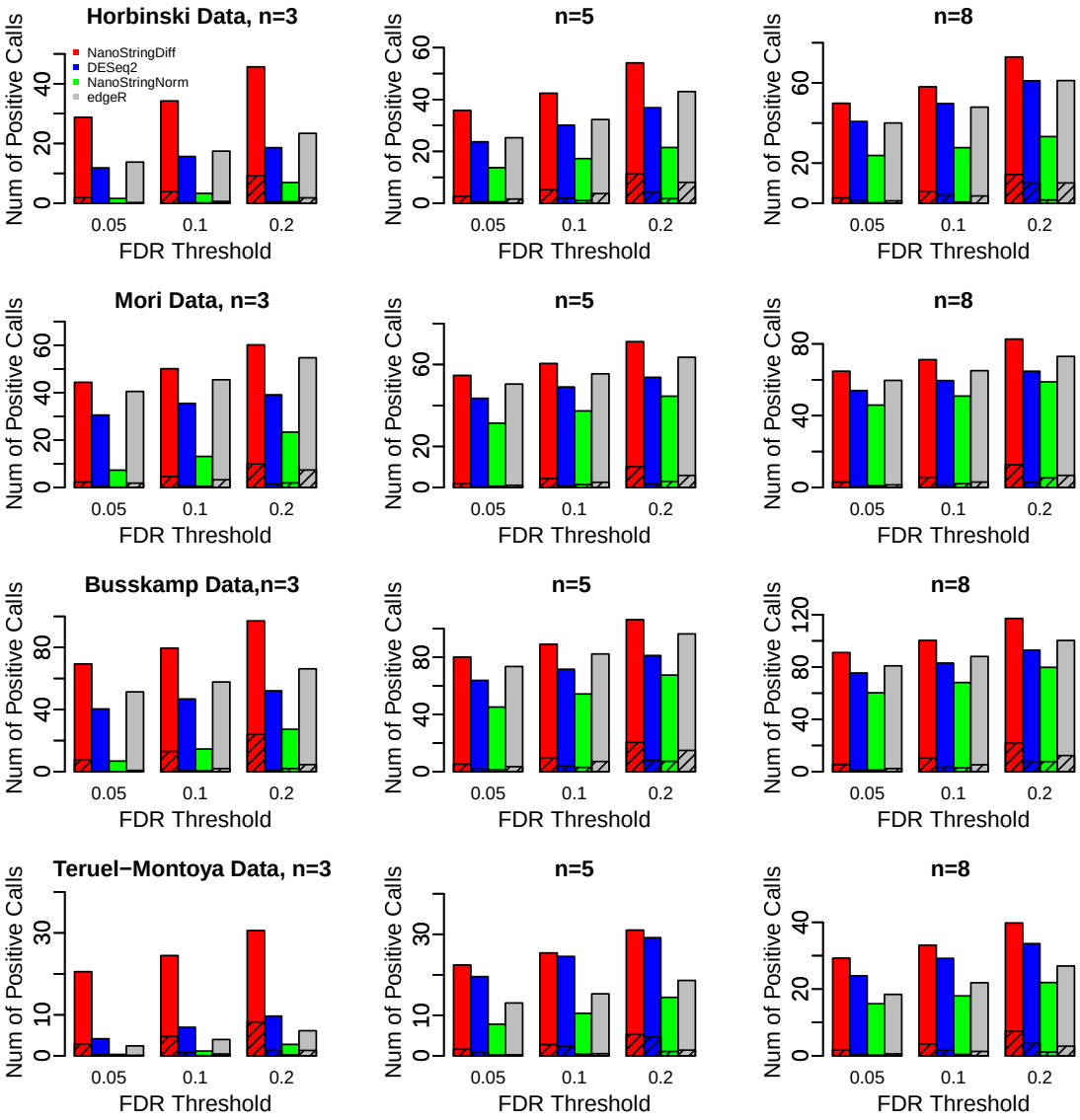

Figure 2.9: Bar charts for number of positive calls under a given FDR threshold comparing different methods. Results were averaged across 100 datasets simulated based on the four real data sets with 3, 5 or 8 replicates. For each simulation scenario, three different FDR thresholds, 0.05, 0.1 and 0.2, were considered. The shaded area represents false discoveries, i.e. the number of non-DE genes within positive calls.

approximations in claculating p-values for the likelihood ratio test (Figure 4.9). The performance of NanoStringDiff improved as the number of replicates increased. The estimated FDR from edgeR appeared to be close to the true FDR for the situation of 3 replicates. However, the FDR estimate became conservative as the number of

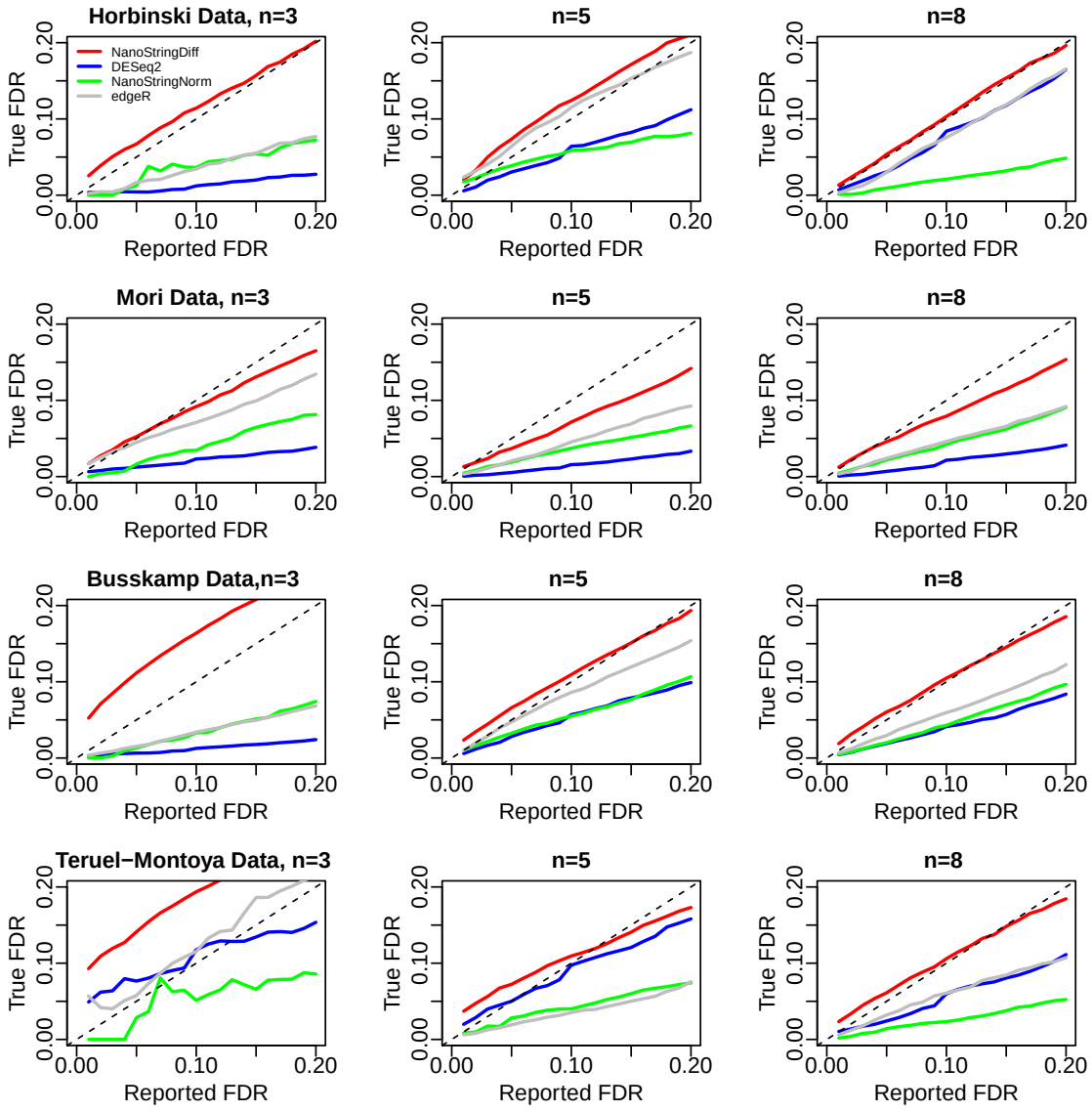

Figure 2.10: FDR estimation comparing different methods. Results were averaged across 100 datasets simulated based on the four real data sets with 3, 5 or 8 replicates.

replicates increased.

Simulation results under secondary simulation setting

To demonstrate the robustness of our method to the distributional assumption for the prior distribution of the log dispersion parameter, we considered a distribution-

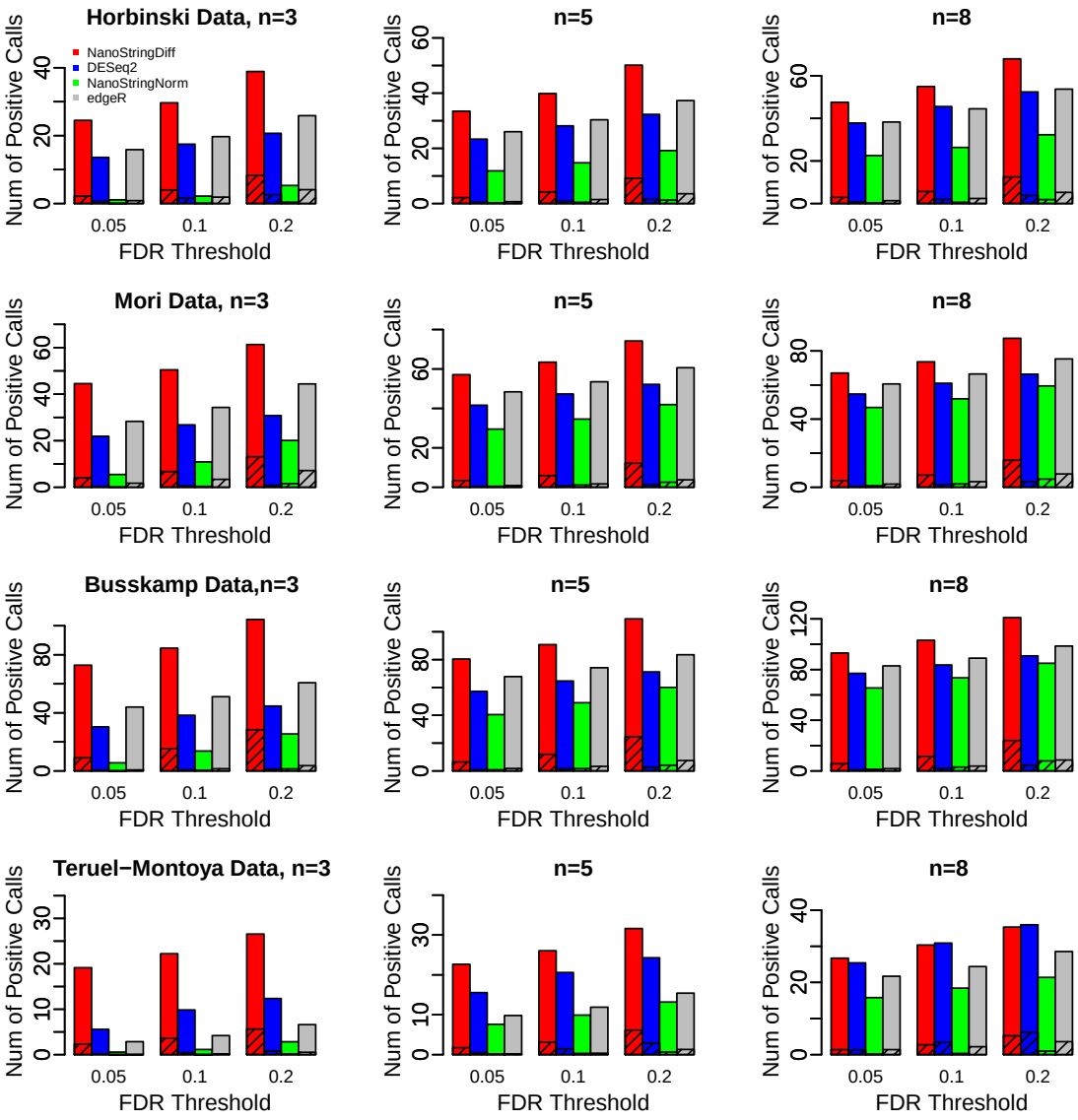

Figure 2.11: Number of positive calls for data simulated based on the four real datasets when the log dispersion parameter was generated by randomly re-sampling from the dispersions calculated from the real data

free method to generate the dispersions in our simulations, that is, the dispersion parameter was randomly re-sampled from the dispersions calculated from the real datasets. The other simulation settings remained the same. Figure 2.11 shows the number of positive callings for a given FDR threshold using data simulated from the

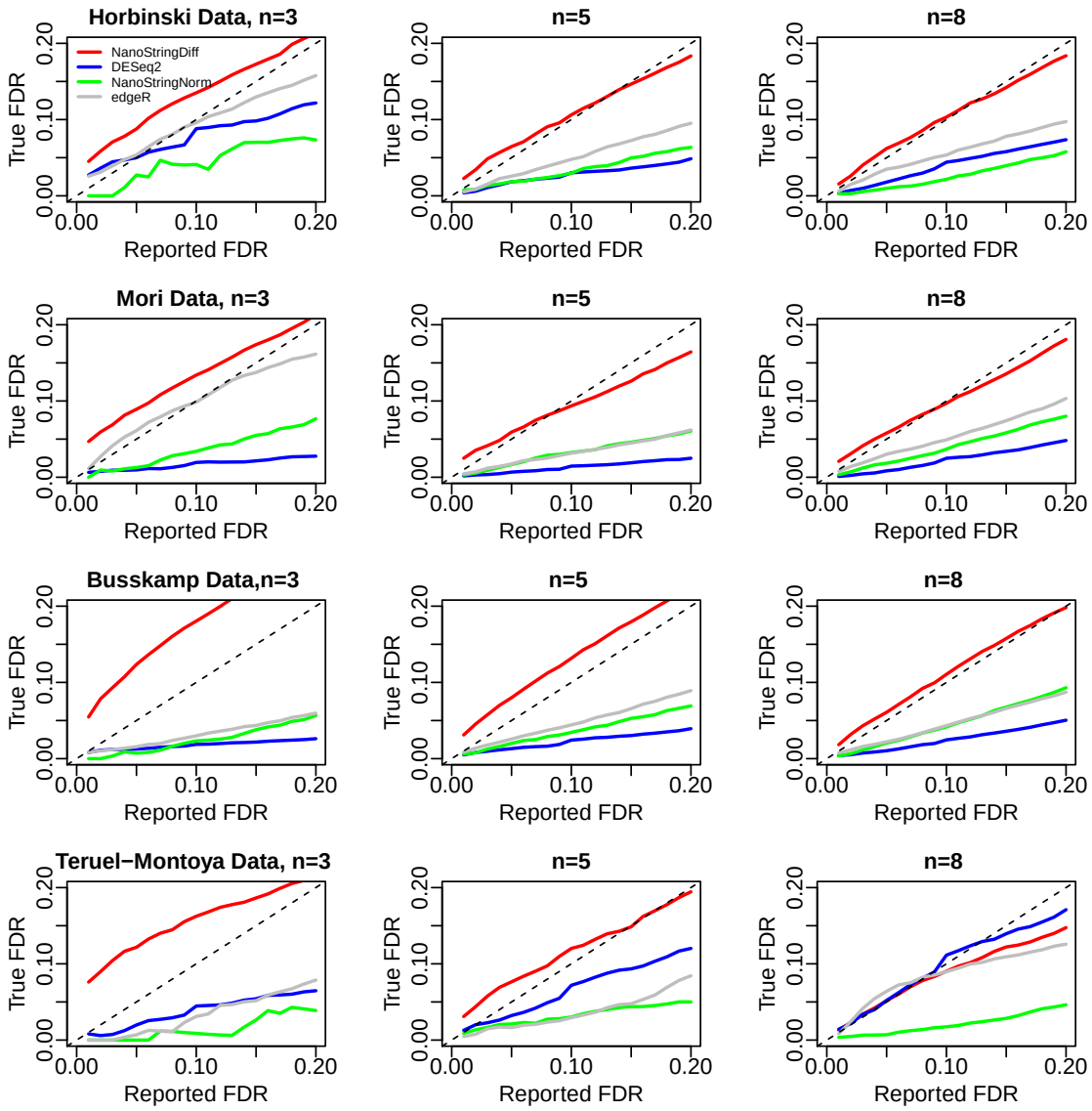

Figure 2.12: FDR estimation for data simulated based on the four real datasets when the log dispersion parameter was generated by randomly re-sampling from the dispersions calculated from the real data

four real datasets. These results have similar patterns as we presented under the original simulation settings. NanoStringDiff was able to detect more true DE genes than the other three methods. Figure 2.12 presents results about FDR estimation. Similar to the original simulation settings, NanoStringDiff showed a better estimation

of FDR than the other three methods. The reported FDR from NanoStringDiff was very close to true FDR when sample size was 5 or 8. Both Figure 2.11 and Figure 2.12 indicate that NanoStringDiff is not sensitive to the assumption on the prior distribution of the log dispersion parameter.

2.8 Real data analysis

We applied NanoStringDiff to the Horbinski data to identify miRNAs differentially expressed between IDH1 mutant and GFP control. For methods comparison, we also considered NanoStringNorm, DESeq2, edgeR, and NanoStriDE (Brumbaugh et al., 2011), which is an online application to perform DE analysis for NanoString nCounter data. NanoStriDE provided two options: DESeq and t-test. We considered both options with their default settings in our analysis. Choosing 0.01 as the FDR threshold, NanoStringDiff identified 14 DE miRNAs, which are listed in Table 1. In contrast, DESeq2 only identified 2 DE miRNAs (indicated by * in Table 1), both edgeR and NanaStriDE with the DESeq option only identified 1 DE miRNA (indicated by $^+$ in Table 1), neither NanoStringNorm nor NanoStriDE with the t-test option identified any DE miRNAs.

Almost all the identified DE miRNAs were downregulated in IDH1 mutant, which is consistent with the role of IDH1 mutation as a general suppressor of many genes via promoter hypermethylation. Many of the DE miRNAs have been previously reported to be related to glioma and/or other types of cancer. Agrawal *et al.*(Agrawal et al., 2014) showed that miR-145-5p is upregulated in hypoxic glioblastoma cells. The upregulation of miR-145-5p is associated with more advanced colorectal cancer stage

Table 2.1: Differential expression analysis results for Horbinski data. FDR threshold was chosen as 1%. The table lists the 14 DE miRNAs identified by NanoStringDiff. Two of those miRNAs (indicated by *) were also identified by DESeq2 and one of those (indicated by $^+$) was also identified by both edgeR and NanoStriDE with the DESeq option. NanoStringNorm and NanoStriDE with the t-test option did not identify any DE miRNA. The $\log_2$ fold change quantifies difference in miRNA expression comparing IDH1 mutant vs. wild type.

miRNA	$\log_2$ fold change	q-value
$hsa-miR-145-5p\|0^{*+}$	-1.843	< 0.001
$hsa-miR-374a-5p\|0$	-1.190	< 0.001
$hsa-miR-181a-5p\|0$	-1.042	< 0.001
$hsa-miR-221-3p\|0^{*}$	-1.087	< 0.001
$hsa-miR-151a-3p\|0$	-1.437	< 0.001
$hsa-miR-374b-5p\|0$	-1.191	< 0.001
$hsa-miR-152\|0$	-2.438	< 0.001
$hsa-miR-29b-3p\|0$	-1.136	< 0.001
$hsa-miR-130a-3p\|0$	-0.885	< 0.001
$hsa-miR-361-5p\|0$	-0.982	< 0.001
$hsa-miR-93-5p\|0$	-0.802	< 0.001
$hsa-miR-143-3p\|0.012$	-1.032	0.0016
$hsa-miR-23b-3p\|0$	-0.823	0.0023
$hsa-miR-142-3p\|0$	2.642	0.0060

(Slattery et al., 2014) and invasive breast cancer (Sun et al., 2014). miR-374a-5p upregulation is associated with reduced risk of dying from colorectal cancer (Slattery et al., 2014). miR-374b-5p contributes to gastric cancer cell metastasis and invasion via inhibition of RECK expression (Xie et al., 2014). miR-181a-5p is elevated in triple negative breast cancer and associates with chemoresistance (Ouyang et al., 2014). It is also upregulated in gastric cancer, with positive correlation with lymph node invasion, nerve invasion and vascular invasion (Chen et al., 2013). miR-221 is downregulated in IDH1 mutant gliomas based on The Cancer Genome Atlas (Wang et al., 2013). It promotes cell invasion and angiogenesis in human glioma cells (Zhang et al., 2012, Yang et al., 2015). The upregulation of miR-221 is associate with poor prognosis

in glioma (Zhang et al., 2012) and colon cancer (Tao et al., 2014). miR-152 was known as a tumor suppressor in glioma stem cells (Ma et al., 2014, Yao et al., 2015), and reduces glioma cell invasion and angiogenesis via MMP-3 (Zheng et al., 2013). miR-23b-3p is upregulated in hypoxic glioblastoma cells (Agrawal et al., 2014). miR-142-3p is heavily downregulated in glioblastoma-infiltrating macrophages. It induces selective apoptosis in M2 macrophages via interacting with the transforming growth factor beta receptor 1 pathway (Xu et al., 2014).

We selected the top 5 miRNA targets identified by NanoStringDiff listed in Table 1 to further confirm their DE patterns. The original total RNA samples used for NanoString were analyzed by Q-PCR. Figure 2.13 presents results using β-actin, 18S or U6 as the internal control. All five targets were validated by Q-PCR analysis. In addition, in order to explore the false negative rate of NanoStringDiff, we selected four miRNAs that were not significantly differentially expressed (non-DE) based on NanoStringDiff and performed Q-PCR analysis to further confirm their non-DE patterns. The 4 miRNAs were selected based on the following criteria: 1) large p-vlaues by NanoStringDiff; and 2) expression level > 20. Figure 2.14 presents results using β-actin, and all of the four targets were validated as non-DE by Q-PCR analysis.

We also compared the ranking of miRNAs based on different methods. We analyzed Horbinshi data to identify miRNAs differentially expressed between IDH1 mutant and GFP control using NanoStringDiff, DESeq2, edgeR, NanoStringNorm, and NanoStriDE. For NanoStriDE, we considered two available options: DESeq and t-test.

Figure 2.15 shows the intersections of the top 5 DE miRNAs identified by each

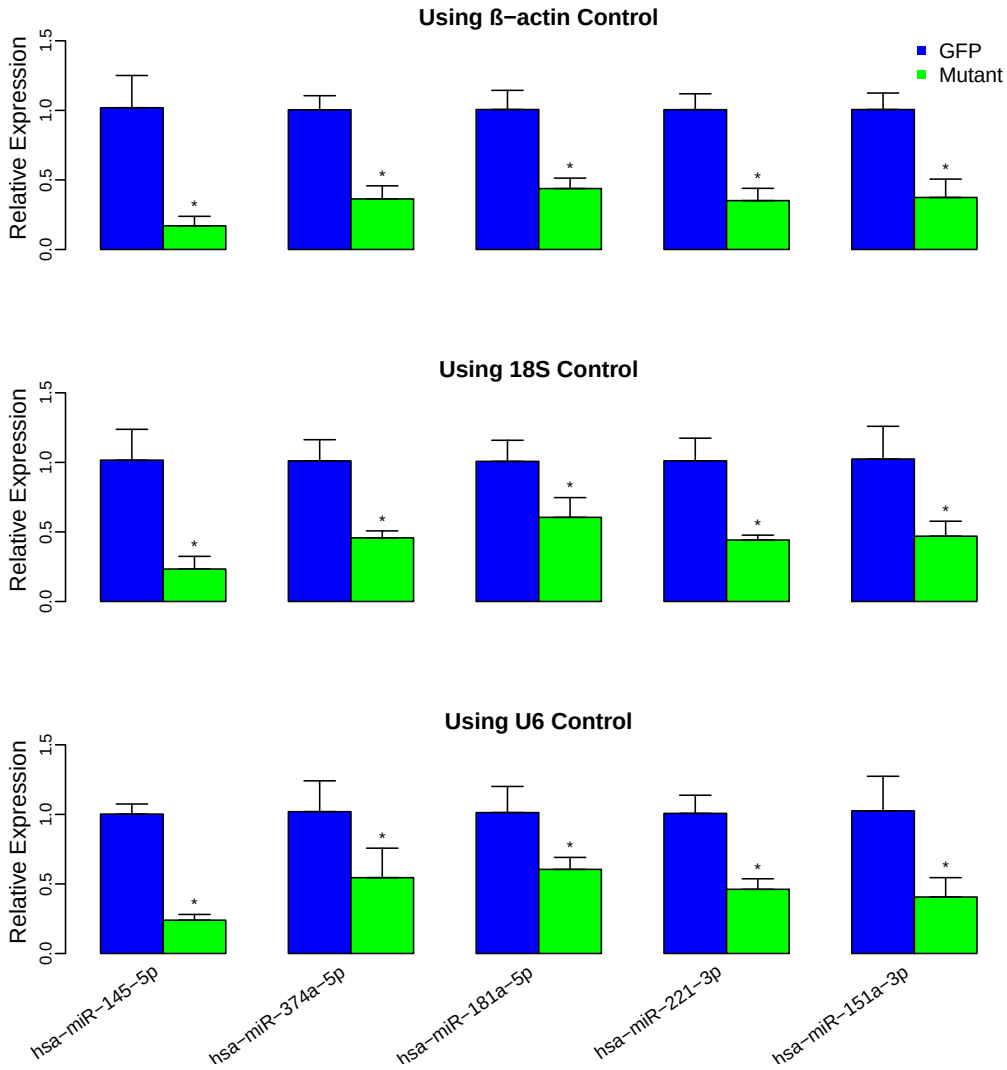

Figure 2.13: microRNA validation using Q-PCR. Total RNA used for NanoString was reversed transcribed to cDNA using specific miRNA primers from the TaqMan MicroRNA Assays and reagents from the TaqMan MicroRNA Reverse Transcription (RT) Kit. Individual miRNA expression levels were assessed by Q-PCR. Values were normalized to β-actin (top panel), 18S (middle panel), or U6 (bottom panel) as indicated and reported relative to GFP control. Experiments are depicted as the mean relative miRNA expression +/- standard deviation based on at least triplicate determinations. * indicates $P < 0.05$ based on a two-sample t-test. Q-PCR analysis performed by Dr. Min Chen's lab

method. The top 5 miRNAs ranked by NanoStringDiff, which were validated by Q-PCR (Figure 2.13), were not all in the top 5 list for any of the other methods. Since the top 5 miRNAs ranked by NanoStringDiff had been validated by Q-PCR,

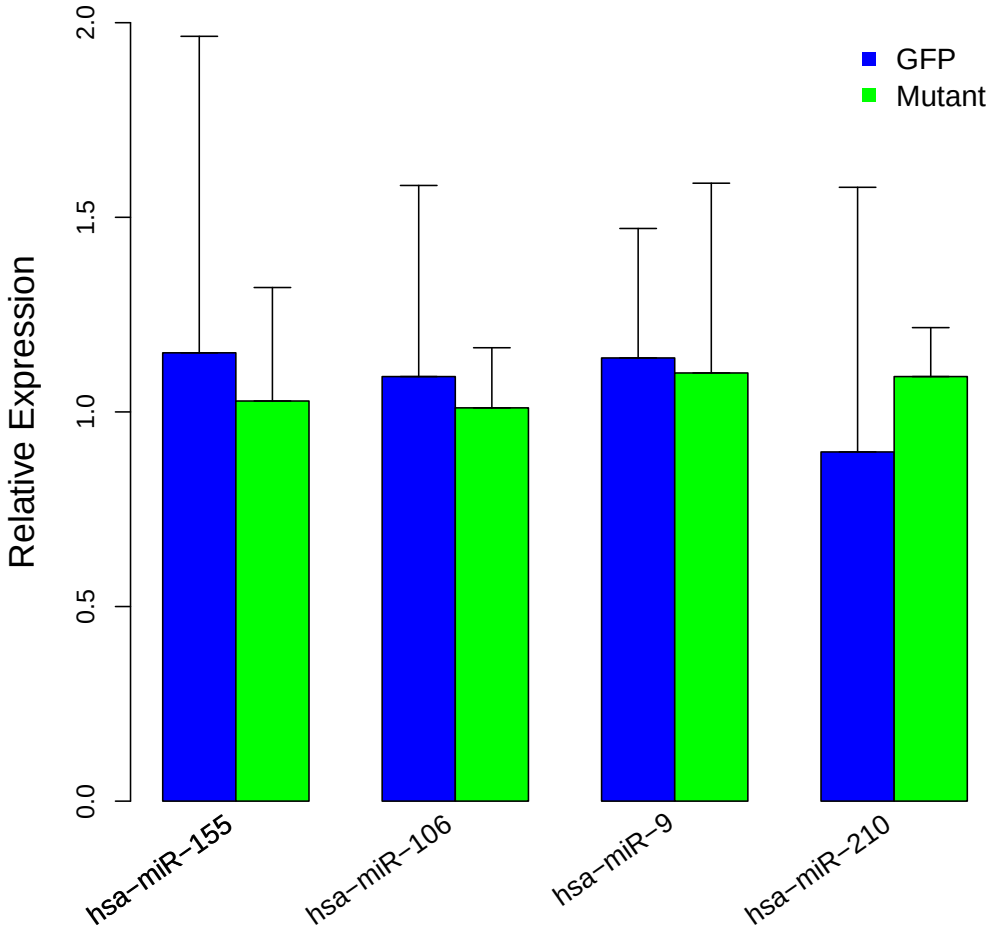

Figure 2.14: Non-DE microRNA validation using Q-PCR. Total RNA used for NanoString was reversed transcribed to cDNA using specific miRNA primers from the TaqMan MicroRNA Assays and reagents from the TaqMan MicroRNA Reverse Transcription (RT) Kit. Individual miRNA expression levels were assessed by Q-PCR. Values were normalized to β-actin and reported relative to GFP control. Experiments are depicted as the mean relative miRNA expression +/- standard deviation based on at least triplicate determinations. All of the 4 miRNAs were non-DE based on two-sample t-tests. Q-PCR analysis performed by Dr. Min Chen's lab

they were considered as true positives. For those 5 validated DE miRNAs, DESeq2 had 4 of them, NanoStriDE with the DESeq option had 3 of them, edgeR had 1 of them, and NanoStringNorm and NanoStriDE with the t-test option did not have any

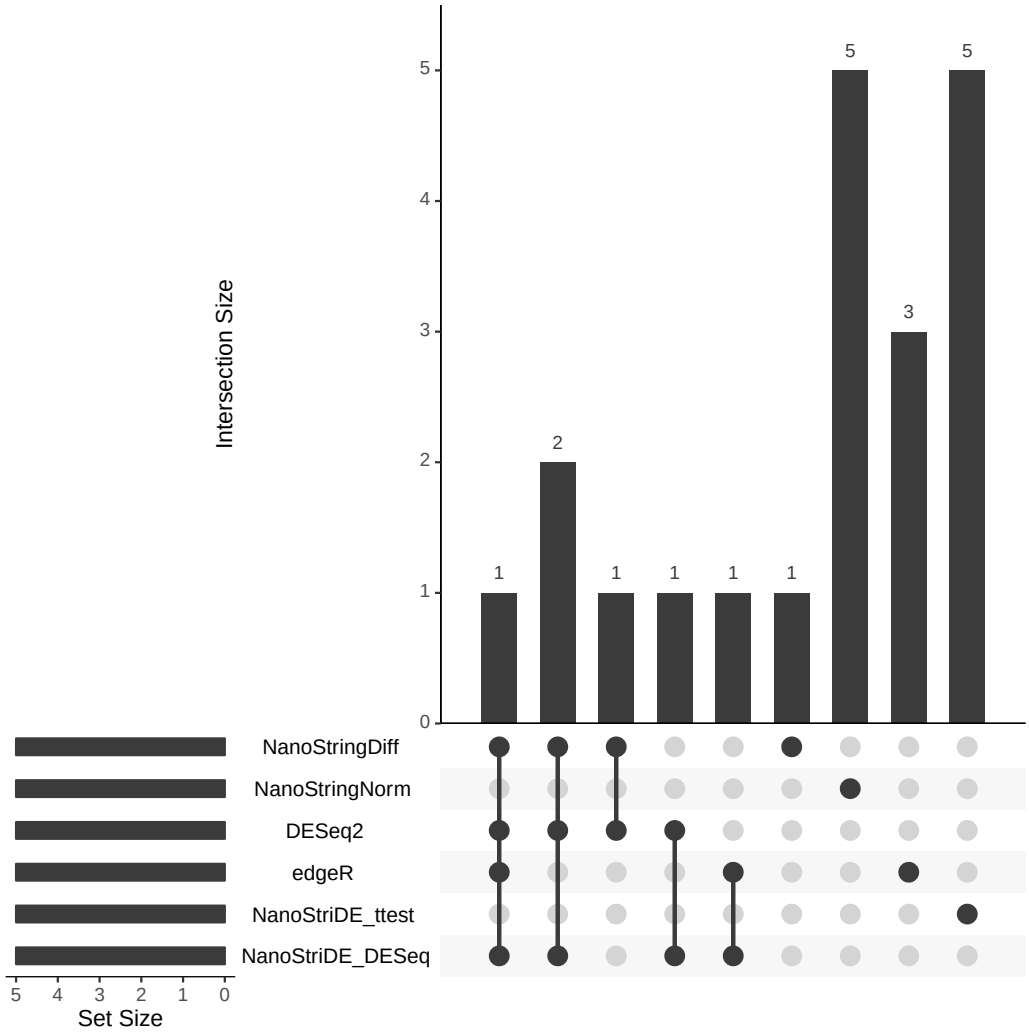

Figure 2.15: Intersections of top 5 ranked miRNAs based on NanoStringDiff, DESeq2, edgeR, NanoStringNorm, and NanoStriDE for the analysis of Horbinski data. The figures were generated by using the upset package in R.

of them in their top 5 lists.

We also compared the top 20 ranked miRNAs from each method. The intersections are presented in the Figure 2.16. None of the miRNAs was commonly identified by all methods and each method had several miRNAs that were only identified by that method alone. As the miRNAs rankings varied from one method to another, different methods had different abilities in selecting most promising candidate miRNAs for

further testing. Therefore, there were large variations in ranking miRNAs for the methods we compared.

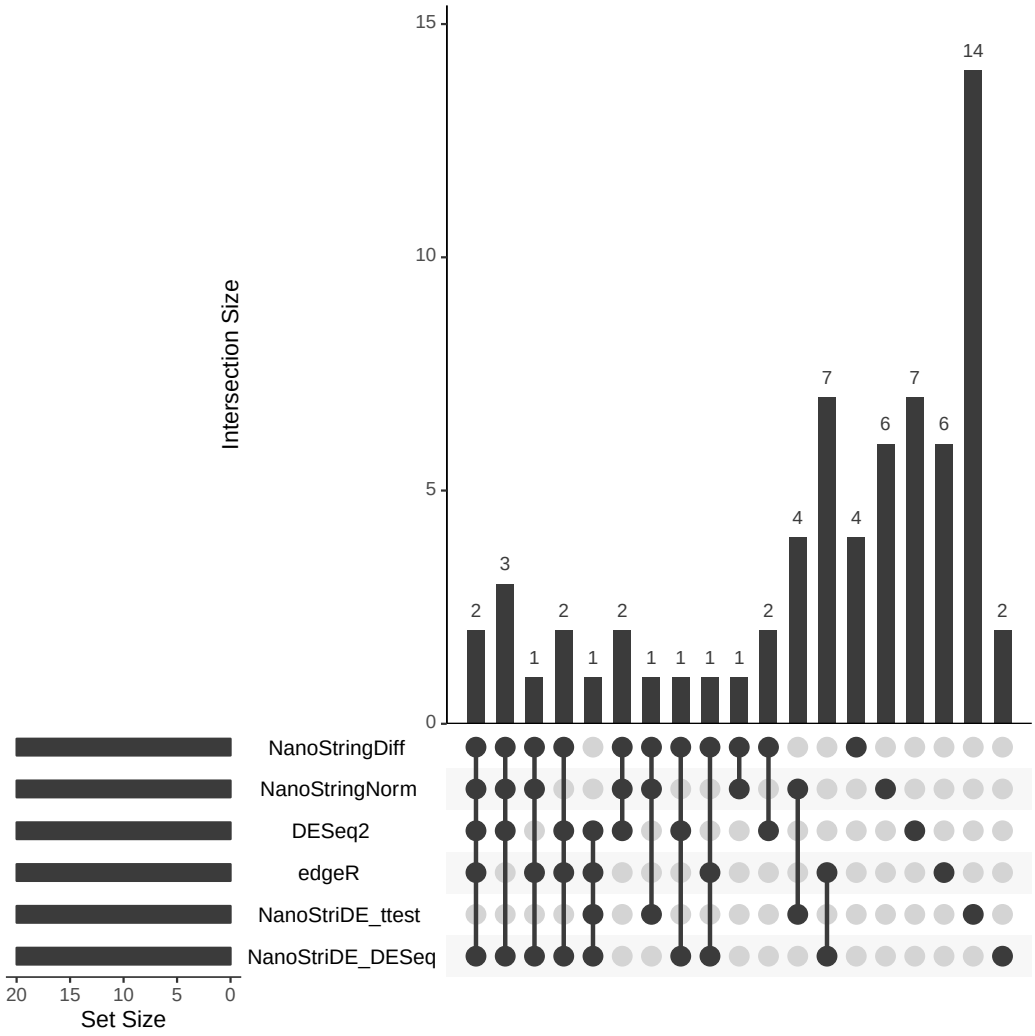

Figure 2.16: Intersections of top 20 ranked miRNAs based on NanoStringDiff, DESeq2, edgeR, NanoStringNorm, and NanoStriDE for the analysis of Horbinski data. The figures were generated by using the upset package in R.

2.9 NanoStringDiff

NanoStringDiff is an R package, which is designed for differential analysis based on NanoString nCounter data. NanoStringDiff consider a generalized linear model of

the negative binomial family to characterize count data and allows for multi-factor design. Data normalization is incorporated in the model framework by including data normalization parameters estimated from positive controls, negative controls and housekeeping genes embedded in the nCounter system. The present method use an empirical Bayes shrinkage approach to estimate the dispersion parameter and a likelihood ratio test to identify differential expression genes.

2.10 Availability

The proposed methods are implemented in an open source R package NanoStringDiff, which is available at Bioconductor. The code for performing all the analyses in this paper is available at http://sweb.uky.edu/∼cwa236/NanoStringDiff/.

The NanoString nCounter data, referred to as the Horbinski data, are available at Gene Expression Omnibus under accession number GSE80821.

Chapter 3 A Beta-Binomial Model to Identify Genes with Altered Mutation Rate in Cancer

3.1 Introduction

Cancer arises from somatically acquired genetic and epigenetic alterations. While large consortia like The Cancer Genome Atlas (TCGA) (Collins and Barker, 2007) and the International Cancer Genome Consortium (ICGC) (Hudson et al., 2010) have profiled genomic somatic mutations of thousands of tumor samples from various cancer types based on whole-genome/exome sequencing (Network et al., 2015, McLendon et al., 2008, Network et al., 2012, 2014), meaningful mechanistic interpretation of these gene variation results are still very limited. One basic yet challenging task is to distinguish driver mutations, which are causally implicated in cancer development, from passenger mutations, which occur randomly with neutral effect. Tumor genomes contain from tens to thousands of somatic mutations. However, the general understanding in the field is that only a few of them are driver mutations. One approach of finding driver mutations/genes is to identify genes that have different non-silent mutation rate compare to silent mutation rate. Lawrence et al proposed a new cancer-associated gene search algorithm MutSigCV (Lawrence et al., 2013). MutSigCV accurate accounting of mutational processes improve the accuracy of identification of new cancer-associated genes.

One limitation of the MutSigCV is that it only can detect mutated genes with

higher non-silent mutation rate compare to it's background mutation rate. However, some mutated genes with lower non-silent mutation rate also play important biological role. We developed a beta-binomial model-based approach to detect cancer-associated genes while fully taking into account all the complexities that are ignored by current methods. Specifically, our method, accounts for various types of variations in mutation rate and adjusts for baseline covariates. We propose an empirical Bayes shrinkage approach to estimate the dispersion parameter in the beta-binomial model and a likelihood ratio test to identify differentially mutated genes.

3.2 Mutation Data Structure

We propose a novel statistical model to characterize mutation rates while taking into account heterogeneity in mutational process. We classify mutations into three types: 1) non-silent; 2) silent and 3) non-coding mutation in the surrounding regions. We add silent mutation and non-coding mutation together and treat the sum as background mutation. Our focus of interest is the mutation rate of non-silent mutations after adjusting for the background (silent and non-coding) mutation rate. We also follow the approach of Lawrence et al. (Lawrence et al., 2013) to classify mutations into seven categories and assume a separate mutation rate for each category: (i) transition mutations at CpG dinucleotides; (ii) transversion mutations at CpG dinucleotides; (iii) transition mutations at C:G basepairs not in CpG dinucleotides; (iv) transversion mutations at C:G basepairs not in CpG dinucleotides; (v) transition mutations at A:T basepairs; (vi)transversion mutations at A:T basepairs; and (vii) small insertions/deletions, nonsense and splice site mutations.

In model building and parameters estimation section, we organized data in the gene-category level. In the Hypothesis testing section, we consider data in gene level, to detect mutated gene. Figure 3.1 shows a small part of mutation data arranged in the gene-category level. Figure 3.2 shows the coverage data corresponding to Figure 3.1.

gene-cateory	genename	category	sample1	sample2	sample3	sample4	sample5	sample6	sample7	sample8
A1BG-1	A1BG	1	0	0	0	0	0	0	0	0
A1BG-2	A1BG	2	0	0	0	0	0	0	0	0
A1BG-3	A1BG	3	0	0	0	0	0	0	1	0
A1BG-4	A1BG	4	0	0	0	0	0	0	1	0
A1BG-5	A1BG	5	0	0	0	0	0	0	0	0
A1BG-6	A1BG	6	0	0	0	0	0	0	0	0
A1BG-7	A1BG	7	0	0	0	0	0	0	0	0
A1CF-1	A1CF	1	0	0	1	0	0	0	0	0
A1CF-2	A1CF	2	0	0	1	0	0	0	0	0
A1CF-3	A1CF	3	0	0	0	0	0	0	0	0
A1CF-4	A1CF	4	0	0	0	0	0	0	0	0
A1CF-5	A1CF	5	0	0	0	0	0	0	0	0
A1CF-6	A1CF	6	0	0	0	0	0	0	0	0
A1CF-7	A1CF	7	0	0	0	0	0	0	0	0
A2M-1	A2M	1	0	0	0	0	0	0	0	0
A2M-2	A2M	2	0	0	0	1	0	0	0	0
A2M-3	A2M	3	0	0	0	0	0	0	0	0
A2M-4	A2M	4	0	0	0	0	0	0	0	0
A2M-5	A2M	5	0	0	0	0	0	0	0	0
A2M-6	A2M	6	0	0	0	0	0	1	0	0
A2M-7	A2M	7	0	0	0	0	0	0	0	0
A2ML1-1	A2ML1	1	0	1	0	0	0	0	0	0
A2ML1-2	A2ML1	2	0	0	0	0	0	0	0	0
A2ML1-3	A2ML1	3	0	0	0	0	0	0	0	0
A2ML1-4	A2ML1	4	0	0	2	0	0	0	0	0
A2ML1-5	A2ML1	5	0	0	0	0	0	0	0	0
A2ML1-6	A2ML1	6	0	0	0	0	0	0	0	0

Figure 3.1: Example of mutation data at gene-category level

3.3 Data description

The following two real somatic mutation datasets were used to perform real data analysis and generate simulation studies to evaluate the performance of our proposed method in the next two chapters.

gene-cateory	genename	category	sample1	sample2	sample3	sample4	sample5	sample6	sample7	sample8
A1BG-1	A1BG	1	23	25	26	26	23	24	24	25
A1BG-2	A1BG	2	36	41	42	42	37	37	39	41
A1BG-3	A1BG	3	110	114	116	115	113	112	114	115
A1BG-4	A1BG	4	112	115	118	116	116	114	116	117
A1BG-5	A1BG	5	29	28	29	29	29	29	29	29
A1BG-6	A1BG	6	37	36	38	37	37	38	38	38
A1BG-7	A1BG	7	351	363	371	368	359	357	362	368
A1CF-1	A1CF	1	5	5	5	5	5	5	5	5
A1CF-2	A1CF	2	12	12	12	12	12	12	12	12
A1CF-3	A1CF	3	71	71	73	73	72	73	73	73
A1CF-4	A1CF	4	72	72	72	73	72	72	73	72
A1CF-5	A1CF	5	114	115	115	116	115	114	116	115
A1CF-6	A1CF	6	120	121	121	121	121	120	121	121
A1CF-7	A1CF	7	397	400	402	404	402	399	404	402
A2M-1	A2M	1	15	15	15	15	15	15	15	15
A2M-2	A2M	2	23	23	23	23	23	23	23	23
A2M-3	A2M	3	247	247	248	247	248	248	248	248
A2M-4	A2M	4	230	230	232	230	232	232	232	231
A2M-5	A2M	5	223	222	224	223	224	224	224	223
A2M-6	A2M	6	253	252	254	252	254	254	254	252
A2M-7	A2M	7	995	992	999	994	999	999	999	995
A2ML1-1	A2ML1	1	17	17	17	17	17	17	17	17
A2ML1-2	A2ML1	2	31	32	31	32	31	32	32	32
A2ML1-3	A2ML1	3	253	252	253	252	250	253	253	253
A2ML1-4	A2ML1	4	218	216	217	215	215	218	218	217
A2ML1-5	A2ML1	5	212	210	211	210	208	212	212	211
A2ML1-6	A2ML1	6	235	235	234	234	231	236	236	235

Figure 3.2: Example of corresponding coverage data at gene-category level

LUSC data set (Network et al., 2012): tumor samples were obtained from 178 patients with previously untreated stage IIV squamous cell lung cancer. Germline DNA was obtained from adjacent, histologically normal tissues resected at the time of surgery (n = 137) or from peripheral blood (n = 41). DNA and RNA were extracted from patient specimens and measured by several genomic assays, which included standard quality-control assessments.

LUAK data set: Appalachian Kentucky is home to the highest incidence rate of lung cancer in the United States (108.48 per 100,000 (Collins and Barker, 2007) compared to 55.50 per 100,000 nationally (Hudson et al., 2010)). In this Genomics Research in Lung Cancer in Appalachia Study, Drs. Susanne Arnold and Chi Wang

Table 3.1: Percentage for mutation counts based on LUAK data set

Number of mutation	Non-Silent Mutation	Background Mutation
0	90.8%	74.8%
1	7.9%	17.6%
>= 2	1.3%	7.6%

Table 3.2: Percentage for mutation counts based on LUSC data set

Number of mutation	Non-Silent Mutation	Background Mutation
0	73.5%	65.4%
1	18.4%	19.3%
>= 2	8.1%	15.3%

at UK performed whole exome sequencing on matched tumor and normal tissue pairs from 51 lung squamous cell carcinoma patients in Appalachian. Data anlaysis showed that the number of somatic mutations identified varies dramatically across patients. A patient could have as few as 10 mutations or as many as 1542 mutations. Similar phenomenon was also observed from LUSC data.

Table 3.1 and Table 3.2 showed the component of the mutation data, we found that mutation data is very sparse, less than 10% genes have more than 2 mutation counts across all samples for non-silent mutation data. Extreme sparsity is a special and major property for the mutation data, which presents challenge in model building and data analysis.

3.4 The data model

Let Y_{gci} and π_{gci} be the number and rate of mutations for gene g in category c from patient i, respectively. The number of mutations is a binomial random variable given mutation rate:

$$Y_{gci}|\pi_{gci} \sim Binomial(N_{gci}, \pi_{gci})$$

where N_{gci} is the number of possible sites for this mutation type and category that have sufficient coverage in the sequencing data. Note background mutation and non-silent mutation follow the same date model.

Due to biological variation, the mutation rates among patients from the same treatment group are not identical. This results in the over-dispersion problem, where the observed variation is larger than expected by the Binomial model. Figure 3.3 plot the estimated variance for the observed date against the variance predicted by Binomial distribution. We found for each data set that the observed variance was greater than the variance estimated if we assume the model is binomial distributed.

A common approach to address the overdispersion problem is to consider a Bayesian Hierarchical model, where a Beta distribution is used to characterize the variation in the underlying mutation rate:

$$\pi_{gci} \sim Beta(u_{gci}, f_{gc})$$

Here the Beta distribution is parameterized with mean u_{gci} and dispersion parameter f_{gc}. We assume all samples for one specific gene-category share the same dispersion parameter. In common parameterization, $\alpha = u * \frac{1-f}{f}$ and $\beta = (1-u) * \frac{1-f}{f}$.

It can be shown that the marginal distribution of Y_{gci} is a beta-binomial:

$$Y_{gci} \sim BetaBinomial(N_{gci}, u_{gci}, f_{gc})$$

In order to adjust other important factors such as demographic and clinical co-

variance.pdf

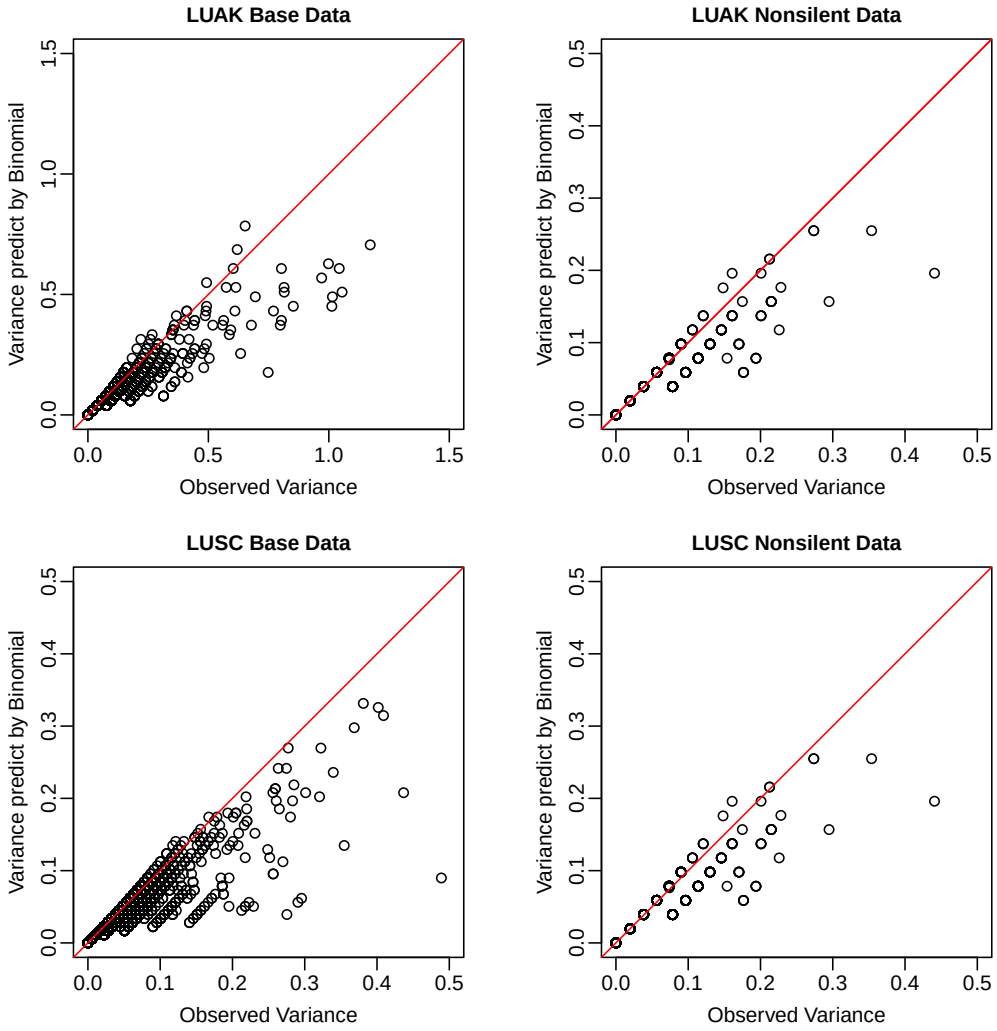

Figure 3.3: plot the estimated variance for the observed date against the variance predicted by Binomial distribution. The observed variance directly estimated using function "Var(.)" in R. The predicted variance estimated using formula derived from Binomial distribution.

variates, we use generalized linear model to describe the data. To be specific, we column bind background mutation data and non-silent mutation data together and treat them as two groups, and we want to test if the difference of mean rate between these two groups is equals to zero or not. The mean parameter u_{gci} is specified based on a generalized linear model with logit link function:

$$\log \frac{u_{gci}}{1 - u_{gci}} = X_i \beta_{gc}^T,$$

where X_i represents the i^{th} row of the design matrix, which is a vector of covariates that specifies the mutation type (background mutation and non-silent mutation) and other clinical factors applied to sample i. β_{gc} is a vector of regression coefficients quantifying the covariates effects for gene g category c. Let element β_{gc}^1 denote the log odds of mean background mutation rate for gene g in category c and β_{gc}^2 is corresponding log odds ratio of mean non-silent mutation rate compare to it's mean background mutation rate. The mutated gene can be detected by evaluating the hypothesis $H_0 : \beta_{gc}^2 = 0$, where β_{gc}^2 is the 2^{th} element of β_{gc}.

Accurate estimation of the dispersion parameter plays an important role in mutation detection. As we mentioned in the data description section, the most cases of mutation data are too sparse to provide enough information to obtain accurate estimates. From our first project, the shrinkage estimators have been shown to be useful when the information is limited from data. Specifically, we consider an empirical Bayes shrinkage method, which introduces a prior distribution for the dispersion parameter and borrows information from the ensemble of genes and categories to estimate the dispersion parameter for a specific gene-category. Figure3.4 show the empirical distribution of $\log(f_{gc})$ for LUSC data set and LUAK data set

Motivated by empirical distribution from real data(Figure3.4), we choose Log-

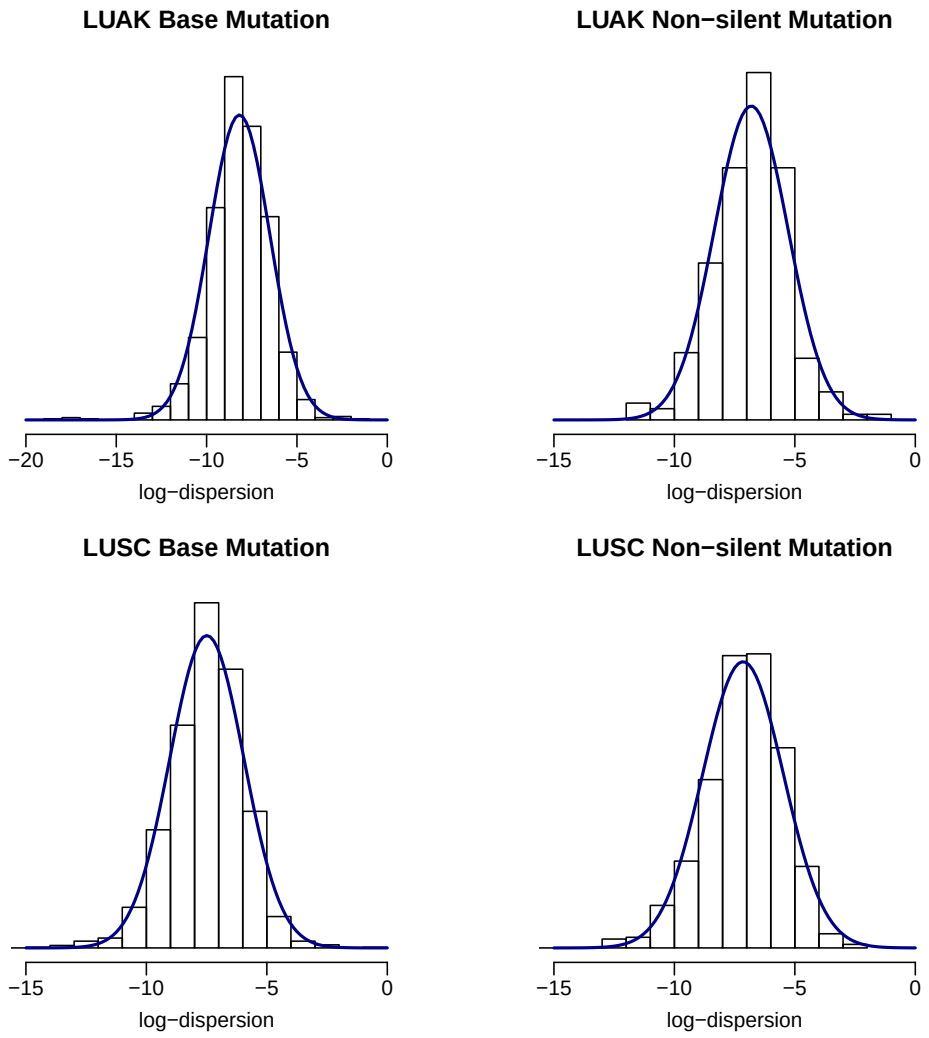

Figure 3.4: Histogram for the logarithm of estimated dispersions f_{gc} from LUSC data and LUAK data. The solid lines are density curves for normal distribution with parameters estimated from $\log(f_{gc})$. It can be seen that f_{gc} can be approximately modeled as a log-normal distribution.

normal distribution as prior. Thus, we consider the following prior:

$$f_{gc} \sim \text{log-normal}(m, \tau^2),$$

where m and τ^2 are hyper-parameters representing mean and variance for the normal distribution respectively.

To sum up, the hierarchical model we consider is as follows:

$$
\begin{aligned}
Y_{gci} &\sim BetaBinomial(N_{gci}, u_{gci}, f_{gc}) \\
\log \frac{u_{gci}}{1-u_{gci}} &= X_i \beta_{gc}^T \\
f_{gc} &\sim \text{log-normal}(m, \tau^2).
\end{aligned} \tag{3.1}
$$

3.5 Parameter estimation and differential detection analysis

Estimating hyper-parameters for the prior distribution of the dispersion parameter

The estimation of hyper-parameters for the prior distribution of the dispersion parameter is not straightforward. First of all, we roughly estimate hyper parameters and denoted as $\hat{m^r}$ and $\hat{\tau^r}$ using a five step procedure. First step, we roughly estimate the variance and mean mutation rate from the real data, and denote as $\widehat{Var(Y)}$ and $\hat{u}$ respectively. Second step, estimate the dispersion parameter using formula derived from Beta Binomial distribution , that is $\hat{f} = \frac{\widehat{Var(Y)} - N\hat{u}(1-\hat{u})}{N\hat{u}(1-\hat{u})(N-1)}$, where N is corresponding coverage data. Third step, due to the sparsity of the mutation data, we can observed some elements of $\hat{f}$ are negative, which means these data are not beta binomial distributed, Therefore we estimate the proportion of this kind of data, denote as $\hat{p^r}$. Fourth step, we filter the dispersion parameter $\hat{f}$ using two criterion: (1) $\hat{f}$>0; and (2) at least have 2 mutation counts across all samples. Fifth step, after filtering, we have a new dispersion vector $\hat{f'}$, and estimate $\hat{m^r} = mean(\log(\hat{f'}))$ and $\hat{\tau^r} = var(\log(\hat{f'}))$.

Since the simulation study showed that $\hat{\tau^r}$ overestimate the true value of τ. And we consider if we know the true value of m_t and τ, we can simulate pseudo data Y^s. Using the above five steps methods, we can estimate proportion parameter $\hat{p^s}$, mean parameter $\hat{m^s}$ and variance parameter $\hat{tau^s}$. Let $d = |\hat{p^r} - \hat{p^s}| + |\hat{m^r} - \hat{m^s}| + |\hat{\tau^r} + \hat{\tau^s}|$. We expect the true value of m and τ can minimize the distance d, so we finally estimate hyper parameters by minimizing the distance d, and denote as $\tilde{m}$ and $\tilde{\tau}$. Figure 3.5 shows the work flow for estimating hyper-parameters for the prior distribution of the dispersion parameter.

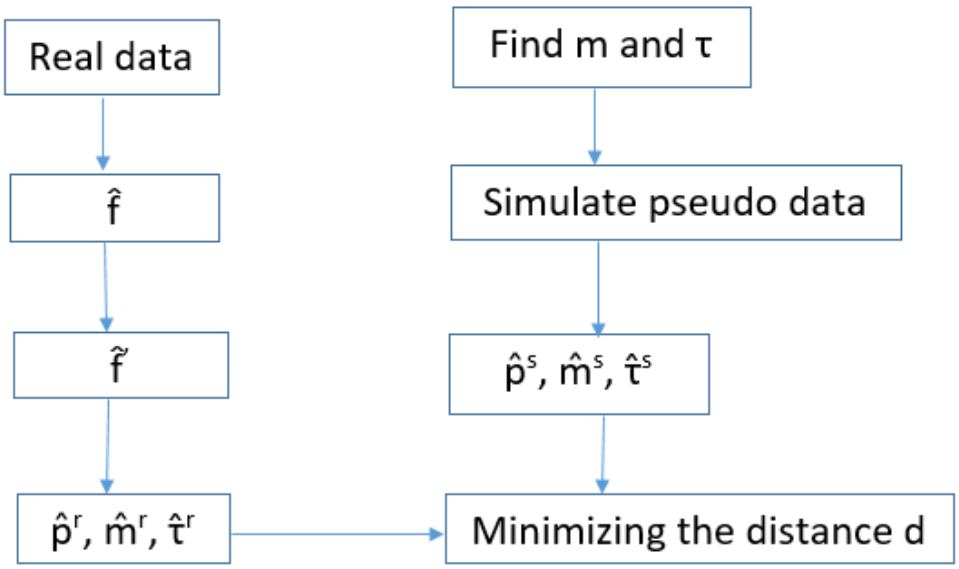

Figure 3.5: Work flow for estimating hyper-parameters for the prior distribution of the dispersion parameter

Estimating model coefficients and dispersion parameter

The marginal distribution of Y_{gci} is $BetaBinomial(N_{gci}, u_{gci}, f_{gc})$, where $N_{gci}u_{gci}$ is the mean and $N_{gci}u_{gci}(1-u_{gci}) + N_{gci}u_{gci}(1-u_{gci})(N_{gci}-1)f_{gc}$ is the variance. The

probability mass function is as follows:

$$P(Y_{gci} = k) = \binom{N_{gci}}{k} \frac{B(k + u_{gci}v_{gc}, N_{gci} - k + (1 - u_{gci})v_{gc})}{B(u_{gci}v_{gc}, (1 - u_{gci})v_{gc})}.$$

where $v_{gc} = \dfrac{1 - f_{gc}}{f_{gc}}$ and $B(.)$ is beta function.

To estimate f_{gc}, we consider its posterior distribution given Y_{gci}, N_{gci} and β_{gc}. In order to derive posterior penalize log likelihood function for f_{gc} we assume a log normal prior on the dispersion f_{gc}, e.g., $f_{gc} \sim$ log-normal(m, τ^2), with density function:

$$p(f_{gc}) = \frac{1}{f_{gc}\sqrt{2\pi\tau^2}} \exp\{-\tfrac{(\log f_{gc} - m)^2}{2\tau^2}\}.$$

Then the conditional posterior distribution for f_{gc} given all the observed counts and β_{gc} is:

$$p(f_{gc}|Y_{gci}, \beta_{gc}) \propto p(f_{gc})\Pi_i p(Y_{gci}|f_{gc}, N_{gci}, \beta_{gc}).$$

Therefore,

$$
\begin{aligned}
& \log\{p(f_{gc}|Y_{gci}, N_{gci}, \beta_{gc}, i = 1, ..., n)\} \\
\propto \quad & \log\{p(f_{gc})\} + \sum_i \log\{p(Y_{gci}|f_{gc}, N_{gci}, \beta_g)\} \\
\propto \quad & \sum_i \log B(Y_{gci} + u_{gci}v_{gc}, N_{gci} - Y_{gci} + (1 - u_{gci})v_{gc}) \\
- \quad & \sum_i \log B(u_{gci}v_{gc}, (1 - u_{gci})v_{gc}) \\
- \quad & \frac{(\log f_{gc} - m)^2}{2\tau^2} - \log f_{gc} - \log \tau,
\end{aligned}
\tag{3.2}
$$

where $\log B(.)$ is the log beta function. Equation (3.2) can be viewed as a penalized log likelihood function with penalty $-\frac{(\log f_{gc}-m)^2}{2\tau^2} - \log f_{gc}$. The first term in the penalty, $-\frac{(\log f_{gc}-m)^2}{2\tau^2}$ penalizing values that deviate far from the common prior m and the second term adds an additional penalty for positive value of $\log f_{gc}$. Estimates of hyper-parameters are plugged into Equation (3.2) and treated as constants. For a given u_{gci}, we obtain the estimate of f_{gc}, denoted by $\hat{f}_{gc}$, by maximizing Equation (3.2).

To estimate β_{gc}, we consider its likelihood function for a given f_{gc}:

$$L(\beta_{gc}|f_{gc}, N_{gci}) \propto \Pi_i p(Y_{gci}|f_{gc}, \beta_{gc}, N_{gci}),$$

therefor, the log likelihood function is:

$$
\begin{aligned}
& \log\{L(\beta_{gc}|f_{gc}, N_{gci})\} \\
\propto & \sum_i \log\{p(Y_{gci}|f_{gc}, \beta_{gc}, N_{gci})\} \\
\propto & \sum_i \log B(Y_{gci} + u_{gci}v_{gc}, N_{gci} - Y_{gci} + (1 - u_{gci})v_{gc}) \\
- & \sum_i \log B(u_{gci}v_{gc}, (1 - u_{gci})v_{gc}),
\end{aligned}
\tag{3.3}
$$

where $u_{gci} = \dfrac{\exp(X_i\beta_{gc}^T)}{1 + \exp(X_i\beta_{gc}^T)}$. We obtain the MLE, $\hat{\beta}_{gc}$ by maximizing Equation (3.3).

In this procedure, we first roughly estimate u_{gci} and plug it into Equation (3.2) to obtain $\tilde{f}_{gc}$. Then we plug $\tilde{f}_{gc}$ into Equation (3.3) to obtain $\tilde{\beta}_{gc}$. Second step is we plug $\tilde{\beta}_{gc}$ into Equation (3.2) to obtain $\hat{f}_{gc}$ again and treat them as final estimation. Then we plug $\hat{f}_{gc}$ into Equation (3.3) again to update $\hat{\beta}_{gc}$ and treat them as final estimation. Figure 3.6 shows the work flow of this procedure.

Hypothesis testing and false discovery rate

A Likelihood ratio test has been used for mutated gene detection. For each gene category level, we compare the maximum of Equation (3.3) under the null hypothesis versus that without any constraint, the chi-square approximation is used to obtain a p-value. We consider two approach to obtain p-value for each gene: In the first approach, for a given gene, we assume all categories are independent and the test

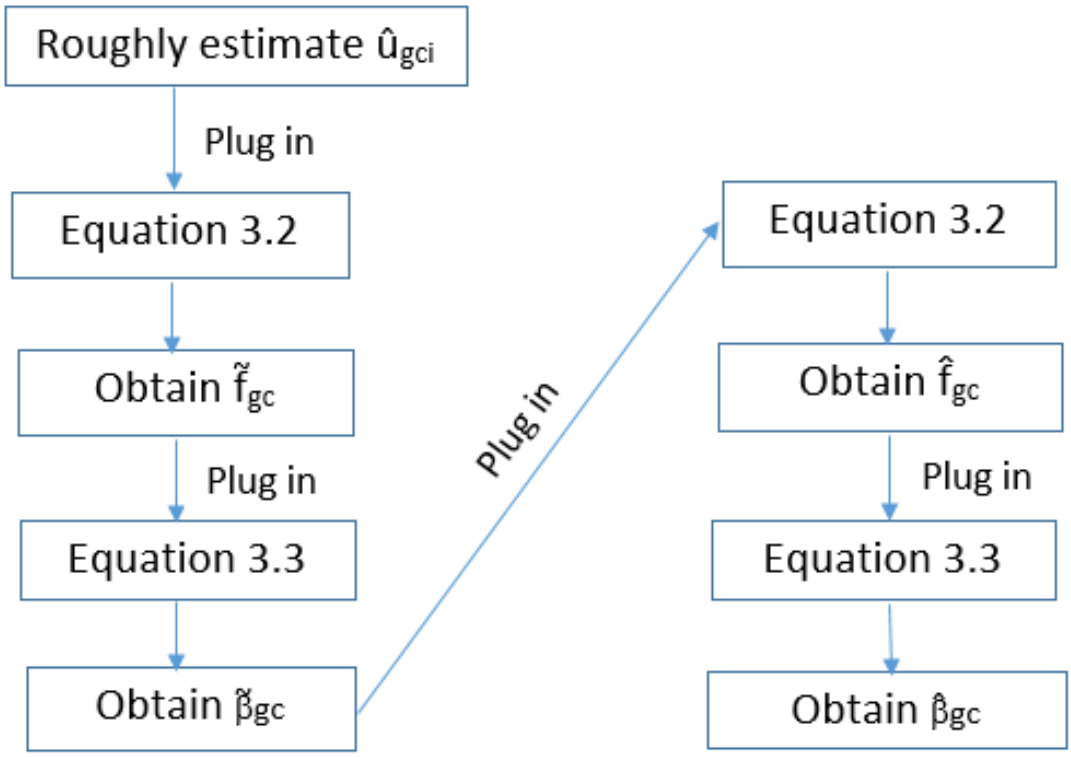

Figure 3.6: Work flow of estimating model coefficients and dispersion

statistic for that gene is the sum of the test statistics from all categories. Using the feature of the chi-square distribution, the gene test statistic approximately follow chi-square distribution with degree freedom is number of categories. The second approach is minP method. For each gene, we order the category p-value from low to high $p_{(1)} < p_{(2)} < ... < p_{(m)}$, where m is number of category. Then we assume $p_{(1)} \sim Beta(1, m)$ and obtain gene level p-value using Beta distribution.

For likelihood ratio multiple testing, the Benjamini and Hochberg procedure(Benjamini and Hochberg, 1995) is used to calculation the false discovery rate (FDR), which provides a choice of a cutoff for statistical significance.

Table 3.3: Differential mutation analysis results for highly mutated gene based on LUSC data. Gene level p-value obtained using chi-square distribution. The table lists the top 10 significantly mutated genes. Five of those mutated genes (indicated by *) were also identified by MutSig

Gene	p-value	q-value
$TP53^*$	< 0.001	< 0.001
CSMD3	< 0.001	< 0.001
$PIK3CA^*$	< 0.001	0.029
TTN	< 0.001	0.106
$CDKN2A^*$	< 0.001	0.111
$MLL2^*$	< 0.001	0.145
$KEAP1^*$	< 0.001	0.199
COL11A1	< 0.001	0.228
BAI3	< 0.001	0.243
RYR2	< 0.001	0.268

3.6 Real data analysis

The Cancer Genome Atlas Research Network using a modified version of the MutSig algorithm to identify mutated genes associate with squamous cell lung cancers based on LUSC data.(Network et al., 2012) Choosing 0.01 as the false discover rate(FDR) threshold, they identified 10 significantly mutated genes: TP53, CDKN2A, PTEN, PIK3CA, KEAP1, MLL2, HLA-A, NFE2L2, NOTCH1 and RB1. Those mutated gene have significantly higher non-silent mutation rate compare to it's background mutation rate. For method comparison, we also applied our method to LUSC data to identify significantly mutated genes with higher mutation rate. We rank the genes by FDR and p-value, then we pick top 10 genes list in Table 3.3 and Table 3.4 based on two different p-value adjust approaches.

As we mentioned before, MutSig only can identify mutated gene with higher mutation rate. However, the genes with significantly lower mutation rate also play specific and important role in cancer development. Based on LUSC data, we identify

Table 3.4: Differential mutation analysis results for highly mutated gene based on LUSC data. Gene level p-value obtained using minP method. The table lists the top 10 significantly mutated genes. Five of those mutated genes (indicated by *) were also identified by MutSig

Gene	p-value	q-value
$TP53^*$	< 0.001	< 0.001
CSMD3	< 0.001	< 0.001
$PIK3CA^*$	< 0.001	0.002
TTN	< 0.001	0.009
$MLL2^*$	< 0.001	0.024
COL11A1	< 0.001	0.03
$CDKN2A^*$	< 0.001	0.054
BAI3	< 0.001	0.107
$RB1^*$	< 0.001	0.181
TPTE	< 0.001	0.187

Table 3.5: Differential mutation analysis results for lowly mutated gene based on LUSC data. Gene level p-value obtained using chi-square distribution. FDR threshold was chosen as 5%

Gene	p-value	q-value
OR2L13	< 0.001	< 0.001
ZNF595	< 0.001	< 0.001
RMND5A	< 0.001	< 0.001
NACA	< 0.001	< 0.017
POTEH	< 0.001	< 0.017
PLEKHB2	< 0.001	< 0.019
Mar1	< 0.001	< 0.02
PSG6	< 0.001	< 0.029

Table 3.6: Differential mutation analysis results for lowly mutated gene based on LUSC data. Gene level p-value obtained using minP method. FDR threshold was chosen as 5%

Gene	p-value	q-value
OR2L13	< 0.001	< 0.001
ZNF595	< 0.001	< 0.016

lowly mutated genes and list result in Table 3.5 and Table 3.6 based on two different approaches. FDR threshold was chosen as 1% .

Chapter 4 A Beta-Binomial Model to Compare Somatic Mutation Rates Between Groups of Cancer Patients

4.1 Introduction

whole exome sequencing (WES) provides a powerful approach to profile somatic gene mutations in cancer genomes. A fundamental question in the analysis of WES data is how to compare somatic mutation patterns between groups of patients with varying characteristics such as geographic region, tumor stage, or response to therapy. Mutational processes are complex and stochastic processes, which affected by a lot of factors, including patients, treatments and environment. As show in Figure 4.1, mutation rate different across different categories, different patients and different groups.

Currently, the most frequently used statistical method for differential analysis between two groups is the Fisher's exact test (Ellis et al., 2012). One major limitation for using the Fisher's exact test to compare somatic mutation patterns between groups is that it assumes a constant mutation rate for patients from the same group. However, as shown in Figure 4.1, mutation rates are highly variable within different category and patients. In addition, Fisher's exact tests, do not recognize different types of mutations, e.g. transition vs. transversion, that have different mutation rates, do not account for the background mutation rate, and do not adjust for demographic and clinical covariates. The Fisher's exact test oversimplifies the comparison

rate.pdf

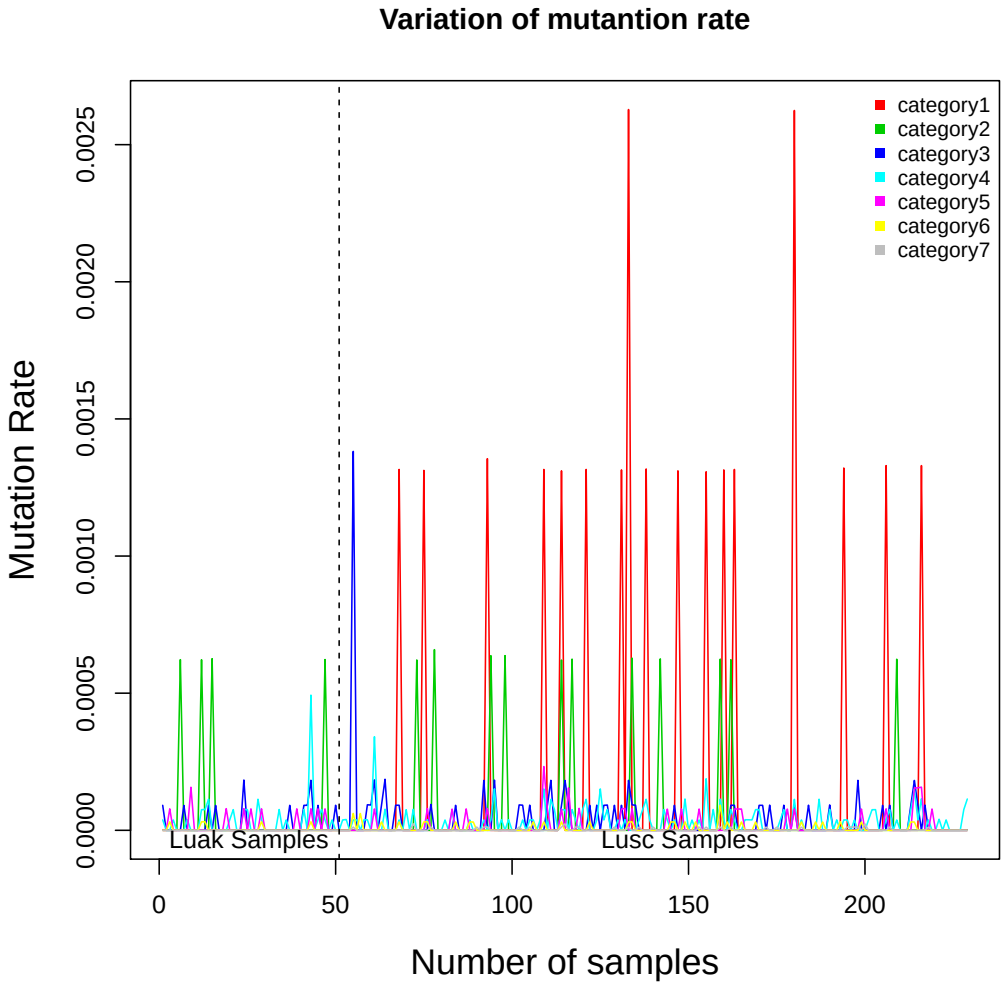

Figure 4.1: Plot mutation rate against different samples for one selected gene from LUSC data set and LUAK deata set

problem and may lead to less than optimal results.

We developed a beta-binomial model-based approach to compare somatic mutations between patient groups while fully taking into account the complexities that are ignored by current methods. Specifically, our method, named MutDiff, accounts for various types of variations in mutation rate, normalizes data based on background mutation rate, and adjusts for baseline covariates. We propose an empirical Bayes

shrinkage approach to estimate the dispersion parameter in the beta-binomial model and a likelihood ratio test to identify differentially mutated genes. Our method is applied to a squamous cell lung cancer genomic study to identify unique somatic mutation patterns in Appalachian Kentucky, which has the highest lung cancer rate in the nation, by comparing mutation rates between Appalachian samples from the study and non-Appalachian samples from The Cancer Genome Atlas.

4.2 Data description

In this chapter, we compared Appalachian samples (LUAK data) to non-Appalachian samples (LUSC data) from The Cancer Genome Atlas (TCGA) to identify genomic alteration patterns that are unique to patients in Appalachian Kentucky. Real data analysis and simulation study all based on these two data sets. The detail description about LUAK data and LUSC data presented in chapter 3, real data description section.

4.3 The data model

In this chapter, we still using beta binomial distribution to describe the number of mutation, let Y_{gci} and u_{gci} be the number and rate of mutations for gene g in category c from patient i, respectively. We have

$$Y_{gci} \sim BetaBinomial(N_{gci}, u_{gci}, f_{gc}),$$

where N_{gci} in corresponding sequencing coverage data and f_{gc} is dispersion pa-

rameter to address over-dispersion problem caused by mutational heterogeneity and biological variation.

For two group comparison, the model also can adjust other important demographic and clinical factors such as smoking status, we use generalized linear model to describe the data, the mean parameter u_{gci} is specified based on a generalized linear model with logit link function:

$$\log \frac{u_{gci}}{1 - u_{gci}} = X_i \beta_{gc}^T,$$

where X is design matrix with intercept column as first column and X_i represents the i^{th} row of the design matrix, which is a vector of covariates that specifies the treatment conditions applied to sample i. β_{gc} is a vector of regression coefficients quantifying the covariates effects for gene g category c.

We consider a simple example without any other covariates, we have two groups with sample size are 2 and 3 respectively. In this example, we have four data sets: group1 background mutation data, group2 background mutation data, group1 non-silent mutation data and group non-silent mutation. We column bind these four data sets in data analysis, correspondingly we have 4 elements in the β_{gc} vector, that is $\beta_{gc} = (\beta_{gc}^1, \beta_{gc}^2, \beta_{gc}^3, \beta_{gc}^4)$. β_{gc}^1 represent the log odds of background mean mutation rate for group1; β_{gc}^2 represent the log odds ratio of background mean mutation rate for group2 compare to group1; β_{gc}^3 represent the log odds ratio of non-silent mean mutation rate for group1 compare to it's background mean mutation rate; β_{gc}^4 represent

the difference of log odds ratios. Note that the first 4 elements in the β_{gc} vector are fixed. If we want to adjust other covariates, we can extend β from the fifth element. The design matrix is as follows:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Letting $(u_{gc}^1, u_{gc}^2, u_{gc}^3, u_{gc}^4)$ represent mean mutation rate for group1 background mutation data, group2 background mutation data, group1 non-silent mutation data and group2 non-silent mutation data respectively, we have

$$
\begin{aligned}
\log \frac{u_{gc}^1}{1-u_{gc}^1} &= \beta_{gc}^1 \\
\log \frac{u_{gc}^2}{1-u_{gc}^2} &= \beta_{gc}^1 + \beta_{gc}^2 \\
\log \frac{u_{gc}^3}{1-u_{gc}^3} &= \beta_{gc}^1 + \beta_{gc}^3 \\
\log \frac{u_{gc}^4}{1-u_{gc}^4} &= \beta_{gc}^1 + \beta_{gc}^2 + \beta_{gc}^3 + \beta_{gc}^4,
\end{aligned}
$$

Then, we can derive β_{gc}^4 as the difference of log odds ratios:

$$
\begin{aligned}
\beta_{gc}^4 &= (\log \frac{u_{gc}^4}{1-u_{gc}^4} - (\beta_{gc}^1 + \beta_{gc}^2)) - \beta_{gc}^3 \\
&= (\log \frac{u_{gc}^4}{1-u_{gc}^4} - \log \frac{u_{gc}^2}{1-u_{gc}^2}) - (\log \frac{u_{gc}^3}{1-u_{gc}^3} - \beta_{gc}^1) \\
&= (\log \frac{u_{gc}^4}{1-u_{gc}^4} - \log \frac{u_{gc}^2}{1-u_{gc}^2}) - (\log \frac{u_{gc}^3}{1-u_{gc}^3} - \log \frac{u_{gc}^1}{1-u_{gc}^1}).
\end{aligned}
$$

The differential mutation(DM) pattern analysis can be performed by evaluating the hypothesis $H_0 : \beta_{gc}^4 = 0$, where β_{gc}^4 is the 4^{th} element of β_{gc}.

Shrinkage estimators have been shown to be useful when the information is limited from data. In the two group comparison project, we still consider an empirical Bayes shrinkage method , which introduces a prior distribution for the dispersion parameter and borrows information for the estimate dispersion parameters by bor-

rowing information from other genes. Motivated by empirical distribution from real data(Figure3.4), we choose a Log-normal distribution as prior. Thus, we consider the following prior:

$$f_{gc} \sim \text{log-normal}(m, \tau^2),$$

where m and τ^2 are hyper-parameters representing mean and variance for the normal distribution respectively.

A key challenge in analysis of mutation data is the data sparsity and the limitation of sample size. The common phenomenon observed from real data set is that there is no any mutation for one gene across all samples. In this case, it is impossible to obtain stable and accurate mean estimators only using information from this gene, it is often useful to combine information from other genes to improve the estimation. We again employ an empirical Bayes procedure and choose normal distribution as prior motivated by naive estimators from real data(Figure 4.2) . Thus, we consider the following prior:

$$\beta_{\mathbf{gc}} \sim \text{Normal}(\mathbf{U_{gc}}, \Sigma),$$

where $U = (u_1, u_2, u_3, u_4)$ and Σ are hyper-parameters representing mean vector and variance covariance matrix for the normal distribution, respectively. We assume $\beta^{'}s$ are independent, so the variance covariate matrix Σ has the form

$$\begin{pmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \sigma_3^2 & 0 \\ 0 & 0 & 0 & \sigma_4^2 \end{pmatrix}$$

To sum up, the hierarchical model we consider is as follows:

$$\begin{aligned} Y_{gci} &\sim BetaBinomial(N_{gci}, u_{gci}, f_{gc}) \\ \log \frac{u_{gci}}{1-u_{gci}} &= X_i \beta_{gc}^T \\ f_{gc} &\sim \text{log-normal}(m, \tau^2) \\ \beta_{\mathbf{gc}} &\sim \text{Normal}(\mathbf{U_{gc}}, \Sigma). \end{aligned} \tag{4.1}$$

4.4 Parameter estimation and differential detection analysis

Estimating hyper-parameters for the prior distribution of the dispersion parameter

We used the same method and five step procedure described in chapter 3, to estimate the the hyper-parameters for the prior distribution of the dispersion parameter. Figure 3.5 show the work flow for this method.

Estimating hyper-parameters for the prior distribution of the β coefficients

As we mentioned before, β^2 and β^3 are log odds ratio and β^4 is the difference of the log odds ratios, so they are zero centered, we set $u_2 = u_3 = u_4 = 0$. Estimating u_1 is straightforward. First, we roughly estimate coefficients and denote as $\hat{\beta}^1$, $\hat{\beta}^2$, $\hat{\beta}^3$ and

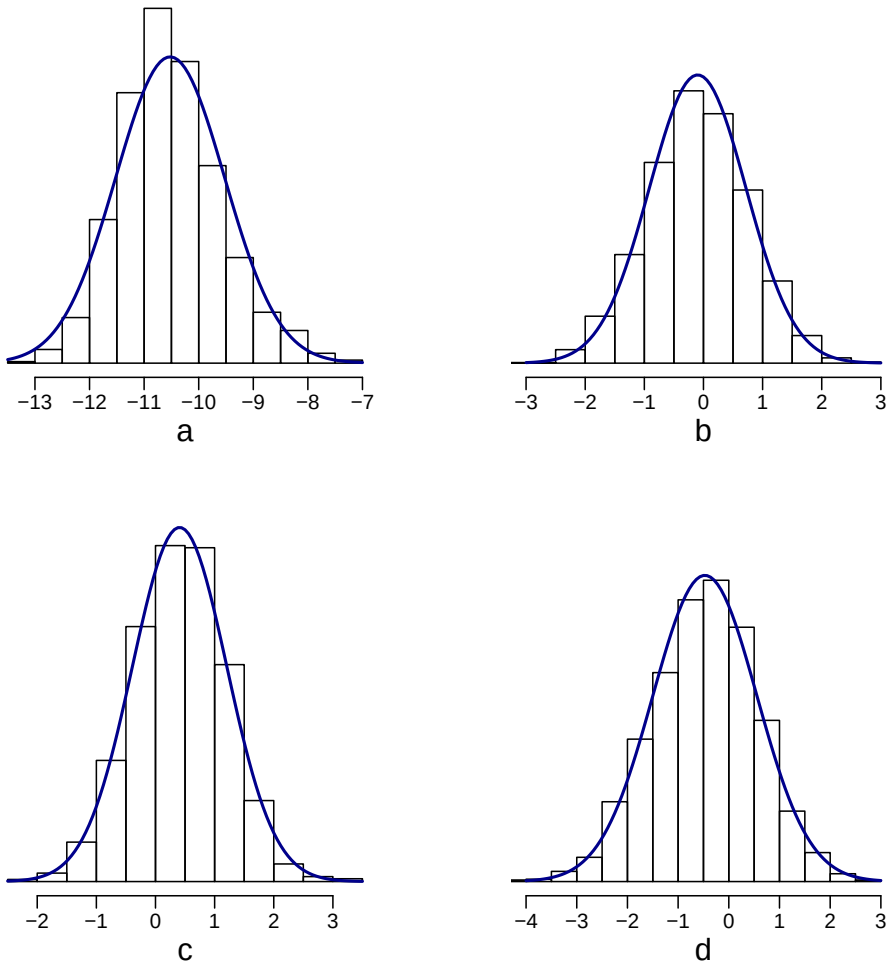

Figure 4.2: Histogram for the estimated mean parameters using LUAK data and LUSC data. Histogram (a) represent the log odds of mean silent mutation rate for LUAK data; Histogram (b) represent the log odds ratio of mean silent mutation rate for LUSC data compare to LUAK; Histogram (c) represent the log odds ratio of mean non-silent mutation rate for LUAK data compare to it's mean silent mutation rate; Histogram (d) represent the log odds ratio of mean non-silent mutation rate for LUSC data compare to it's mean silent mutation rate; The solid lines are density curves for normal distribution with parameters estimated from mean parameters. It can be seen that these mean parameters can be approximately modeled as a normal distribution.

$\hat{\beta}^4$. And then we estimate $\hat{u_1} = mean(\hat{\beta}^1)$.

In order to get robust estimators against outliers, we using quantile matching procedure to estimate the prior variance of the β coefficients. For zero centered

coefficients, we use the (1-p) empirical quantile of the observed absolute value of β_j which matches the (1-p/2) theoretical quantile of the prior, $N(u_j, \sigma_j)$, where $j = 2, 3, 4$ and set $P = 0.05$ by default. If we denote the empirical upper quantile of the observed absolute value of $\hat{\beta}_j$ as $Q_{|\beta^j|(1-p)}$ and the theoretical quantile of a normal distribution as $Z_{(1-p/2)}\sigma_j$, then the prior variance is calculated as:

$$\sigma_j = \frac{Q_{|\beta^j|(1-p)}}{Z_{(1-p/2)}}, j = 2, 3, 4.$$

Since β^1 is not zero centered, we using the (1-p) empirical quantile of the observed β_j matches the (1-p) theoretical quantile of the prior, then calculate $\sigma_1 = \frac{Q_{\beta^1(1-p)}}{Z_{(1-p)}}$.

Estimating model coefficients and dispersion parameter

The marginal distribution of Y_{gci} is $BetaBinomial(N_{gci}, u_{gci}, f_{gc})$ with the probability mass function is as follows:

$$P(Y_{gci} = k) = \binom{N_{gci}}{k} \frac{B(k + u_{gci}v_{gc}, N_{gci} - k + (1 - u_{gci})v_{gc})}{B(u_{gci}v_{gc}, (1 - u_{gci})v_{gc})}.$$

where $v_{gc} = \frac{1 - f_{gc}}{f_{gc}}$ and $B(.)$ is beta function. The likelihood function for β_{gc} is:

$$L(\beta_{gc}|N_{gci}, f_{gc}) \propto \Pi_i p(Y_{gci}|N_{gci}, f_{gc}, \beta_{gc}). \tag{4.2}$$

We assume a log normal prior on the dispersion f_{gc}, then we derive the penalize log likelihood function(Detail steps described in chapter 3):

$$
\begin{aligned}
& \log\{p(f_{gct}|Y_{gcti}, N_{gcti}, \beta_{gc}, i = 1, ..., n)\} \\
\propto \;& \log\{p(f_{gct})\} + \sum_i \log\{p(Y_{gcti}|f_{gct}, N_{gcti}, \beta_g)\} \\
\propto \;& \sum_i \log B(Y_{gcti} + u_{gcti}v_{gct}, N_{gcti} - Y_{gcti} + (1 - u_{gcti})v_{gct}) \\
- \;& \sum_i \log B(u_{gcti}v_{gct}, (1 - u_{gcti})v_{gct}) \\
- \;& \frac{(\log f_{gct} - m_t)^2}{2\tau_t^2} - \log f_{gct} - \log \tau_t,
\end{aligned}
\tag{4.3}
$$

Estimates of hyper-parameters are plugged into Equation (4.3) and treated as constants. For a given u_{gci}, we obtain the estimate of f_gc by maximizing Equation (4.3).

To estimate β_{gc}, we need consider its posterior distribution given Y_{gci}, N_{gci} and f_{gc}. In order to derive posterior penalize log likelihood function for β_{gc} we assume normal distributions as prior , e.g., $\beta_{\mathbf{gc}} \sim \text{Normal}(\mathbf{U_{gc}}, \Sigma)$., with density function:

$$p(\beta_{gc}) = \frac{1}{\sqrt{2\pi\Sigma}} \exp\{-\tfrac{(\beta_{gc} - U_{gc})^2}{2\Sigma}\}.$$

Then the conditional posterior distribution for β_{gc} given all the observed counts and f_{gc} is:

$$p(\beta_{gc}|Y_{gci}, N_{gci}f_{gc}) \propto p(\beta_{gc})\Pi_i p(Y_{gci}|\beta_{gc}, f_{gc}, N_{gci}).$$

Therefore,

$$
\begin{aligned}
& \log\{p(\beta_{gc}|Y_{gci}, N_{gci}, f_{gc}, i = 1, ..., n)\} \\
\propto \;& \log\{p(\beta_{gc})\} + \sum_i \log\{p(Y_{gci}|f_{gc}, N_{gci}, \beta_g)\} \\
\propto \;& \sum_i \log B(Y_{gci} + u_{gci}v_{gc}, N_{gci} - Y_{gci} + (1 - u_{gci})v_{gc}) \\
- \;& \sum_i \log B(u_{gci}v_{gc}, (1 - u_{gci})v_{gc}) \\
- \;& \frac{(\log \beta^1_{gc} - u_1)^2}{2\sigma_1^2} - \log \sigma_1 \\
- \;& \frac{(\log \beta^2_{gc})^2}{2\sigma_2^2} - \log \sigma_2 \\
- \;& \frac{(\log \beta^3_{gc})^2}{2\sigma_3^2} - \log \sigma_3 \\
- \;& \frac{(\log \beta^4_{gc})^2}{2\sigma_4^2} - \log \sigma_4,
\end{aligned}
\tag{4.4}
$$

Equation (4.4) can be viewed as a penalized log likelihood function with penalty $-\frac{(\log \beta^1_{gc} - u_1)^2}{2\sigma_1^2} - \log \sigma_1 - \frac{(\log \beta^2_{gc})^2}{2\sigma_2^2} - \log \sigma_2 - \frac{(\log \beta^3_{gc})^2}{2\sigma_3^2} - \log \sigma_3 - \frac{(\log \beta^4_{gc})^2}{2\sigma_4^2} - \log \sigma_4$. Estimates of hyper-parameters are plugged into Equation (**??**) and treated as constants. For a given f_{gc}, we obtain the estimate of β_{gc}, denoted by $\hat{\beta}_{gc}$, by maximizing Equation (4.4).

We used the same procedure we described in chapter 3 to estimate model coefficients and dispersion parameter iteratively. We first roughly estimate u_{gci} and plug it into Equation (4.3) to obtain $\tilde{f}_{gc}$ and plug into Equation (4.4) to obtain $\tilde{\beta}_{gc}$. In the second step, we put $\tilde{\beta}_{gc}$ back into Equation (4.3) to obtain $\hat{f}_{gc}$ again and treat them as the final estimation. Then we put $\hat{f}_{gc}$ into Equation (4.4) again to update $\hat{\beta}_{gc}$ and

treat them as the final estimation.

Hypothesis testing and false discovery rate

We still consider a likelihood ratio test for differential mutation pattern detection. For each gene category level, we compare the maximum of Equation (4.2) under the null hypothesis versus that without group constraint. The hypothesis test for gene g category c is straightforward:

$$H_0 \colon \beta_{gc}^4 = 0$$

$$H_a \colon \beta_{gc}^4 \neq 0.$$

To obtain p-value for each gene, we assume all categories are independent and sum the test statistics together for each gene, and assume that the summation approximately follows a chi-square distribution with the degree of freedom is equal to the number of categories.

For likelihood ratio multiple testing, the Benjamini and Hochberg procedure(Benjamini and Hochberg, 1995) is used to calculate the false discovery rate (FDR), which provides a choice of a cutoff for statistical significance.

4.5 Simulations

Simulation setting

We performed comprehensive simulation studies to evaluate the performance of our MutDiff method and to compare with Fisfer's exact test that is the most frequently used statistical method for differential analysis between two groups. For Fisher's exact test, we ignore coverage data set, silent mutation data set and only focus on non-silent mutation data set for each group. For each gene, the test compares the proportions of patients with non-silent mutations in that gene between the two groups.

To evaluate differential mutation pattern detection under known truth, and to conduct the comparison under realistic scenarios encountered in experiments, simulated data were generated based on a model (4.1) using parameters estimated from the two real datasets described above. We focused on the two-group comparison situation and simulated two group of data based on two real data sets respectively. To be specific, we simulated group 1 mutation data and coverage data for both non-silent and background mutation data based on parameters estimated from LUAK data; We simulated group 2 mutation data and coverage data for both non-silent and background mutation data based on parameters estimated from LUSC data.

Expressions of 5000 genes with their categories were generated from

$$Y_{gci} \sim BetaBinomial(N_{gci}, u_{gci}, f_{gc}),$$

for two groups, where model parameter values were set based on two real datasets.

Coverage data N_{gc} were randomly generated from a Poisson distribution. Specifically, for group1, we obtain $N_{gc0} \sim Poisson(\lambda_0^1)$ for background coverage data, $N_{gc1} \sim Poisson(\lambda_1^1)$ for non-silent coverage data, where λ_0^1 and λ_1^1 were calculated from LUAK background mutation coverage data and non-silent mutation coverage data respectively. Here 0 denotes background data and 1 denotes nonsilent data. For group2, we get $N_{gc0} \sim Poisson(\lambda_0^2)$ for silent coverage data, $N_{gc1} \sim Poisson(\lambda_1^2)$ for non-silent coverage data. Again, λ_0^2 and λ_1^2 were calculated from LUSC background mutation coverage data and non-silent mutation coverage data respectively. The mean parameter, u_{gci}, was randomly re-sampled from the parameters calculated from the real datasets for each group. We considered 5% genes as truly differentially mutated with the log odds ratio randomly generated from a mixture distribution $0.5N(4, 0.5) + 0.5N(-4, 0.5)$. For the log dispersion parameter, f_{gc}, we use a normal distribution with distribution parameters calculated from real data. Explicitly, we obtained $f_{gc0} \sim LogN(-10.68, 2.75^2)$ for background mutation data, $f_{gc1} \sim LogN(-10.68, 3.63^2)$ for non-silent mutation data. We considered 100 simulations for each simulation scenario.

In the simulation study, we considered the simulation settings with non-consistent category mutation rate for differential mutation gene. Explicitly, for one specific DM gene, we randomly selected some categories with non-silent mutation rate from group1 higher than those from group2, but the other categories on the opposite way. The non-silent mutation rate all adjusted by their own background mutation rate.

Simulation results

We first evaluated the estimation of dispersion parameters. Figures 4.3 and 4.4 show the estimated versus true dispersions for background mutation and non-silent mutation based on two simulated datasets. Our empirical Bayes shrinkage estimator tracks the real dispersion for gene-category levels with higher mutation counts. For gene-category levels with lower mutation counts, the dispersion parameter estimates were shrunk more towards the average dispersion, because there was little information about dispersion for those gene-category levels in the data.

We next evaluate the estimation of coefficient parameters $\beta_{gc} = (\beta_{gc}^1, \beta_{gc}^2, \beta_{gc}^3, \beta_{gc}^4)$. Figure 4.5(a) plots the estimated log odds of background mean mutation rate for group1 against it's true value. Figure 4.5(b) plots the estimated log odds ratio of background mean mutation rate for group2 compare to group1 against it's true value; MutDiff provides accurate estimation of those parameters. Figure 4.5(c) plots the estimated log odds ratio of non-silent mean mutation rate for group1 and background mean mutation rate against it's true value. Figure 4.5(d) plots the estimated difference of log odds ratio. As expected, the plot has three separate parts, one part is zero centered non-differential mutated gene-category levels and the other two parts are truly differentially mutated with the log odds ratio randomly generated from a mixture distribution $0.5N(-4, 0.5) + 0.5N(4, 0.5)$.

We then evaluate the estimation of mean mutation rate for both non-silent and background mutation data based on two simulated datasets. Figure 4.6 shows the estimated versus true mean mutation rate. The plot shows that the estimated mean

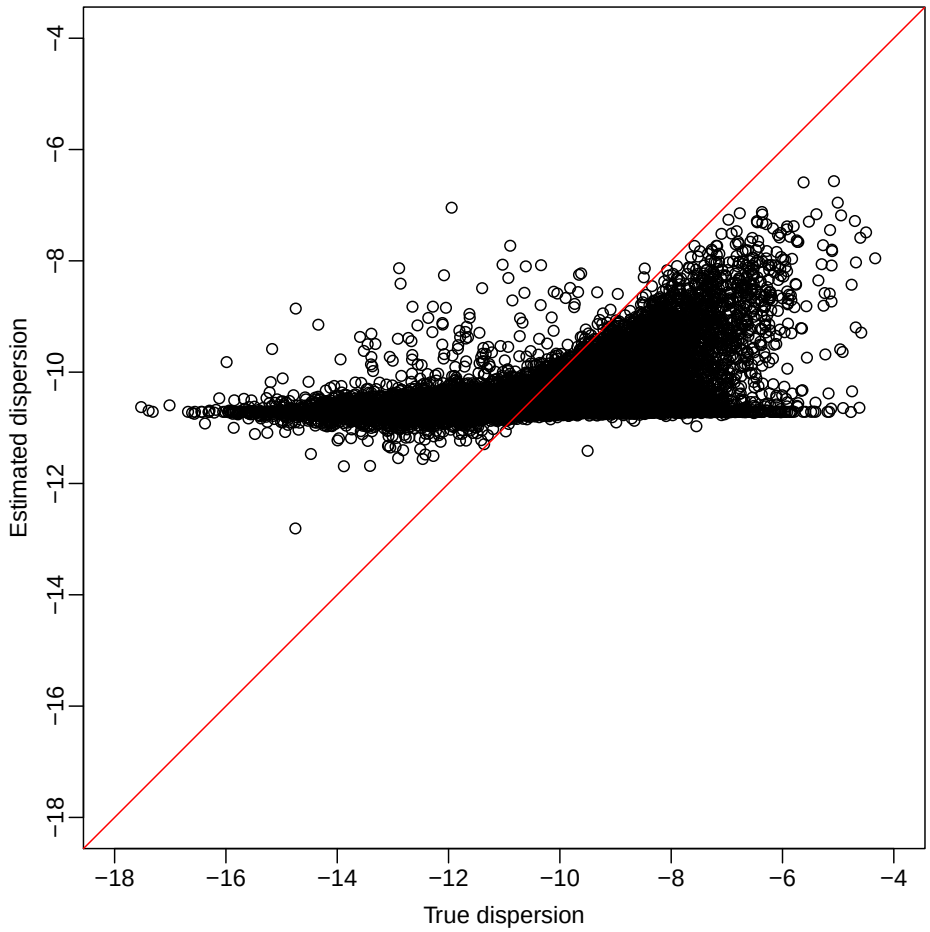

Figure 4.3: Estimation of dispersion based on background mutation data

mutation rate track the true value well especially for the non-silent mutation rate.

An important task of differential mutation pattern analysis is to rank genes based on their evidence of being differentially mutated. From this point of view, the ability to have as many true positives as possible in the top-ranked genes is a critical part of evaluating the performance of a method. We first compared the number of true differential mutation genes identified for a given top ranked genes (150, 200, or 250) between different methods. As shown in Figure 4.7, MutDiff perform well to have

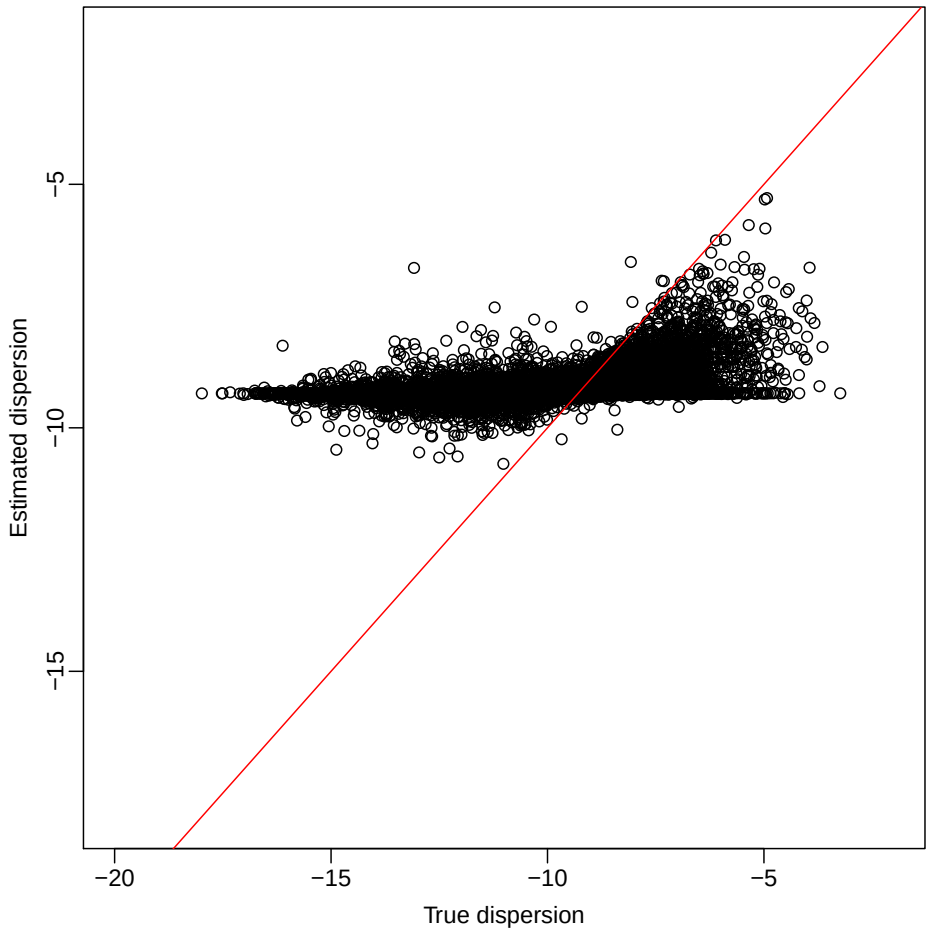

Figure 4.4: Estimation of dispersion based on non-silent mutation data

as many true positives as possible in the top-ranked genes compare to Fisher's exact test. To be specific, top 150 genes ranked by MutDiff include 148 true DM genes, top 200 genes include 196 true DM genes and top 250 genes include 217 true DM genes.The including true DM genes increase smoothly as rank increase. Note that we have 250 true DM genes in the simulated dataset, around 86.8% true DM genes include in top 250 genes ranked by MutDiff and around 67.7% true DM genes include in top 250 genes ranked by Fisher's exact test.

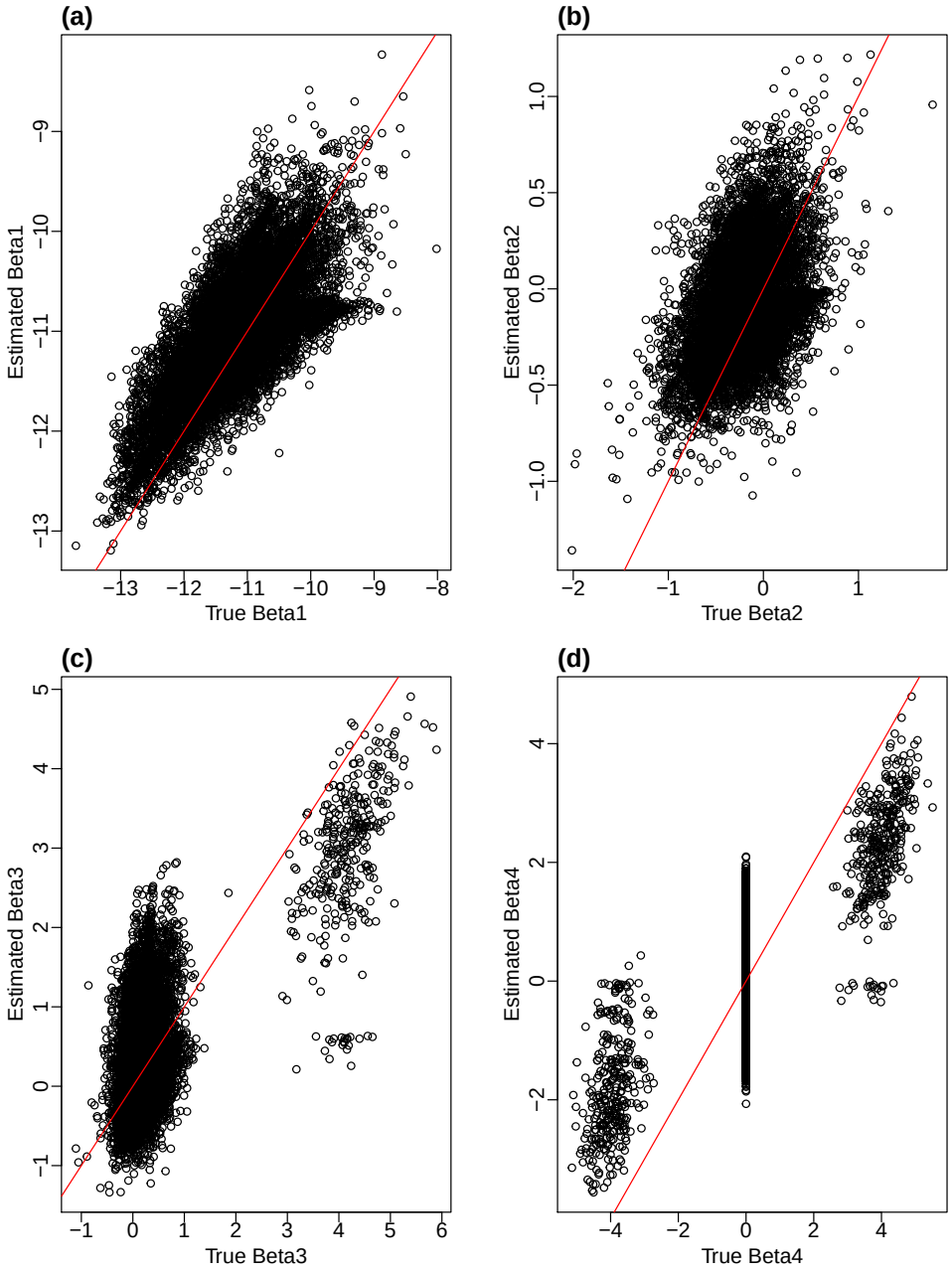

Figure 4.5: Estimation of coefficients

In practice, DM genes are often declared based on a user-specified FDR threshold. Given the threshold, a powerful method is expected to identify as many true DM genes as possible. We compared the number of DM genes identified under a given FDR threshold (0.005, 0.01, 0.05, 0.1 or 0.2) between two methods. As shown in Figure

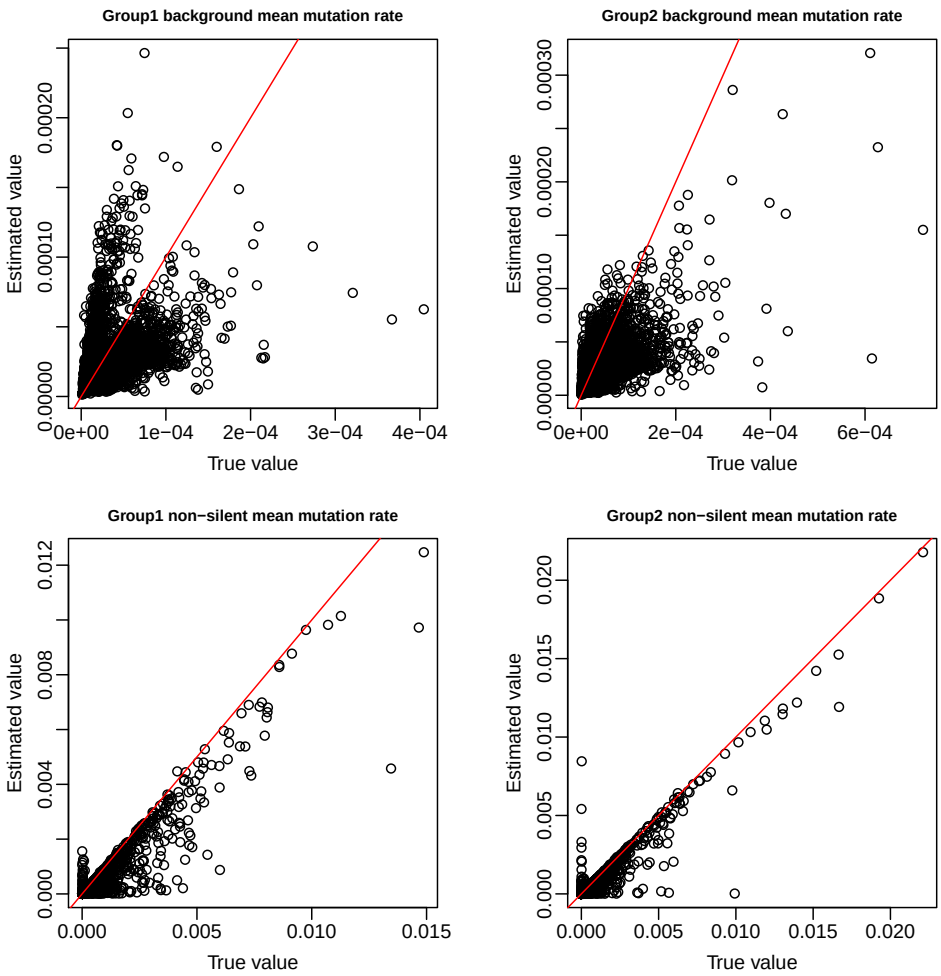

Figure 4.6: Estimation of mean mutation rate

4.8, MutDiff detected more true DM genes than Fisher's exact test. The colorful bars represent DM genes identified by different methods, and the shaded area represents false discoveries, that is the number of non-DM genes within positive calls.

4.6 Real data analysis

We applied MutDiff to identify novel somatic mutation patterns that are unique to squamous cell lung cancer patients in Appalachian Kentucky by comparing mutations

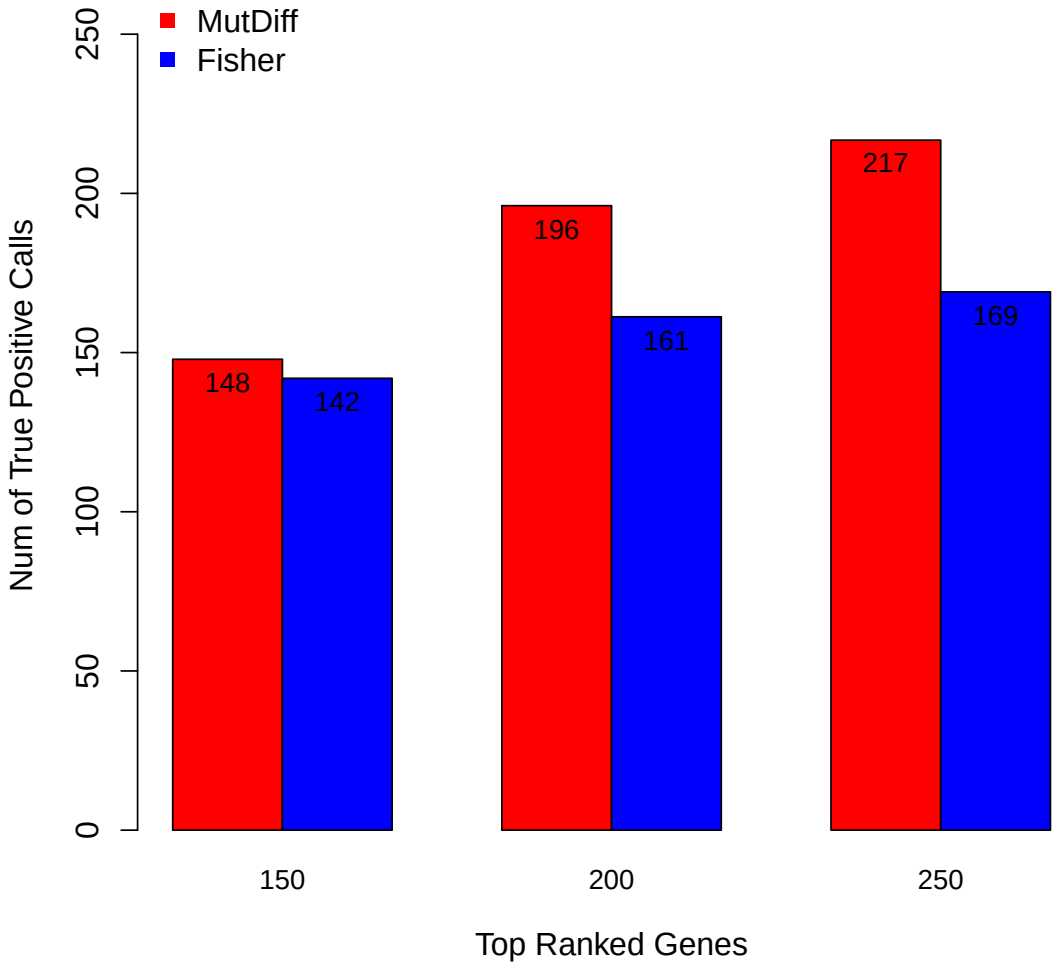

Figure 4.7: Bar charts for number of true differential mutation pattern for a given top ranked gene. Results were averaged across 50 datasets simulated based on the real data with 100 replicates.

between Appalachian (LUAK data) and non-Appalachian (LUSC data) samples.For methods comparison, we also considered Fisher's exact test. We compared the top 10 genes identified by each method and found only one gene that were identified by both method. Figure 4.9 shows that MutDiff and Fisher's exact test have different ranking mechanism. We will further explore the biological meaning and function for

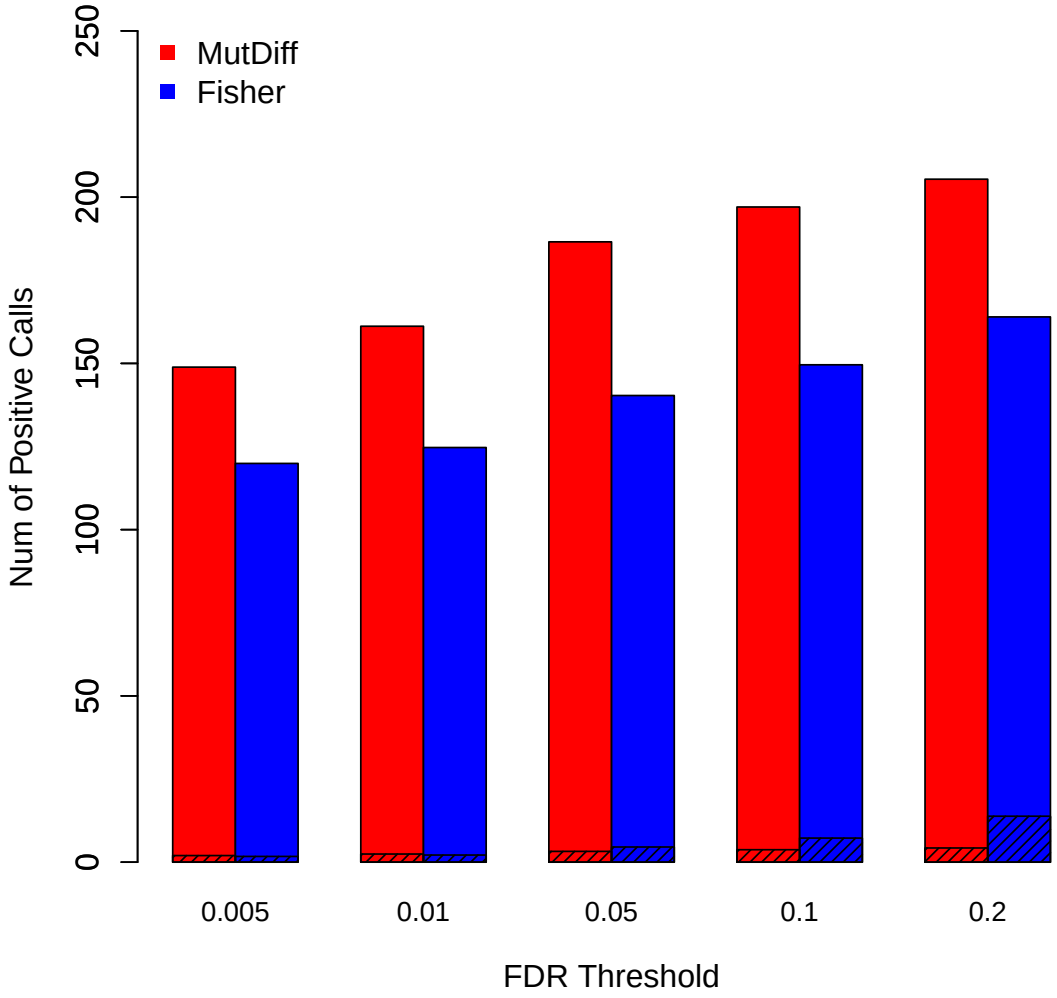

Figure 4.8: Bar charts for number of positive calls under a given FDR threshold (0.005, 0.01, 0.05, 0.1 or 0.2) comparing different methods. Results were averaged across 50 datasets simulated based on the real data sets with 100 replicates. The shaded area represents false discoveries, i.e. the number of non-DE genes within positive calls.

these genes to evaluate whether our method is able to identify more biologically and clinically-relevant genes to squamous cell lung cancer and whether the Fisher's exact method tends to make more false positive calls.

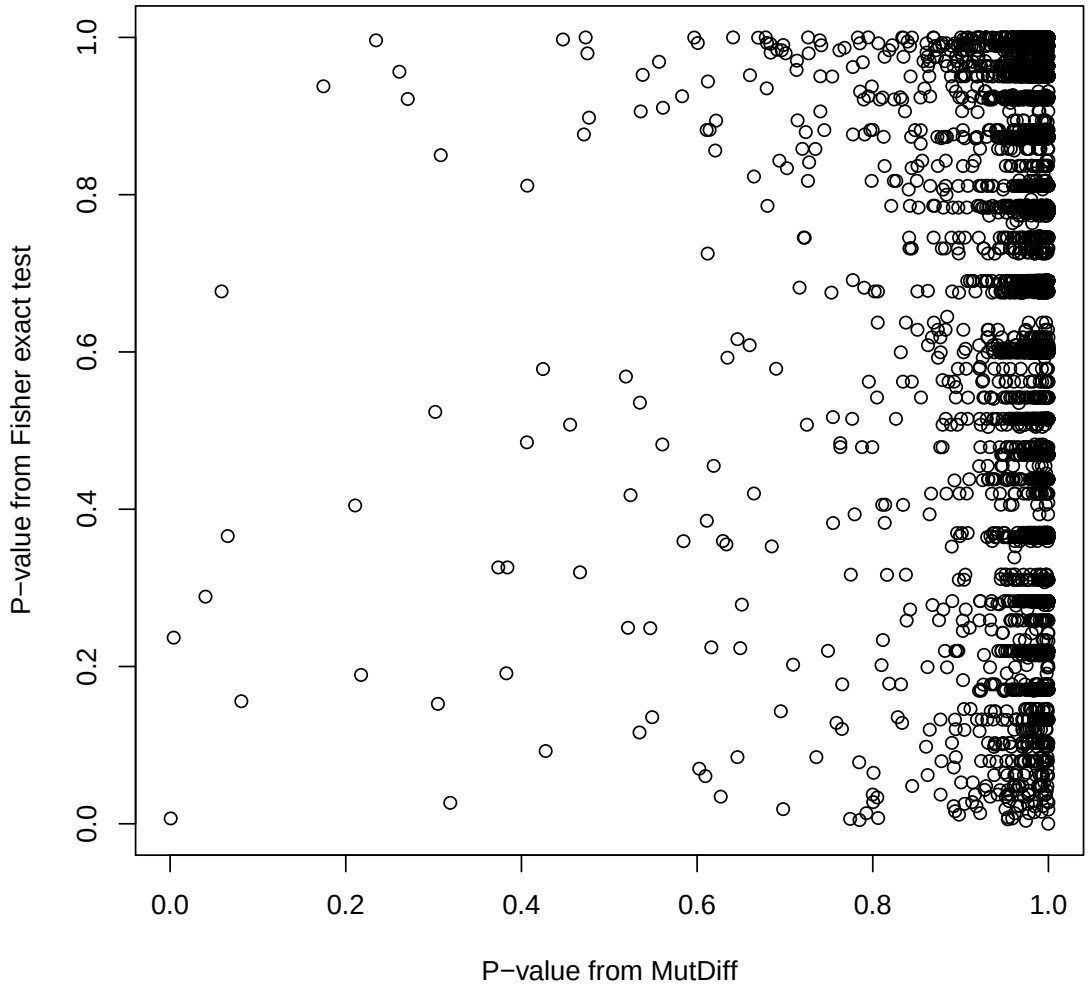

Figure 4.9: Gene level p-value obtained from MutDiff against gene level p-value from Fisher's exact test.

Chapter 5 Discussion

5.1 Discussion for project based on NanoString nCounter data

NanoStringDiff offers a comprehensive and general framework to characterize NanoString nCounter data and to detect DE genes for both simple and complex experimental designs. As a method specifically designed for nCounter data, it utilizes a negative binomial-based model to fit the discrete nature of the data and incorporates several normalization parameters in the model to fully adjust for platform source of variation, sample content variation and background noise. Simulation and real data analyses results show that this new method outperforms the existing methods in DE detection.

The choice of housekeeping genes is a crucial part of the experimental design. It is expected that those housekeeping genes are stable in their expression levels, that is, the observed read counts should not vary much across samples or replicates. In real data analysis, however, this is not always the case. We therefore recommend checking the variation of housekeeping genes, removing those showing large variation, prior to estimating housekeeping size factor. One possible approach is to use only the top three housekeeping genes with the smallest variation to calculate the housekeeping size factor. Further investigation of this issue to develop an optimal approach to select and use housekeeping genes will be very important.

We first choose a normal distribution as the prior for the log-dispersion parameter.

Our model appears to be robust to this specification. In simulations where the dispersions were generated by randomly re-sampling from the dispersions estimated from the real data, our method still provided satisfactory results in terms of the number of positive calls and FDR control.

The NanoStringDiff is computationally more intensive than NanoStringNorm, DESeq2, and edgeR. In order to adjust the effect of background noise, we assume the distribution of read count is the convolution of a negative binomial and a Poisson distribution, which introduces a summation from zero to the observed read count within the log operator in Equation (2.4) and makes the algorithm more time consuming. This is not a big issue when the observed read counts are not too large. But when many of the observed counts are larger than a thousand, the algorithm can be slow. Developing an approximation approach to enable faster calculation of Equation (2.4) is an objective of our future research.

The likelihood ratio test in our algorithm utilizes a chi-square approximation to calculate p-values. The performance of this approximation was evaluated in Figure 4.9, where we plotted the reported type I error rate against the true type I error rate based on simulated data. When the sample size was not very small or the biological variation was not large, the reported type I error rate was close to the true value, suggesting the approximation was accurate. However, when the sample size was very small and the biological variation was large, the reported type I error rate was smaller than the true value. Therefore, the approximation led to an inflated type I error rate under such situation. As a result, the FDR was also inflated, making our method anti-conservative. An important topic for future research is to develop a

correction method to improve the performance of the chi-square approximation under such situation.

5.2 Discussion for projects based on mutation data

In the proposed study, we harness state-of-the-art statistical and bioinformatics techniques to address the gap in differential analysis of somatic mutation data from WES. We are the first to propose a comprehensive statistical model to rigorously account for the background mutation rate and various sources of variation in mutation data. One important source of variation is the biological variation in mutation rate for patients within a certain group, which is characterized by a gene-specific dispersion parameter in our model. Due to the small sample size and limited number of mutations in each gene, estimation of the dispersion parameter can be unstable. We therefore propose an empirical Bayes shrinkage method to borrow information from the ensemble of genes, which can provide more stable estimation of the dispersion parameter and mean parameter about each gene individually. To our knowledge, this is the first time this method will be used for somatic mutation differential analysis.

Our project will have great translational potential. Our short-term goal is to continue to build collaborations with clinician scientists as well as molecular epidemiologists. For example, we may discover novel markers amenable for targeted treatment. In collaboration with clinicians, our findings can shed light on potentially more appropriate and personalized strategies for therapeutic interventions in the Appalachian KY population that can be developed into clinical trials. Collaborations with a molecular epidemiologist will allow development of population-based studies

and interventions within the community.

The likelihood ratio test in our algorithm utilizes a chi-square approximation to calculate p-values. However, we estimate the mean parameters using empirical Bayes shrinkage method to borrow information from other genes and shrink the parameters to the prior mean. In this case, the chi-square distribution might not be an appropriate model to describe the test statistics form likelihood ratio test. As a result, the type I error and false discovery rate can't control well in the algorithm. An important topic for future research is to develop a correction method to improve the performance of the chi-square approximation under such situation. An alternative approach is to use Wald test to calculate the p-values in the project.

Mutational processes is a complex processes, there are three main types of heterogeneity associate within this process: heterogeneity across patients in the same group, heterogeneity in the mutational spectrum of the tumor and regional heterogeneity across the genome. In our algorithm, we considered the first two heterogeneities. To be specific, we developed a beta-binomial model-based approach to compare somatic mutations between patient groups accounts for various types of variations in mutation rate, normalizes data based on background mutation rate, and adjusts for baseline covariates. However, in this project, we haven't consider the regional heterogeneity across the genome. Gene expression level and replication time are two important factors in explaining the regional heterogeneity. An important step in the future study is try to consider regional heterogeneity and include gene expression level and replication time in the model framework.

The MuDiff is computationally more intensive than Fisher's exact test. For each

gene, the Fisher's exact test compares the proportions of patients with non-silent mutations in that gene between the two groups, which oversimplifies the comparison problem and may lead to less than optimal results. We propose a beta-binomial model-based approach to compare somatic mutation patterns between two groups while fully taking into account the complexities that are ignored by the Fishers exact test and provides more accurate and powerful results in differential mutation pattern detection in cost of time consuming. Developing an approximation approach to enable faster calculation is an other objective of our future research.

In our algorithm, we using empirical Bayes shrinkage method to estimate mean parameters and and shrink the parameters to the prior mean. Based on the current design matrix used in the algorithm, the order of the dataset is fixed. To recover the symmetry problem between all levels, the future work will consider an alternative approach to introduce an expanded design matrix, which include an indicator variable for each level of each factor, in addition to an intercept column.

The method proposed in this project tries to detect differential somatic mutation patterns and/or driver mutations at the gene level. Due to the low mutation rates in most genes, the power of the differential analysis is limited. As genes are usually involved in biological pathways, an extension of our work is to perform differential mutation analysis at the pathway level. As a pathway contains a set of genes, its mutation rate is much higher than individual genes. Therefore, the differential pathway analysis is likely to yield higher power to detect biologically interesting patterns.

Appendix

Main Functions in the first project

R function glm.LRT

```
glm.LRT <- function(NanoStringData, design.full, Beta = ncol(design.full), contrast = NULL) {

c = positiveFactor(NanoStringData)
d = housekeepingFactor(NanoStringData)
k = c * d
lamda_i = negativeFactor(NanoStringData)

if (length(k) == 0) {
stop("Before calling function glm.LRT, should get normalization factors \n
first using function estNormalizationFactors")
}

Y = exprs(NanoStringData)
Y_n = sweep(Y, 2, lamda_i, FUN = "-")
Y_nph = sweep(Y_n, 2, k, FUN = "/")
Y_nph[Y_nph <= 0] = 0.1

X.full = design.full
nsamples = ncol(Y)
Beta.names = colnames(design.full)

result.full = glmfit.full(NanoStringData, design.full)
Beta.full = result.full$Beta.full
U.full = result.full$mean.full
phi.hat = result.full$dispersion
df.full = result.full$df.full
m0 = result.full$m0
sigma = result.full$sigma

V.full = sweep(U.full, 2, k, FUN = "*")

## Make reduced design matrix.
## Here we borrow the idea from paskage edgeR to make reduced design matrix

if (is.null(contrast)) {
if (length(Beta) > 1)
Beta = unique(Beta)

if (is.character(Beta)) {
```

```
check.Beta = Beta %in% Beta.names
if (any(!check.Beta))
stop("The name(s) of Beta arguments do not match the \n
name(s) of the design matrix.")
Beta = match(Beta, Beta.names)
}

logFC = Beta.full[, Beta, drop = FALSE]/log(2)
} else {
contrast = as.matrix(contrast)
qrc = qr(contrast)
ncontrasts = qrc$rank

if (ncontrasts == 0)
stop("Need at least one none zero contrast")

Beta = 1:ncontrasts

if (ncontrasts < ncol(contrast))
contrast = contrast[, qrc$pivot[Beta]]

logFC = drop((Beta.full %*% contrast)/log(2))

Dvec = rep.int(1, nsamples)
Dvec[Beta] = diag(qrc$qr)[Beta]
Q = qr.Q(qrc, complete = TRUE, Dvec = Dvec)
design.full = design.full %*% Q
}

design.reduce = design.full[, -Beta, drop = FALSE]

if (ncol(design.reduce) == 1) {

result.reduce = glmfit.OneGroup(NanoStringData, m0, sigma, phi.hat)

} else {

result.reduce = glmfit.reduce(NanoStringData, design.reduce, m0, sigma,
    phi.hat)

}

Beta.reduce = result.reduce$Beta.reduce
U.reduce = result.reduce$mean.reduce
df.reduce = result.reduce$df.reduce

V.reduce = sweep(U.reduce, 2, k, FUN = "*")

get.loglikelihood <- function(dat) {

y = dat[1:nsamples]
Ey = dat[nsamples + (1:nsamples)]
phi = dat[2 * nsamples + 1]

```

```
alpha = 1/phi
tmp1 = 1/(1 + Ey * phi)
tmp2 = 1 - tmp1
tmp2[tmp2==0] = 1e-08

item1 = function(yy) {
y_gi = yy[1]
lamda_gi = yy[2]
tmp2_gi = yy[3]
t = c(0:y_gi)
tmp33.t = exp(lgamma(t + alpha) + (y_gi - t) * log(lamda_gi) + t * log(
    tmp2_gi) -
lfactorial(t) - lfactorial(y_gi - t))
tmp33.tt = log(max(sum(tmp33.t), 1e-08))
}

tmp3 = apply(cbind(matrix(y, ncol = 1), matrix(lamda_i, ncol = 1), 
    matrix(tmp2,
ncol = 1)), 1, item1)

sum(tmp3) - nsamples * lgamma(alpha) + alpha * sum(log(tmp1)) - sum(
    lamda_i)
}

## compute likelihood under null
tmpl1 = cbind(Y, V.reduce, phi.hat)
l0 = apply(tmpl1, 1, get.loglikelihood)

## compute likelihood under alternative

tmpl2 = cbind(Y, V.full, phi.hat)
la = apply(tmpl2, 1, get.loglikelihood)
lr = -2 * (l0 - la)
lr[which(lr <= 0)] = 0
df = df.full - df.reduce
pval = 1 - pchisq(lr, df = df)
qval = p.adjust(pval, method = "BH")

if (length(Beta) == 1)
logFC <- as.vector(logFC)

table = data.frame(logFC = logFC, lr = lr, pvalue = pval, qvalue = qval)

list(table = table, dispersion = phi.hat, log.dispersion = log(phi.hat), 
    design.full = X.full,
design.reduce = design.reduce, Beta.full = Beta.full, mean.full = U.full
    ,
Beta.reduce = Beta.reduce, mean.reduce = U.reduce, m0 = m0, sigma = 
    sigma)
}
```

R function glmfit.full

```
glmfit.full <- function(NanoStringData, design.full) {
```

```
2
3 c = positiveFactor(NanoStringData)
4 d = housekeepingFactor(NanoStringData)
5 k = c * d
6 lamda_i = negativeFactor(NanoStringData)
7 Y = exprs(NanoStringData)
8 Y_n = sweep(Y, 2, lamda_i, FUN="-")
9 Y_nph = sweep(Y_n, 2, k, FUN="/")
10 Y_nph[Y_nph <= 0] = 0.1
11
12 nsamples = ncol(Y)
13 ngenes = nrow(Y)
14 nbeta = ncol(design.full)  # number of full parameters (Beta)
15
16 # Beta matrix from linear model, starting value for Betas in optim
17 Blm = matrix(NA, ngenes, nbeta)
18
19 for (i in 1:ngenes) {
20
21 model = lm(log(Y_nph[i, ]) ~ 0 + design.full)
22 Blm[i, ] = model$coefficients
23
24 }
25
26 U = exp(Blm %*% t(design.full))
27 V = sweep(U, 2, k, FUN="*")
28
29 phi.g = est.dispersion(Y, Y_nph, lamda_i, c, d)$phi
30
31 ii = rowMins(Y) > max(negativeControl(NanoStringData))
32
33 l = length(which(ii == TRUE))
34 if (l > 0) {
35
36 phi.g0 = phi.g[ii]
37 lphi.g0 = log(phi.g0)
38 m0 = median(lphi.g0, na.rm = TRUE)
39 sigma2.mar = (IQR(lphi.g0, na.rm = TRUE)/1.349)^2
40 # Here we borrow the idea to compute the base sigma for DSS The function
41 # compute.baseSigma borrow the idea from Hao Wu's function
42 # compute.baseSigma.nontrend in DSS Package
43 sigma2.base = compute.baseSigma(exp(m0), Y[ii, ], V[ii, ], nsamples)
44 sigma = sqrt(max(sigma2.mar - sigma2.base, 0.01))
45 } else {
46 cat("There is no data satisied that min of endo great than max
47 of negative control", "\n")
48 m0 = -2
49 sigma = 1
50 lphi.g0 = 10
51 }
52
53 max.phi = max(lphi.g0, 10, na.rm = TRUE)
54 max.mean = max(rowMeans(Y_nph))
55
```

```
get.phi <- function(dat) {
y = dat[1:nsamples]
Ey = dat[nsamples + (1:nsamples)]

obj = function(phi) {
alpha = 1/phi
tmp1 = 1/(1 + Ey * phi)
tmp2 = 1 - tmp1
tmp2[tmp2==0] = 1e-08

item1 = function(yy) {
y_gi = yy[1]
lamda_gi = yy[2]
tmp2_gi = yy[3]
t = c(0:y_gi)
com = matrix(700, length(t), 1)

tmp33.t = exp(rowMins(cbind(lgamma(t + alpha) + (y_gi - t) * log(lamda_
    gi) +
t * log(tmp2_gi) - lfactorial(t) - lfactorial(y_gi - t), com)))
tmp33.tt = log(max(sum(tmp33.t), 1e-08))

}
tmp3 = apply(cbind(matrix(y, ncol = 1), matrix(lamda_i, ncol = 1), 
    matrix(tmp2,
ncol = 1)), 1, item1)

-(sum(tmp3) - nsamples * lgamma(alpha) + alpha * sum(log(tmp1)) - ((log(
    phi) -
m0)^2)/(2 * (sigma^2)) - log(sigma) - sum(lamda_i))
}
return(optimize(obj, interval = c(0.005, max.phi))$minimum)
}

get.beta.full <- function(dat) {

n = nsamples
y = dat[1:n]
phi = dat[n + 1]
Bstart = dat[(n + 2):(n + 1 + nbeta)]

obj = function(beta) {

alpha = 1/phi
xb=beta %*% t(design.full)
xb[xb>700]=700            ## control upper band for exp operation
tmp1 = 1/(1 + exp(xb) * k * phi)
tmp2 = 1 - tmp1
tmp2[tmp2==0] = 1e-08

item1 = function(yy) {

y_gi = yy[1]
lamda_gi = yy[2]
```

```
tmp2_gi = yy[3]
t = c(0:y_gi)
com = matrix(700, length(t), 1)

tmp33.t = exp(rowMins(cbind(lgamma(t + alpha) + (y_gi - t) * log(lamda_
    gi) +
t * log(tmp2_gi) - lfactorial(t) - lfactorial(y_gi - t), com)))
tmp33.tt = log(max(sum(tmp33.t), 1e-08))
}

tmp3 = apply(cbind(matrix(y, ncol = 1), matrix(lamda_i, ncol = 1), 
    matrix(tmp2,
ncol = 1)), 1, item1)

-(sum(tmp3) - n * lgamma(alpha) + alpha * sum(log(tmp1)) - ((log(phi) -
m0)^2)/(2 * (sigma^2)) - log(sigma) - sum(lamda_i))
}

return(optim(Bstart, obj)$par)
}

id = c(1:ngenes)
Beta.full = matrix(0, ngenes, nbeta)
phi.full = rep(0, ngenes)

phi.s = apply(cbind(matrix(Y, ncol = nsamples), matrix(V, ncol = 
    nsamples)),
1, get.phi)
B.s = Blm
Y.t = Y

con11 = 1
con21 = 1

j = 0
while ((con11 >= 0.5 | con21 >= 0.001) & j < 50) {
j = j + 1
Beta = apply(cbind(matrix(Y.t, ncol = nsamples), matrix(phi.s, ncol = 1)
    ,
matrix(B.s, ncol = nbeta)), 1, get.beta.full)

xb=t(design.full %*% Beta)
xb[xb>700]=700                    ## control the upper band of exp 
    operation
U.t = exp(xb)
V.t = sweep(U.t, 2, k, FUN = "*")

phi.t = apply(cbind(matrix(Y.t, ncol = nsamples), matrix(V.t, ncol = 
    nsamples)),
1, get.phi)
con1 = rowMaxs(abs((B.s - t(Beta))/t(Beta)))
con2 = abs((phi.s - phi.t)/phi.s)
con11 = max(con1)
con21 = max(con2)
```

```
155
156 idx = which(con1 < 0.5 & con2 < 0.001)
157
158 if (!length(idx) == 0) {
159 Beta.full[id[idx], ] = t(Beta)[idx, ]
160 phi.full[id[idx]] = phi.t[idx]
161 }
162
163 phi.s = phi.t
164 B.s = t(Beta)
165
166 if (!length(idx) == 0 & !length(idx) == length(id)) {
167 Y.t = Y.t[-idx, ]
168 phi.s = phi.s[-idx]
169 B.s = B.s[-idx, ]
170 id = id[-idx]
171 }
172
173 }
174
175 if (j == 50 & !length(idx) == length(id)) {
176
177 if(length(idx) == 0) {
178 Beta.full[id, ] = t(Beta)
179 phi.full[id] = phi.t
180 } else {
181 Beta.full[id, ] = t(Beta)[-idx, ]
182 phi.full[id] = phi.t[-idx]
183 }}
184 }
185
186 U.full = exp(Beta.full %*% t(design.full))
187 V.full = sweep(U.full, 2, k, FUN = "*")
188 eta = log(phi.full)
189 list(Beta.full = Beta.full, design = design.full, dispersion = phi.full,
      log.dispersion = eta,
190 m0 = m0, sigma = sigma, df.full = nbeta, mean.full = U.full, nineration
      = j)
191 }
```

R function glmfit.reduce

```
1 glmfit.reduce <- function(NanoStringData, design.reduce, m0, sigma, phi)
    {
2
3 c = positiveFactor(NanoStringData)
4 d = housekeepingFactor(NanoStringData)
5 k = c * d
6 lamda_i = negativeFactor(NanoStringData)
7 Y = exprs(NanoStringData)
8 Y_n = sweep(Y, 2, lamda_i, FUN = "-")
9 Y_nph = sweep(Y_n, 2, k, FUN = "/")
10 Y_nph[Y_nph <= 0] = 0.1
11
```

```
nsamples = ncol(Y)
ngenes = nrow(Y)
nbeta = ncol(design.reduce)  # number of parameters (Beta)

# Beta matrix from linear model, starting value for Betas in optim
Blm = matrix(NA, ngenes, nbeta)

for (i in 1:ngenes) {

model = lm(log(Y_nph[i, ]) ~ 0 + design.reduce)
Blm[i, ] = model$coefficients

}

get.beta.reduce <- function(dat) {

n = nsamples
y = dat[1:n]
phi = dat[n + 1]
Bstart = dat[(n + 2):(n + 1 + nbeta)]

obj = function(beta) {

alpha = 1/phi
xb=beta %*% t(design.reduce)
xb[xb>700]=700                              ## control upper band for exp operation
tmp1 = 1/(1 + exp(xb) * k * phi)
tmp2 = 1 - tmp1
tmp2[tmp2==0] = 1e-08

item1 = function(yy) {

y_gi = yy[1]
lamda_gi = yy[2]
tmp2_gi = yy[3]
t = c(0:y_gi)
com = matrix(700, length(t), 1)

tmp33.t = exp(rowMins(cbind(lgamma(t + alpha) + (y_gi - t) * log(lamda_gi) +
t * log(tmp2_gi) - lfactorial(t) - lfactorial(y_gi - t), com)))
tmp33.tt = log(max(sum(tmp33.t), 1e-08))
}

tmp3 = apply(cbind(matrix(y, ncol = 1), matrix(lamda_i, ncol = 1), matrix(tmp2,
ncol = 1)), 1, item1)

-(sum(tmp3) - n * lgamma(alpha) + alpha * sum(log(tmp1)) - ((log(phi) -
m0)^2)/(2 * (sigma^2)) - log(sigma) - sum(lamda_i))
}

return(optim(Bstart, obj)$par)
```

```
}

Beta.reduce = apply(cbind(matrix(Y, ncol = nsamples), matrix(phi, ncol = 1),
matrix(Blm, ncol = nbeta)), 1, get.beta.reduce)
U.reduce = exp(t(design.reduce %*% Beta.reduce))
V.reduce = sweep(U.reduce, 2, k, FUN = "*")

list(Beta.reduce = t(Beta.reduce), mean.reduce = U.reduce, dispersion = phi,
df.reduce = nbeta)
}
```

R function glmfit.OneGroup

```
glmfit.OneGroup <- function(NanoStringData, m0, sigma, phi) {

c = positiveFactor(NanoStringData)
d = housekeepingFactor(NanoStringData)
k = c * d
lamda_i = negativeFactor(NanoStringData)
Y = exprs(NanoStringData)
Y_n = sweep(Y, 2, lamda_i, FUN = "-")
Y_nph = sweep(Y_n, 2, k, FUN = "/")
Y_nph[Y_nph <= 0] = 0.1

n = ncol(Y)
max.mean = max(rowMeans(Y_nph))

get.mu <- function(dat) {

y = dat[1:n]
phi = dat[n + 1]
obj = function(mu) {
alpha = 1/phi
tmp1 = 1/(1 + mu * k * phi)
tmp2 = 1 - tmp1
tmp2[tmp2==0] = 1e-08

item1 = function(yy) {
y_gi = yy[1]
lamda_gi = yy[2]
tmp2_gi = yy[3]
t = c(0:y_gi)
com = matrix(700, length(t), 1)

tmp33.t = exp(rowMins(cbind(lgamma(t + alpha) + (y_gi - t) * log(lamda_gi) +
t * log(tmp2_gi) - lfactorial(t) - lfactorial(y_gi - t), com)))
tmp33.tt = log(max(sum(tmp33.t), 1e-08))
}

tmp3 = apply(cbind(matrix(y, ncol = 1), matrix(lamda_i, ncol = 1), matrix(tmp2,
```

```
ncol = 1)), 1, item1)

-(sum(tmp3) - n * lgamma(alpha) + alpha * sum(log(tmp1)) - ((log(phi) -
m0)^2)/(2 * (sigma^2)) - log(sigma) - sum(lamda_i))
}

return(optimize(obj, interval = c(0.1, max.mean))$minimum)
}

mu = apply(cbind(matrix(Y, ncol = n), matrix(phi, ncol = 1)), 1, get.mu)

Beta.reduce = log(mu)
U.reduce = matrix(rep(mu, n), ncol = n)
V.reduce = sweep(U.reduce, 2, k, FUN = "*")
eta = log(phi)

list(Beta.reduce = Beta.reduce, mean.reduce = U.reduce, dispersion = phi
    , df.reduce = 1)
}
```

Main function in the third project

R function MutDiff

```
MutDiff<-function(Datalist1, Datalist2){

n1=ncol(Datalist1$mut.base)-2
n2=ncol(Datalist2$mut.base)-2
nc=n1+n2
m=2*nc

#####################################################################
#####################################################################
####Make data sets for analysis: data.mut.all, data.cov.all##
#####################################################################

datalist1=CleanData1(Datalist1)
datalist2=CleanData1(Datalist2)

data=prepare.data(datalist1, datalist2)
data.mut.all=data$data.mut.all
data.cov.all=data$data.cov.all
gene=data$gene
categ=data$categ

#################################################
#################################################
####make full design matrix###################

group=c(rep(0,n1),rep(1,n2))
group=as.factor(group)
X0=model.matrix(~group)
```

```
X1=rbind(X0,X0)
Z=c(rep(0,(n1+n2)),rep(1,(n1+n2)))   ##0 indicate base and 1 indicate
    nonsilent
X2=Z*X1
design.full=cbind(X1,X2)

#############################################################################
#############################################################################
##########Make data sets and calculate prior mean and variavtion###
#############################################################################

datalist3=CleanData2(Datalist1)
datalist4=CleanData2(Datalist2)

coeff0=prepare.prior(datalist3,datalist4,design.full)

gamma=coeff0[,1]
dgamma=coeff0[,2]
beta=coeff0[,3]
dbeta=coeff0[,4]

gamma.mean=mean(gamma)
#gamma.sd=sd(gamma)
dgamma.mean=mean(dgamma)
#dgamma.sd=sd(dgamma)
beta.mean=mean(beta)
#beta.sd=sd(beta)
dbeta.mean=mean(dbeta)
#dbeta.sd=sd(dbeta)

gamma.sigma=quantile(gamma,0.95)/qnorm(0.95,gamma.mean,1)
dgamma.sigma=quantile(dgamma,0.95)/qnorm(0.975,dgamma.mean,1)
#dgamma.sigma=quantile(dgamma,0.95)/qnorm(0.975,0,1)

beta.sigma=quantile(beta,0.95)/qnorm(0.95,beta.mean,1)
dbeta.sigma=quantile(dbeta,0.95)/qnorm(0.975,dbeta.mean,1)
#dbeta.sigma=quantile(dbeta,0.95)/qnorm(0.975,0,1)

################################################
################################################
##get prior for dispersion parameter f####
datalist5=CleanData3(Datalist1)
datalist6=CleanData3(Datalist2)

nsim=1
hyper=EstimateHyper(datalist5,datalist6,nsim)
m0.base=hyper$base[1]
tao.base=hyper$base[2]
m0.non=hyper$nonsilent[1]
tao.non=hyper$nonsilent[2]

################################################
################################################

```

```
data.base.mut=data.mut.all[,1:(n1+n2)]
data.base.cov=data.cov.all[,1:(n1+n2)]

data.nonsilent.mut=data.mut.all[,(n1+n2+1):(2*n1+2*n2)]
data.nonsilent.cov=data.cov.all[,(n1+n2+1):(2*n1+2*n2)]

U.base=matrix(rep(rowSums(data.base.mut)/rowSums(data.base.cov),nc),ncol
    =nc)

##get dispersion parameter for base data##
tmp=cbind(data.base.mut,data.base.cov,U.base)
s=apply(tmp,1,get.skrinkage.s,nc=nc,m0=m0.base,tao=tao.base)
f.base=1/(1+exp(s))

#####################################################################
#####################################################################
U.non=matrix(rep(rowSums(data.nonsilent.mut)/rowSums(data.nonsilent.cov)
    ,nc),ncol=nc)

##get dispersion parameter for nonsilent data##
tmp=cbind(data.nonsilent.mut,data.nonsilent.cov,U.non)
s=apply(tmp,1,get.skrinkage.s,nc=nc,m0=m0.non,tao=tao.non)
f.nonsilent=1/(1+exp(s))

f=cbind(matrix(rep(f.base,nc),ncol=nc),matrix(rep(f.nonsilent,nc),ncol=
    nc))

#####################################################################
#####################################################################
###first time to get coefficient under alternative############

get.coeff.a<-function(dat){

Y=dat[1:m]
N=dat[(m+1):(2*m)]
f=dat[(2*m+1):(3*m)]

obj<-function(coeff){
gamma=coeff[1]
dgamma=coeff[2]
beta=coeff[3]
dbeta=coeff[4]
U=exp(coeff%*%t(design.full))/(1+exp(coeff%*%t(design.full)))
V=(1-f)/f
llog=-(sum(lbeta(Y+V*U,N-Y+(1-U)*V)-lbeta(U*V,(1-U)*V))
-(gamma-gamma.mean)^2/(2*gamma.sigma^2)
-(dgamma)^2/(2*dgamma.sigma^2)
-(beta-beta.mean)^2/(2*beta.sigma^2)
-(dbeta)^2/(2*dbeta.sigma^2))
print(llog)
}

starts=c(gamma.mean,dgamma.mean,beta.mean,dbeta.mean)
return(optim(starts,obj)$par)
```

```
}

tmp=cbind(data.mut.all,data.cov.all,f)

coeff.a=t(apply(tmp,1,get.coeff.a))

u1=exp(coeff.a[,1])/(1+exp(coeff.a[,1]))
U1=matrix(rep(u1,n1),ncol=n1)

u2=exp(coeff.a[,1]+coeff.a[,2])/(1+exp(coeff.a[,1]+coeff.a[,2]))
U2=matrix(rep(u2,n2),ncol=n2)

u3=exp(coeff.a[,1]+coeff.a[,3])/(1+exp(coeff.a[,1]+coeff.a[,3]))
U3=matrix(rep(u3,n1),ncol=n1)

u4=exp(coeff.a[,1]+coeff.a[,2]+coeff.a[,3]+coeff.a[,4])/(1+exp(coeff.a
    [,1]+coeff.a[,2]+coeff.a[,3]+coeff.a[,4]))
U4=matrix(rep(u4,n2),ncol=n2)
############################################################
############################################################
## Get f using new mean parameters ####################
U.base=cbind(U1,U2)
## get dispersion parameter for base data##
tmp=cbind(data.base.mut,data.base.cov,U.base)
s=apply(tmp,1,get.skrinkage.s,nc=nc,m0=m0.base,tao=tao.base)
f.base=1/(1+exp(s))


U.non=cbind(U3,U4)
## get dispersion parameter for nonsilent data##
tmp=cbind(data.nonsilent.mut,data.nonsilent.cov,U.non)
s=apply(tmp,1,get.skrinkage.s,nc=nc,m0=m0.non,tao=tao.non)
f.nonsilent=1/(1+exp(s))

f=cbind(matrix(rep(f.base,nc),ncol=nc),matrix(rep(f.nonsilent,nc),ncol=
    nc))

#################################################################
#################################################################
###Second time to get coefficient under alternative ###########
tmp=cbind(data.mut.all,data.cov.all,f)

coeff.a=t(apply(tmp,1,get.coeff.a))

#################################################################
#################################################################
## estimate coefficients under null ############################

design.reduce=cbind(X1,Z)

get.coeff.n<-function(dat){

Y=dat[1:m]
N=dat[(m+1):(2*m)]
```

```
f=dat[(2*m+1):(3*m)]

obj<-function(coeff){
gamma=coeff[1]
dgamma=coeff[2]
beta=coeff[3]
U=exp(coeff%*%t(design.reduce))/(1+exp(coeff%*%t(design.reduce)))
V=(1-f)/f
llog=-(sum(lbeta(Y+V*U,N-Y+(1-U)*V)-lbeta(U*V,(1-U)*V))
-(gamma-gamma.mean)^2/(2*gamma.sigma^2)
-(dgamma)^2/(2*dgamma.sigma^2)
-(beta-beta.mean)^2/(2*beta.sigma^2))
print(llog)
}

starts=c(gamma.mean,dgamma.mean,beta.mean)
return(optim(starts,obj)$par)
}

tmp=cbind(data.mut.all,data.cov.all,f)

coeff.n=t(apply(tmp,1,get.coeff.n))

##################################################
##################################################
get.likelihood<-function(dat){

l=length(dat)
Y=dat[1:m]
N=dat[(m+1):(2*m)]
f=dat[(2*m+1):(3*m)]
coeff=dat[(3*m+1):l]

U=exp(coeff%*%t(X))/(1+exp(coeff%*%t(X)))
V=(1-f)/f
sum(lbeta(Y+V*U,N-Y+(1-U)*V)-lbeta(U*V,(1-U)*V))

}

X=design.full
tmp=cbind(data.mut.all,data.cov.all,f,coeff.a)
llog.a=apply(tmp,1,get.likelihood)
##################################################
X=design.reduce
tmp=cbind(data.mut.all,data.cov.all,f,coeff.n)
llog.n=apply(tmp,1,get.likelihood)
##################################################

lr = -2 * (llog.n - llog.a)

length(which(lr<0))
lr[which(lr<0)]=0
pval = 1 - pchisq(lr, df = 1)
qval = p.adjust(pval, method = "BH")
```

```
239
240 data1.base.cov=ceiling(rowMeans(data.cov.all[,1:n1]))
241 data2.base.cov=ceiling(rowMeans(data.cov.all[,(n1+1):(n1+n2)]))
242
243 data1.nonsilent.cov=ceiling(rowMeans(data.cov.all[,(n1+n2+1):(2*n1+n2)])
      )
244 data2.nonsilent.cov=ceiling(rowMeans(data.cov.all[,(2*n1+n2+1):(2*n1+2*
      n2)]))
245
246 data1.base.mut=rowSums(data.mut.all[,1:n1])
247 data2.base.mut=rowSums(data.mut.all[,(n1+1):(n1+n2)])
248
249 data1.nonsilent.mut=rowSums(data.mut.all[,(n1+n2+1):(2*n1+n2)])
250 data2.nonsilent.mut=rowSums(data.mut.all[,(2*n1+n2+1):(2*n1+2*n2)])
251
252 table.position=data.frame(genename=gene,category=categ,pvalue=pval,
      qvalue=qval,lr=lr,
253 f.base=f.base,f.nonsilent=f.nonsilent,
254 v.base=(1-f.base)/f.base,v.non=(1-f.nonsilent)/f.nonsilent,
255 coeffa=coeff.a,␣coeffn=coeff.n,
256 base.cov1=data1.base.cov,␣base.cov2=data2.base.cov,
257 nonsilent.cov1=data1.nonsilent.cov,
258 nonsilent.cov2=data2.nonsilent.cov,
259 base.mut1=data1.base.mut,base.mut2=data2.base.mut,
260 nonsilent.mut1=data1.nonsilent.mut,
261 nonsilent.mut2=data2.nonsilent.mut)
262 index=order(table.position["qvalue"],table.position["pvalue"])
263 table.position=table.position[index,]
264
265 ##########################################################
266 ##########################################################
267 ##########################################################
268 df=get.df(gene)
269 genename=unique(gene)
270
271 llog.ga=PositionSum(llog.a,df)
272 llog.gn=PositionSum(llog.n,df)
273
274 lr.g=-2*(llog.gn-llog.ga)
275 length(which(lr.g<0))
276 lr.g[which(lr.g<0)]=0
277 pval.g␣=␣1␣-␣pchisq(lr.g,␣df␣=␣df)
278 qval.g␣=␣p.adjust(pval.g,␣method␣=␣"BH")
279
280 table.gene=data.frame(pvalue=pval.g,qvalue=qval.g,lr=lr.g)
281 row.names(table.gene)=genename
282
283 index=order(table.gene["qvalue"],table.gene["pvalue"])
284 table.gene=table.gene[index,]
285
286 return(list(table.gene=table.gene,␣table.position=table.position))
287
288 }
```

Bibliography

Agrawal, R., Pandey, P., Jha, P., Dwivedi, V., Sarkar, C., and Kulshreshtha, R. (2014). Hypoxic signature of micrornas in glioblastoma: insights from small rna deep sequencing. *BMC genomics*, 15(1):686.

Alexandrov, L. B., Nik-Zainal, S., Wedge, D. C., Aparicio, S. A., Behjati, S., Biankin, A. V., Bignell, G. R., Bolli, N., Borg, A., Børresen-Dale, A.-L., et al. (2013). Signatures of mutational processes in human cancer. *Nature*, 500(7463):415–421.

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

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300.

Brumbaugh, C. D., Kim, H. J., Giovacchini, M., and Pourmand, N. (2011). Nanostride: normalization and differential expression analysis of nanostring ncounter data. *BMC bioinformatics*, 12(1):479.

Busskamp, V., Lewis, N. E., Guye, P., Ng, A. H., Shipman, S. L., Byrne, S. M., Sanjana, N. E., Murn, J., Li, Y., Li, S., et al. (2014). Rapid neurogenesis through transcriptional activation in human stem cells. *Molecular systems biology*, 10(11).

Chen, G., Shen, Z.-L., Wang, L., Lv, C.-Y., Huang, X.-E., and Zhou, R.-P. (2013). Hsa-mir-181a-5p expression and effects on cell proliferation in gastric cancer. *Asian Pac J Cancer Prev*, 14:3871–3875.

Collins, F. S. and Barker, A. D. (2007). Mapping the cancer genome. *Scientific American*, 296(3):50–57.

Cui, Q. (2010). A network of cancer genes with co-occurring and anti-co-occurring mutations. *PLoS One*, 5(10):e13180.

Doroshow, J. (2014). National cancer institute's precision medicine initiatives for the new national clinical trials network. American Society of Clinical Oncology.

Ellis, M. J., Ding, L., Shen, D., Luo, J., Suman, V. J., Wallis, J. W., Van Tine, B. A., Hoog, J., Goiffon, R. J., Goldstein, T. C., et al. (2012). Whole-genome analysis informs breast cancer response to aromatase inhibition. *Nature*, 486(7403):353–360.

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

Hudson, T. J., Anderson, W., Aretz, A., Barker, A. D., Bell, C., Bernabé, R. R., Bhan, M., Calvo, F., Eerola, I., Gerhard, D. S., et al. (2010). International network of cancer genome projects. *Nature*, 464(7291):993–998.

Lag, R., Harkins, D., Krapcho, M., Mariotto, A., Miller, B., Feuer, E., Clegg, L., Eisner, M., Horner, M., Howlader, N., et al. (1975). Seer cancer statistics review. *Bethesda, National Cancer Institute*, pages 1975–2003.

Lawrence, M. S., Stojanov, P., Polak, P., Kryukov, G. V., Cibulskis, K., Sivachenko, A., Carter, S. L., Stewart, C., Mermel, C. H., Roberts, S. A., et al. (2013). Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature*, 499(7457):214–218.

Leiserson, M. D., Blokh, D., Sharan, R., and Raphael, B. J. (2013). Simultaneous identification of multiple driver pathways in cancer. *PLoS Comput Biol*, 9(5):e1003054.

Leiserson, M. D., Wu, H.-T., Vandin, F., and Raphael, B. J. (2015). Comet: a statistical approach to identify combinations of mutually exclusive alterations in cancer. *Genome biology*, 16(1):1.

Liang, H., Cheung, L. W., Li, J., Ju, Z., Yu, S., Stemke-Hale, K., Dogruluk, T., Lu, Y., Liu, X., Gu, C., et al. (2012). Whole-exome sequencing combined with functional genomics reveals novel candidate driver cancer genes in endometrial cancer. *Genome research*, 22(11):2120–2129.

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

Ma, J., Yao, Y., Wang, P., Liu, Y., Zhao, L., Li, Z., Li, Z., and Xue, Y. (2014). Mir-152 functions as a tumor suppressor in glioblastoma stem cells by targeting krüppel-like factor 4. *Cancer letters*, 355(1):85–95.

McLendon, R., Friedman, A., Bigner, D., Van Meir, E. G., Brat, D. J., Mastrogianakis, G. M., Olson, J. J., Mikkelsen, T., Lehman, N., Aldape, K., et al. (2008). Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455(7216):1061–1068.

Mori, M., Triboulet, R., Mohseni, M., Schlegelmilch, K., Shrestha, K., Camargo, F. D., and Gregory, R. I. (2014). Hippo signaling regulates microprocessor and links cell-density-dependent mirna biogenesis to cancer. *Cell*, 156(5):893–906.

Network, C. G. A. et al. (2015). Comprehensive genomic characterization of head and neck squamous cell carcinomas. *Nature*, 517(7536):576–582.

Network, C. G. A. R. et al. (2012). Comprehensive genomic characterization of squamous cell lung cancers. *Nature*, 489(7417):519–525.

Network, C. G. A. R. et al. (2014). Comprehensive molecular profiling of lung adenocarcinoma. *Nature*, 511(7511):543–550.

Ouyang, M., Li, Y., Ye, S., Ma, J., Lu, L., Lv, W., Chang, G., Li, X., Li, Q., Wang, S., et al. (2014). Microrna profiling implies new markers of chemoresistance of triple-negative breast cancer. *PloS one*, 9(5):e96228.

Reis, P. P., Waldron, L., Goswami, R. S., Xu, W., Xuan, Y., Perez-Ordonez, B., Gullane, P., Irish, J., Jurisica, I., and Kamel-Reid, S. (2011). mrna transcript quantification in archival samples using multiplexed, color-coded probes. *BMC biotechnology*, 11(1):1.

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

Robinson, M. D. and Smyth, G. K. (2007). Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 23(21):2881–2887.

Robinson, M. D. and Smyth, G. K. (2008). Small-sample estimation of negative binomial dispersion, with applications to sage data. *Biostatistics*, 9(2):321–332.

Slattery, M. L., Herrick, J. S., Mullany, L. E., Valeri, N., Stevens, J., Caan, B. J., Samowitz, W., and Wolff, R. K. (2014). An evaluation and replication of mirnas with disease stage and colorectal cancer-specific mortality. *International Journal of Cancer*.

Stephens, P. J., Davies, H. R., Mitani, Y., Van Loo, P., Shlien, A., Tarpey, P. S., Papaemmanuil, E., Cheverton, A., Bignell, G. R., Butler, A. P., et al. (2013). Whole exome sequencing of adenoid cystic carcinoma. *The Journal of clinical investigation*, 123(7):2965–2968.

Stransky, N., Egloff, A. M., Tward, A. D., Kostic, A. D., Cibulskis, K., Sivachenko, A., Kryukov, G. V., Lawrence, M. S., Sougnez, C., McKenna, A., et al. (2011). The mutational landscape of head and neck squamous cell carcinoma. *Science*, 333(6046):1157–1160.

Sun, E., Zhou, Q., Liu, K., Wei, W., Wang, C., Liu, X., Lu, C., and Ma, D. (2014). Screening mirnas related to different subtypes of breast cancer with mirnas microarray. *European review for medical and pharmacological sciences*, 18(19):2783–2788.

Szczurek, E. and Beerenwinkel, N. (2014). Modeling mutual exclusivity of cancer mutations. *PLoS Comput Biol*, 10(3):e1003503.

Tamborero, D., Gonzalez-Perez, A., Perez-Llamas, C., Deu-Pons, J., Kandoth, C., Reimand, J., Lawrence, M. S., Getz, G., Bader, G. D., Ding, L., et al. (2013). Comprehensive identification of mutational cancer driver genes across 12 tumor types. *Scientific reports*, 3:2650.

Tao, K., Yang, J., Guo, Z., Hu, Y., Sheng, H., Gao, H., and Yu, H. (2014). Prognostic value of mir-221-3p, mir-342-3p and mir-491-5p expression in colon cancer. *American journal of translational research*, 6(4):391.

Teruel-Montoya, R., Kong, X., Abraham, S., Ma, L., Kunapuli, S. P., Holinstat, M., Shaw, C. A., McKenzie, S. E., Edelstein, L. C., and Bray, P. F. (2014). Microrna expression differences in human hematopoietic cell lineages enable regulated transgene expression. *PloS one*, 9(7):e102259.

Waggott, D., Chu, K., Yin, S., Wouters, B. G., Liu, F.-F., and Boutros, P. C. (2012). Nanostringnorm: an extensible r package for the pre-processing of nanostring mrna and mirna data. *Bioinformatics*, 28(11):1546–1548.

Wang, L., Tsutsumi, S., Kawaguchi, T., Nagasaki, K., Tatsuno, K., Yamamoto, S., Sang, F., Sonoda, K., Sugawara, M., Saiura, A., et al. (2012). Whole-exome sequencing of human pancreatic cancers and characterization of genomic instability caused by mlh1 haploinsufficiency and complete deficiency. *Genome research*, 22(2):208–219.

Wang, Z., Bao, Z., Yan, W., You, G., Wang, Y., Li, X., and Zhang, W. (2013). Isocitrate dehydrogenase 1 (idh1) mutation-specific microrna signature predicts favorable prognosis in glioblastoma patients with idh1 wild type. *J Exp Clin Cancer Res*, 32(1):59.

Wu, H., Wang, C., and Wu, Z. (2013). A new shrinkage estimator for dispersion improves differential expression detection in rna-seq data. *Biostatistics*, 14(2):232–243.

Xie, J., Tan, Z.-H., Tang, X., Mo, M.-S., Liu, Y.-P., Gan, R.-L., Li, Y., Zhang, L., and Li, G.-Q. (2014). mir-374b-5p suppresses reck expression and promotes gastric cancer cell invasion and metastasis. *World journal of gastroenterology: WJG*, 20(46):17439.

Xu, S., Wei, J., Wang, F., Kong, L.-Y., Ling, X.-Y., Nduom, E., Gabrusiewicz, K., Doucette, T., Yang, Y., Yaghi, N. K., et al. (2014). Effect of mir-142-3p on the m2 macrophage and therapeutic efficacy against murine glioblastoma. *Journal of the National Cancer Institute*, 106(8):dju162.

Yang, F., Wang, W., Zhou, C., Xi, W., Yuan, L., Chen, X., Li, Y., Yang, A., Zhang, J., and Wang, T. (2015). Mir-221/222 promote human glioma cell invasion and angiogenesis by targeting timp2. *Tumor Biology*, 36(5):3763–3773.

Yao, Y., Ma, J., Xue, Y., Wang, P., Li, Z., Liu, J., Chen, L., Xi, Z., Teng, H., Wang, Z., et al. (2015). Knockdown of long non-coding rna xist exerts tumor-suppressive functions in human glioblastoma stem cells by up-regulating mir-152. *Cancer letters*.

Zhang, C., Zhang, J., Hao, J., Shi, Z., Wang, Y., Han, L., Yu, S., You, Y., Jiang, T., Wang, J., et al. (2012). High level of mir-221/222 confers increased cell invasion and poor prognosis in glioma. *J Transl Med*, 10(19):1–11.

Zheng, X., Chopp, M., Lu, Y., Buller, B., and Jiang, F. (2013). Mir-15b and mir-152 reduce glioma cell invasion and angiogenesis via nrp-2 and mmp-3. *Cancer letters*, 329(2):146–154.

Vita

- Place of birth: Taicang, China

- Educational institutions attended and degrees already awarded

  PhD student in Statistics: University of Kentucky
  MS in Statistics: University of Kentucky
  MS in Management: Hohai University, China
  BS in Computer Science: Nanjing Normal University, China

- Professional publications in University of Kentucky:

  Hong Wang, Craig Horbinski, Hao Wu, Yinxing Liu, Shaoyi Sheng, Jinpeng Liu, Heidi Weiss, Arnold J. Stromberg and Chi Wang (2016). NanoStringDiff: A Novel Statistical Method for Differential Expression Analysis Based on NanoString nCounter Data, Nucleic Acids Research

  Experience in University of Kentucky
  Statistical Research Assistant: May, 2015 $\longrightarrow$ Present
  Teaching Assistant: Aug, 2011 $\longrightarrow$ May, 2015