A QUASI-LIKELIHOOD METHOD TO DETECT DIFFERENTIALLY EXPRESSED GENES IN RNA-SEQUENCE DATA

By

CHU-SHU GU

A Thesis Submitted to the

School of Graduate Studies

in Partial Fulfilment of the

Requirement for the Degree of

Doctor of Philosophy

McMaster University

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Acronym	Definition
APL	Adjusted Profile Likelihood
DE	Differentially Expressed
cDNA	Complementary Deoxyribonucleic Acid
EE	Equivalently Expressed
EMC	Experiments with Multiple Conditions
EV	Empty Vector
FDR	False Discovery Rate
FWER	Family-wise Error Rate
GC	Guanine-cytosine
GLM	Generalized Linear Model
LIMMA	Linear Models for Microarray Data
MISE	Mean Integrated Squared Error
NB	Negative Binomial
NGS	Next Generation High-throughput Sequencing Technology
PPFP	PAX8-PPARG Fusion Protein
qCML	Quantile-adjusted Conditional Maximum Likelihood
RLE	Relative Log Expression
RNA	Ribonucleic Acid
RNA-seq	RNA-sequencing
RPKM	Reads per Kilobase per Million Mapped Reads
TMM	Weighted Trimmed Mean of M-values
UCSC	University of California Santa Cruz
UQ	Upper Quartile

ACKNOWLEDGEMENTS

I am deeply indebted to my supervisor at McMaster University, Dr. Angelo Canty. In all stages of this thesis, he shared his expert knowledge, time and enthusiasm. I also wish to extend my appreciation to Drs. Mark Levine and Sonia Anand at McMaster University for their valuable comments and suggestions. I would also like to thank my colleague, Denise Julian, for her editing assistance. My family has been great during this period of time. Yu Guo, my wife, has given me all the love and support. I also dedicate this Ph.D thesis to my two lovely children, Evelyn and Eric, who are the pride and joy of my life.

ABSTRACT

In recent years, the RNA-sequencing (RNA-seq) method, which measures the transcriptome by counting short sequencing reads obtained by high-throughput sequencing, is replacing the microarray technology as the major platform in gene expression studies. The large amount of discrete data in RNA-seq experiments calls for effective analysis methods. In this dissertation, a new method to detect differentially expressed genes based on quasi-likelihood theory is developed in experiments with a completely randomized design with two experimental conditions. The proposed method estimates the variance function empirically and consequently it has similar sensitivities and FDRs across distributions with different variance functions. In a simulation study, the method is shown to have similar sensitivities and FDRs across the data with three different types of variance functions compared with some other popular methods. This method is applied to a real dataset with two experimental conditions along with some competing methods.

The new method is then extended to more complex designs such as an experiment with multiple experimental conditions, an experiment with block design and an experiment with factorial design. The same advantages for the new method have been found in simulation studies. This method and some competing methods are applied to three real datasets with complex designs.

The new method is also applied to analyze reads per kilobase per million mapped reads (RPKM) data. In the simulation, the method is compared with the Linear Models for Microarray Data (LIMMA) originally developed for microarray analysis (Smyth, 2004) and the question of normalization is also examined. It is shown that the new method

and the LIMMA method have similar performance. Further normalization is required for the proper analysis of the RPKM data and the best such normalization is the scaling method. Analyzing raw count data properly has better performance than analyzing the RPKM data. Different normalization and statistical methods are applied to a real dataset with varied gene length across samples.

CHAPTER 1. Introduction

1.1 RNA-seq

In the last two decades, the dominant platform to study gene expression has been the microarray. With the introduction of next generation high-throughput sequencing technology (NGS), gene expression can be studied by the RNA-seq method. In an RNA-seq experiment, ribonucleic acid (RNA) from a biological sample is converted to a library of complementary deoxyribonucleic acid (cDNA) fragments and then sequenced by a high-throughput sequencing platform such as Illumina's Genome Analyzer. Millions of short sequences (reads) are produced in this process. Typically, these reads are then mapped to a reference genome or transcriptome. The number of short reads mapped to a specific gene is a measure of that gene's expression level. Compared with microarray technology, RNA-seq has a wider dynamic range, less noise, and is a more accurate and reproducible method for quantifying gene expression levels (Ansorge, 2009; Metzker, 2010; Ozsolak and Milos, 2011). The gene expressions measured by microarray and RNA-seq have only moderate correlation (Wang et al., 2009).

An example of an RNA-seq dataset, from Squadrito et al. (2014), is shown in Table 1. In this study, mice bone marrow cells were plated in macrophage serum-free medium and cultured to allow macrophage differentiation. They were polarized by adding interleukin-4 to the medium or left untreated. The transcriptomes of the bone marrow-derived macrophages are profiled. In the dataset, there are four treated and four untreated samples. Each cell in Table 1 shows the count of short reads of a specific gene in a

Table 1: Snapshot of an RNA-seq Dataset

(Squadrito et al., 2014)

Ensembl* Gene ID	Mouse Samples							
	Treated				Untreated			
	1	2	3	4	1	2	3	4
ENSMUSG00000025903	1590	2174	1870	1999	2667	2750	2407	2760
ENSMUSG00000033813	746	1054	931	951	1041	1203	1076	1101
ENSMUSG00000033793	4951	5233	4356	4433	8448	9570	8160	9140
ENSMUSG00000025905	0	0	0	0	2	1	0	0
ENSMUSG00000025907	2403	3745	3320	3232	2995	3594	3223	3457
ENSMUSG00000087247	0	0	0	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

** Ensembl is a public genetic database which provides annotation for human, mouse and other genomes (Zerbi et al., 2015; Cunningham et al., 2015). The gene ID information is retrieved from Ensembl release 83.*

biological sample. In the remainder of this dissertation, ‘gene’ is used as a general term for gene, exon or transcript.

1.2 Detecting Differentially Expressed Genes

One of the primary objectives in RNA-seq experiments is to compare gene expression levels across different experimental conditions. Researchers are particularly interested in detecting those differentially expressed (DE) genes. A simple and common study design in RNA-seq experiments is a completely randomized design with two experimental conditions.

The samples in an RNA-seq experiment could be either technical replicates or biological replicates. Technical replicates usually occur when the same sample is divided and measured multiple times. Biological replicates occur when biologically distinct samples are measured. For RNA-seq experiments with technical replicates, the data are usually assumed to have a Poisson distribution. Fisher's exact test (Bloom et al., 2009), the goodness of fit test (Marioni et al., 2008) and the likelihood ratio test (Bullard et al., 2010) are appropriate in this case. For RNA-seq experiments with biological replicates, the data usually exhibit larger variability than data containing only technical replicates. This is often called overdispersion relative to Poisson. For inference about a population, biological replicates are essential and it is assumed in this dissertation that all replicates are biological replicates. "Most statisticians agree that the overdispersion problem needs to be addressed" in order to have a valid statistical test (Wu et al., 2013). The overdispersion is well known to be gene-specific (Pritchard et al., 2001). The negative binomial (NB) distribution has a quadratic mean-variance relation with a dispersion parameter to model this gene-specific overdispersion. Based on the NB assumption, several methods have been proposed, including two popular methods which have been implemented in the R packages edgeR (Robinson et al., 2010) and DESeq (Anders and Huber, 2010). In the edgeR method, the NB dispersion parameter is modeled in different ways and estimated from the data. It can be assumed to be a constant, to have a trend or be gene-specific. In the DESeq method, the NB dispersion parameter is estimated by a smooth curve such as local regression. Once the dispersion parameters are estimated, the statistical tests of DE for both the edgeR and DESeq methods are analogous to Fisher's exact test.

Some researchers argue that the gene-specific overdispersion in RNA-seq data has not been adequately accounted for even in an NB model. Esnaola et al. (2013) proposed a method based on a more general family of distributions for count data which is called the Poisson-Tweedie distribution. In this family of distributions, the variance is defined as the mean raised to a gene-specific power. With a more general or flexible distributional assumption, it fits the gene-specific overdispersion better. A more general distribution appears appealing but it is preferable to have a method without any distributional assumption. Furthermore, it is well known that most genes in RNA-seq data are overdispersed relative to Poisson; however, a smaller, but not negligible, number of genes are underdispersed relative to Poisson. For example, for the dataset presented in Table 1, approximately 9% of the gene-specific sample variances are smaller than their corresponding sample means after preprocessing. The preprocessing includes filtering those genes with zero interquartile range or those genes with low mean count (≤ 1), and then normalizing the rest of the raw count data by the weighted Trimmed Mean of M-values (TMM) method (Robinson and Oshlack, 2010).

There is also a non-parametric statistical method called SAMSeq (Li and Tibshirani, 2013) which does not assume any underlying distribution. The SAMSeq method uses the Wilcoxon rank statistic. A permutation-based approach is used to estimate the false discovery rate (FDR), which is the expected proportion of false positive findings. In a statistical method comparison study (Seyednasrollah et al., 2015), this method is recommended to be applied to RNA-seq data with at least a moderate number of replicates (about 10 or more). Rank-based as well as permutation-based tests have limitations when the sample size is small. In that case, there are a limited number of

unique rank combinations or permutations resulting in a low resolution of obtainable p-values (Di Bucchiano, 1999). The requirement of at least a moderate number of replicates makes SAMSeq not applicable in many RNA-seq studies.

A new quasi-likelihood approach (QuasiDE) is proposed in this dissertation to detect differentially expressed genes for RNA-seq data without assuming any underlying distribution. The methodology is developed in a simple and popular study design, which is a completely randomized design with two experimental conditions. In the theoretical framework of the QuasiDE method, the overdispersed and underdispersed data are all able to be modeled. Also, the QuasiDE method works well in the datasets with small sample size which is a limitation of the SAMSeq method.

The quasi-likelihood method is also used by Lund et al. (2012) in the RNA-seq data analysis. In their work, a linear or quadratic variance function is assumed. The F statistic based on quasi-likelihood theory is used, which is the quasi-likelihood ratio test statistic divided by the estimated quasi-likelihood dispersion parameter. To have more reliable dispersion estimates, two shrinkage methods were proposed using an empirical Bayes method which is similar to that used in the LIMMA method (Smyth, 2004). To evaluate the F statistic, the form of variance function needs to be known. With a linear variance function assumption, this is straightforward. With the quadratic variance function assumption, the gene-specific NB dispersion parameters are estimated using the edgeR method. The main differences between the QuasiDE and NBQLSpline methods are 1) the QuasiDE method estimates the variance function empirically and the NBQLSpline method assumes a quadratic variance function; 2) the QuasiDE method estimates the dispersion by the Pearson moment estimator (McCullagh, 1983) and NBQLSpline

estimates the dispersion by the shrinkage method. In simulation, the QuasiDE method is compared with Lund's recommended approach as well as with other popular approaches such as the edgeR and DESeq methods. The QuasiDE method is shown to have similar sensitivities and FDRs across the data with different forms of the variance function compared with the other methods. In addition, the data requirement of the QuasiDE method is not limited to the count data. It can also be applied to any continuous data, for example, to microarray data or to RPKM data.

1.3 QuasiDE in Other Designs

The QuasiDE method is developed in a completely randomized design with two experimental conditions. With a few modifications, it can be extended to be used in some other common study designs, namely a design with multiple experimental conditions, a block design or a multi-factorial design. With the extensions, the QuasiDE method offers flexibility for the purpose of testing differential expression in experiments with a complex design.

1.4 Analysis of the RPKM Data

The number of short reads mapped to a reference genome is known to be related to the library size and the gene length. Library size is the total number of short reads obtained in an RNA-seq experiment for a specific sample. Different library sizes would produce a proportionally different number of short reads from the same gene for the same sample. Longer genes tend to produce more fragments than shorter genes. In normalization, the impact of these factors needs to be removed or controlled. The widely used RPKM is a measure developed to correct the impact of these factors (Wagner et al., 2012). A number of important issues remain unclear, however. Among these are: 1) the proper

statistical method to detect DE genes based on the RPKM data; 2) whether further normalization of the RPKM data is necessary and if so, what type of normalization is needed in the DE analysis; and 3) the difference in the DE analysis using the RPKM data or using the raw count data. Two statistical methods are found applicable in the analysis of the RPKM data and compared: LIMMA by Smyth (2004) and the QuasiDE method. Seven normalization methods are found to be applicable in the analysis of the RPKM data and their effects are examined: TMM (Robinson and Oshlack, 2010), Relative Log Expression (RLE) (Anders and Huber, 2010), Upper Quartile (UQ) (Bullard et al., 2010), scaling (Affymetrix, 2001), quantile (Bolstad et al., 2003), cyclic loess (Yang et al., 2002) and invariant set (Li and Wong, 2001). To compare the analysis using the RPKM data and using the raw count data, three statistical methods are found to be applicable in the analysis of the raw count data in which the gene length and library size effects are taken into account: LIMMA, QuasiDE, and edgeR. It is shown in simulation that the LIMMA and QuasiDE methods have similar performance in the analysis of the RPKM data. The analysis of the RPKM data with the scaling normalization has the best controlled FDR among the seven normalization methods. The analysis of the raw count data with proper normalization is found to have a better performance than the analysis of the RPKM data.

1.5 Notation

Let y_{ijg} denote the number of read counts of the g^{th} gene for the j^{th} sample in the i^{th} experimental condition. Let z_{ijg} denote y_{ijg} after normalization. N_i denotes the number of biological samples in the i^{th} experimental condition. G, N and I denote the total numbers of genes, samples and experimental conditions respectively. $L_{ij} = \sum_{g=1}^{G} y_{ijg}$

denotes the library size of the j^{th} sample in the i^{th} experimental condition. $\bar{y}_g$ denotes the average raw count for the g^{th} gene. z_g denotes the normalized count vector for the g^{th} gene. $\bar{z}_g$ denotes the average of z_g and S_g^2 denotes the variance of z_g. $\bar{z}_{ig}$ denotes the average normalized count of the g^{th} gene in the i^{th} experimental condition and S_{ig}^2 denotes the corresponding sample variance.

1.6 Data Description

Six real RNA-seq datasets are used in this dissertation. The first dataset is from a study to detect guanine-cytosine (GC) responsive genes in airway smooth muscle cells (Himes et al., 2014). The dataset contains RNA-seq expression profiling of primary airway smooth muscle cells from four white male donors either treated with 1 μM dexamethasone or not treated for 18 hours. The first dataset is referred to as “Himes data” hereafter. The second dataset is from Squadrito et al. (2014) which is partly shown in Table 1. The bone marrow cells of eight-week old C57BL/6 mice were plated in macrophage serum-free medium and cultured for one week. The macrophages were either untreated or stimulated with interleukin-4. The transcriptomes of the bone marrow-derived macrophages are profiled. In this dataset, there are four treated samples and four untreated samples. The second dataset is referred to as “Squadrito data” hereafter. The third dataset is from Dolatshad et al. (2015). The transcriptomes of bone marrow CD34+ cells are compared among three groups: eight myelodysplastic syndrome patients with mutation in gene “SF3B1”, four myelodysplastic syndrome patients without known splicing mutation and five healthy patients as control. The third dataset is referred to as “Dolatshad data” hereafter. The fourth dataset is from Bottomly et al. (2011). The purpose of this study is to detect DE genes in mouse striatum

between two commonly used inbred mouse strains (labelled B6 and D2). This dataset contains RNA-seq profiling of 10 B6 and 11 D2 inbred mice. There are three separate experiments: two experiments both include three B2 and four D2 inbred mice and the third experiment includes four B2 and three D2 inbred mice. In the DE analysis, different experiments are treated as blocks. The fourth dataset is referred to as “Bottomly data” hereafter. In the fifth dataset, the oncogenic PAX8-PPARG fusion protein (PPFP) in thyroid carcinoma was studied (Zhang et al., 2015). PCCL3 rat thyroid cells are used to characterize the PPFP-dependent gene expression. The dataset contains six PPFP cells and six control empty vector (EV) cells. Half of these cells in each group are treated with pioglitazone. The fifth dataset is referred to as “Zhang data” hereafter. In the last dataset, the research interest is to detect those orthologs which have an interaction effect between species (Toxoplasma gondii and Neospora caninum) and time (Reid et al., 2012). Only those orthologs which have a one-to-one relationship between two species are used in the analysis. There are eight samples of Toxoplasma gondii and six samples of Neospora caninum in total. The gene expressions of each two Toxoplasma gondii samples were measured at day 2, 3, 4 and 6. The gene expressions of each two Neospora caninum samples were measured at day 3, 4 and 6. The sixth dataset is referred to as “Reid data” hereafter.

1.7 Thesis Organization

The main chapters of this dissertation focus on the three topics introduced in Section 1.2, 1.3 and 1.4. In Chapter 2, a new quasi-likelihood approach (QuasiDE) is presented to detect DE genes in a completely randomized design with two experimental conditions. In Chapter 3, the QuasiDE method is extended to be used in other common

study designs. In Chapter 4, the statistical analysis of the RPKM data is presented. A summary of my work and possible directions for future research is discussed briefly in Chapter 5.

CHAPTER 2. A New Quasi-likelihood Approach to Detect Differentially Expressed Genes

2.1 Introduction

The recent advent of the NGS technology has revolutionized gene expression studies. The study of gene expression through sequencing (RNA-seq) is an important application of the NGS technology. In an RNA-seq experiment, the RNAs in a biological sample are converted to a library of complementary DNA fragments and then sequenced by the NGS technology. Millions of short sequences or reads are produced and then mapped to a reference genome or transcriptome, or assembled de novo. Those unmapped short reads are usually discarded. The count of reads mapped to a given gene, exon or transcript measures the expression level for this region of the transcriptome. In the statistical analysis of the RNA-seq data, detecting DE genes across experimental conditions is commonly the major goal. Differences in gene expression level have been found to be relevant to many diseases. For example, the subtypes of breast cancers were found to have different expression patterns (Sørlie et al., 2001).

2.2 Preprocessing: Filtering and Normalization

Before the formal statistical analysis of the RNA-seq data, the raw count data are usually preprocessed by some filtering and normalization methods.

2.2.1 Filtering

The filtering method is designed to remove less informative genes such as those with very little or no variation across samples or those with low expression. From a biological

perspective, “a gene must be expressed at some minimal level before it is likely to be translated into a protein or to be biologically important.” (Chen et al., 2016). After filtering and normalization, a statistical test is typically applied to one gene at a time to detect which genes show differential expression. There are a large number of statistical tests, due to a large number of genes. In genome-wide gene expression analysis, it is popular to use the false discovery rate (FDR) as the overall error rate to control. FDR is defined as the expected proportion of false positives among all significant hypotheses. If there are no positives, this proportion is defined to be zero. The use of filtering usually helps achieve better estimated FDR, due to a smaller multiple testing penalty and less impact from irrelevant genes in FDR estimation. There is no universal gold standard filtering method in the RNA-seq DE analysis. To remove low expressed genes, the method proposed by Sultan et al. (2008) is adopted to remove those genes with low mean count (≤ 1). To remove those genes with low variability, the variance filter (zero interquartile range) recommended by Morgan (2014) is used. In this and the next chapter, a filtering method is used in which genes with zero interquartile range or low mean count (≤ 1) will be considered to be equivalently expressed (EE) in different experimental conditions and therefore removed from the subsequent DE analysis.

2.2.2 Normalization

To ensure accurate inference of gene expression levels, normalization of RNA-seq data is essential (Risso et al., 2014). A number of factors were found to be relevant, and these include the between-sample differences such as library size (Mortazavi et al., 2008) and library composition (Robinson and Oshlack, 2010), as well as within-sample gene-specific properties such as gene length (Oshlack and Wakefield, 2009) and GC

content (Pickrell et al., 2010). For example, a larger library size for a biological sample results in proportionally larger read counts for all genes in this sample. Consequently, there are systematic differences in read counts from the same gene among the biological samples due to different library sizes. Another factor is the library composition. When a small number of genes are highly expressed in one sample but not in another, they can consume a substantial proportion of the total library size, causing the remaining genes to be under-sampled. The remaining genes may appear to be down-regulated. The aim of normalization is to remove or control the effect from these factors. For example, the GC content is adjusted in a normalization method developed by Risso et al. (2011). Normalization may also be related to the statistical method in the DE analysis. If the effect of these factors is cancelled out in the DE analysis, normalizing by that factor is unnecessary. For example, the gene length is typically the same across samples if only one species is studied. In that case, it is unnecessary to consider normalizing by the gene length. Provided the experiment is well designed, technical effects unrelated to the experimental conditions are usually cancelled out in the DE analysis. There are two important assumptions for most normalization methods: 1) most genes are not DE genes; and 2) the proportion of over- and under-expression for those DE genes is approximately balanced (Bolstad et al., 2003; Yang et al., 2002). Dillies et al. (2012) stated that “these assumptions appear reasonable in many studies”. In this dissertation, these two assumptions are also used in simulating the RNA-seq data. In a comprehensive evaluation of normalization methods for RNA-seq data (Dillies et al., 2012), the TMM method and the RLE method were found to be the two best methods of seven normalization methods under

evaluation. These two methods “are robust to the presence of different library sizes and widely different library compositions” (Dillies et al., 2012). They are able to control FDR while maintaining the power. In this and the next chapter, the TMM method is used as the normalization method in analyzing the simulated RNA-seq datasets.

2.2.3 Weighted Trimmed Mean of M-values (TMM)

The assumption of this method (Robinson and Oshlack, 2010) is that most genes are not DE genes. One of the observed samples, whose upper quartile is closest to the mean of upper quartiles of all observed samples, is set to be the reference sample. The expression values are first divided by their library sizes. M-values (log-2 expression ratios) and A-values (average log-2 expression values) for these adjusted expression values are calculated between the sample to be normalized and the reference sample. Normalization factors are calculated as a weighted mean of M-values not using those most expressed genes (largest A-values) or those genes with the largest log-ratios (largest M-values). Robinson and Oshlack (2010) suggested using those genes with their M-values in the middle 40% and their A-values in the middle 90%. The same setting for the TMM method is used in this dissertation. The inverse of the variance for each M-value is used as weight in the calculation. As a consequence of the Delta method, the M-value has a sampling distribution that is approximately normal. The variance of the gene-specific M-value MV_g is approximated by the following:

$$Var(MV_g) \approx \frac{L_{ij} - y_{ijg}}{L_{ij} y_{ijg}} + \frac{L_r - y_{rg}}{L_r y_{rg}} \quad (1)$$

where y_{rg} denotes the read count of the g^{th} gene for the reference sample r and L_r denotes the library size of the reference sample r.

The TMM method produces a normalization factor for each sample. The normalization factors multiplied by the corresponding library sizes are referred to as effective library sizes. The calculation of normalization factors is implemented in the R function calcNormFactors which is contained in the edgeR package.

2.3 Statistical Methods Review and Motivation

The samples in an RNA-seq experiment could be either technical replicates or biological replicates. For inference about a population, biological replicates are essential. It is assumed in this dissertation that all replicates are biological replicates. Biological replicates usually exhibit greater variability than technical replicates. The restrictive mean-variance relationship for the Poisson distribution is reasonable for technical replicates but does not adequately accommodate the greater variability in biological replicates. The presence of greater variability in statistics is called overdispersion, sometimes referred to as unobserved heterogeneity. More strictly, overdispersion is the presence of greater variability than would be expected under the assumed distribution or a given statistical model (McCullagh and Nelder, 1989). Most genes in the RNA-seq data are overdispersed when an underlying Poisson distribution is assumed. To have a valid statistical test, the gene-specific overdispersion should be properly taken into account. The advantage of assuming the NB distribution is that its quadratic mean-variance relation is able to model this overdispersion by a gene-specific NB dispersion parameter. Two popular methods based on the NB assumption are implemented in R packages edgeR and DESeq.

In the edgeR method, the raw RNA-seq count data are assumed to have the following NB distribution with mean and variance:

$$y_{ijg} \sim NB(L_{ij}P_{ig}, \quad L_{ij}P_{ig} + \phi_g L_{ij}^2 P_{ig}^2) \tag{2}$$

where L_{ij} is the library size (If considering normalization, the effective library sizes are used) and P_{ig} is the abundance relative to the library size of the g^{th} gene in the i^{th} experimental condition, ϕ_g is the NB dispersion parameter which can be estimated from the data by assuming: 1) all genes have a constant NB dispersion parameter; or 2) there is a trend between NB dispersion parameter and average count; or 3) the NB dispersion parameter is gene-specific. The edgeR method uses the quantile-adjusted conditional maximum likelihood (qCML) method to estimate the NB dispersion parameters for experiments with a single factor. The raw counts are first adjusted using NB quantile-to-quantile method (Robinson and Smyth, 2008) so that their library sizes are all equal to the geometric mean of the original library sizes. The adjusted counts are called pseudo-counts, which "represent the equivalent counts would have been observed had the library sizes all been equal" (Chen et al., 2016). To estimate the common dispersion, the log-likelihood conditioning on the total pseudo-counts for each gene $l_g(\phi)$ is used, where ϕ is the common NB dispersion parameter. The common dispersion is estimated through maximizing the common log-likelihood $l_c(\phi) = \sum_{g=1}^{G} l_g(\phi)$. To estimate the trended dispersion, the genes are divided into bins by average pseudo-counts. The common dispersion in each bin is estimated through maximizing the common log-likelihood in each bin. A smooth curve is fitted between the binned common dispersions and binned average pseudo-counts. Based on the estimated curve, the trended dispersion for each gene is the predicted value of each gene's overall average pseudo-count. To estimate gene-specific dispersion, the strategy in the empirical Bayes method (Robinson and Smyth, 2007) is used. It is implemented

as a weighted conditional log-likelihood $WL(\phi_g) = l_g(\phi_g) + w_g l_c(\phi_g)$, where w_g is the gene-specific weight and the common log-likelihood is for all genes or subset of genes in the same bin. The gene-specific dispersion is obtained by maximizing the weighted conditional log-likelihood. If a dispersion trend is assumed, the common log-likelihood for the subset of genes in the same bin is used. Otherwise, the common log-likelihood for all genes is used. A dispersion trend is assumed in this dissertation. Compared with other estimators such as the maximum likelihood estimator, qCML is the most reliable (Chen et al., 2016). However, the qCML method is only applicable to experiments with one factor. When an experiment has more than one factor, it fails to take that into account. “It has been seen in many RNA-Seq datasets that allowing gene-specific dispersion is necessary in order that differential expression is not driven by outliers” (Chen et al., 2016). Therefore the gene-specific dispersion is strongly recommended. In this dissertation, the gene-specific dispersion is used in the edgeR method.

Once the parameters in Equation (2) are all estimated from the data, the statistical test is analogous to Fisher’s exact test (Robinson and Smyth, 2008). In a two experimental conditions setting, let z_{1g}^* and z_{2g}^* be the sum of pseudo-counts in experimental condition 1 and 2. Under the null hypothesis,

$$z_{ig}^* \sim NB\left(N_i L^* P_{ig}, \quad \phi_g / N_i\right) \quad i \in 1, 2 \tag{3}$$

where L^* is the geometric mean of the original library sizes. Conditional on the sum $z_{1g}^* + z_{2g}^*$, the probability of group totals more extreme than the observed group totals can be calculated under the null hypothesis. The 2-sided p-value is defined as the sum of the probabilities of group totals lower than or equal to the probability of the observed group totals. This p-value is used to test differential expression.

In the DESeq method, the raw RNA-seq count data are assumed to have the following NB distribution with mean and variance:

$$y_{ijg} \sim NB(\eta_{ij}\delta_{ig},\ \ \eta_{ij}\delta_{ig} + \alpha\,\eta_{ij}^2\delta_{ig}^2) \tag{4}$$

where η_{ij} denotes size factor of the j^{th} sample in the i^{th} experimental condition. To calculate the size factor, a reference sample is first constructed using the geometric mean of all samples. The size factor is calculated as the median of the gene-specific ratios between the j^{th} sample in the i^{th} experimental condition and the reference sample. In this calculation, the genes with any zero count are not used since they produce zeros in the reference sample, and the gene-specific ratios become non-evaluable. The raw count data adjusted by the size factor is referred to as the RLE normalization. To use the TMM normalization in DESeq, the size factor is replaced with the normalization factor calculated by the TMM normalization. δ_{ig} is a value proportional to the true concentration of the g^{th} gene in the i^{th} experimental condition. The variance is the sum of a shot noise term $\eta_{ij}\delta_{ig}$ and a raw variance term $\alpha\eta_{ij}^2\delta_{ig}^2$, where α is the dispersion parameter. The dispersion is modeled in four different ways: The first way is to assume that the dispersion is different for each gene in each experimental condition. Based on this assumption, the dispersion parameter is denoted as α_{ig} and estimated by the following equation:

$$\hat{\alpha}_{ig} = \frac{S_{ig}^2 - \frac{\bar{z}_{ig}}{N_i}\left(\sum_{j=1}^{N_i}\frac{1}{\eta_{ij}}\right)}{\bar{z}_{ig}^2} \tag{5}$$

where $\bar{z}_{ig}$ is the average of the normalized counts in the i^{th} experimental condition. The second way is to assume a common dispersion for each gene. In estimation, the experimental condition is ignored and the gene-specific dispersion is estimated as if all

samples are in the same experimental condition. The estimation can be applied to the situation even when there are no replicates. The dispersion is estimated similarly as Equation (5) but ignoring the experimental condition. The third way is also to assume a common dispersion for each gene. The common dispersion for each gene is estimated by:

$$\hat{\alpha}_g = \frac{\tilde{S}_g^2 - \frac{\bar{z}_g}{N}\left(\sum_{i=1}^{2}\sum_{j=1}^{N_i}\frac{1}{\eta_{ij}}\right)}{\bar{z}_g^2} \tag{6}$$

where $\tilde{S}_g^2 = \sum_{i=1}^{2}\sum_{j=1}^{N_i}(z_{ijg} - \bar{z}_{ig})^2/(N-2)$. The fourth way also assumes a common dispersion for each gene but is estimated through maximizing a Cox-Reid adjusted profile likelihood (Cox and Reid, 1987; McCarthy et al., 2012) and is usually used in the case of no replicates in some subgroups in a complex design. In this dissertation, the third method is used to obtain the raw dispersion estimates. The gene-specific abundance δ_g is estimated by $\bar{z}_g$, which is the average of the normalized counts of the g^{th} gene.

With the pairs of raw estimates $\hat{\alpha}_g$ and $\hat{\delta}_g$ by the third method, there are two approaches proposed to fit a smooth curve: 1) a parametric fit using a gamma-family Generalized Linear Model (GLM) to fit $\alpha_g = \alpha_{0g} + \alpha_{1g}/\delta_g$, where α_{0g} and α_{1g} are two coefficient parameters; and 2) local regression. In the second approach, a GLM of gamma family for the local regression is used to fit $\tilde{S}_g^2$ and $\hat{\delta}_g$. The resulting function is denoted as $\widehat{w}$. The smooth function for dispersion is estimated by:

$$\hat{\alpha}(\hat{\delta}_g) = \frac{\widehat{w}(\hat{\delta}_g) - \frac{\hat{\delta}_g}{N}\left(\sum_{i=1}^{2}\sum_{j=1}^{N_i}\frac{1}{\eta_{ij}}\right)}{\hat{\delta}_g^2} \tag{7}$$

For some datasets, the parametric fit in the first approach may give bad results (Anders and Huber, 2016). In this dissertation, the second approach is used for the DESeq method. In the statistical inference, the fitted values from the curve or the raw dispersion estimates are used. There are three choices: 1) only fitted values from the curve are used; 2) only raw dispersion values are used; and 3) to be conservative, the maximum of the fitted values and the raw gene-specific values are used for inference. With the assumption that the variation of the point estimates around the curve simply reflects sampling variability, the fitted values are used for the DESeq method in this dissertation for inference.

In the DE analysis, the statistical test is analogous to Fisher's exact test (Anders and Huber, 2010). Suppose there are two conditions 1 and 2, and the total observed counts of the g^{th} gene in each condition are $\tilde{y}_{1g}$ and $\tilde{y}_{2g}$. Let y_{1g} and y_{2g} be the non-negative integers having the value from 0 to $\tilde{y}_{1g} + \tilde{y}_{2g}$, which also satisfies $y_{1g} + y_{2g} = \tilde{y}_{1g} + \tilde{y}_{2g}$. Under the null hypothesis,

$$y_{ig} \sim NB\left(\hat{\delta}_g \eta_i, \quad \hat{\alpha}^*_{ig} \delta_g^2 \eta_i^*\right) \qquad i \in 1, 2 \tag{8}$$

where $\eta_i = \sum_{j=1}^{N_i} \eta_{ij}$, $\eta_i^* = \sum_{j=1}^{N_i} \eta_{ij}^2$, $\hat{\delta}_g = \bar{z}_g$, and $\hat{\alpha}^*_{ig} = (\hat{\delta}_g \sum_{i=1}^{2} \eta_i + \hat{\alpha}(\hat{\delta}_g)\hat{\delta}_g^2 \sum_{i=1}^{2} \eta_i^* - \hat{\delta}_g \eta_i)/(\hat{\delta}_g^2 \eta_i^2)$. In Equation (8), $\hat{\alpha}(\hat{\delta}_g)$ is calculated by Equation (7), which is based on the normalized data.

The probability of such a combination can be calculated using the above NB distribution. The probability is denoted as $\pi(y_{1g}, y_{2g})$. The p-value is calculated by the following equation:

$$\pi_g = \frac{\sum_{\pi(y_{1g},y_{2g}) \le \pi(\tilde{y}_{1g},\tilde{y}_{2g})} \pi(y_{1g}, y_{2g})}{\sum \pi(y_{1g}, y_{2g})} \quad (9)$$

In the latest implementation (Anders and Huber, 2016), the p-value is calculated differently since Equation (9) has problems when the dispersion is larger than 1. To address that, the probabilities of value pairs having a more extreme fold change are summed in the numerator of Equation (9).

Beyond the NB distribution, there are also other possible distributions which are able to model the gene-specific overdispersion such as the negative binomial power distribution (Di et al., 2011), the Poisson-inverse Gaussian, the Sichel, the Delaporte and related distributions (Rigby et al., 2008). Esnaola et al. (2013) has argued that RNA-seq data can be modeled using a more general family of distributions, the Poisson-Tweedie family. These alternative distributions only have the flexibility to model overdispersed data but not underdispersed data. In real data, some genes are underdispersed. An approach without any distributional assumption and able to handle both overdispersed and underdispersed data is preferable. In Section 2.4, a statistical method is derived based on the theory of quasi-likelihood which is able to handle both overdispersed and underdispersed data without any distributional assumptions.

The quasi-likelihood is defined in McCullagh and Nelder (1989) as

$$Q(\mu, y) = \int_y^\mu \frac{y - t}{\Phi V(t)} dt \quad (10)$$

where μ is the mean, y is the observed value, Φ is the dispersion parameter, and V is the variance function. The variance function describes how the variance of a random variable depends on its mean. In the quasi-likelihood framework, the variance of the random variable is modeled as $\Phi V(\mu)$. To model a gene-specific variance, either Φ or

V or both could be modelled as gene-specific. Analogous with the deviance in GLM (McCullagh and Nelder, 1989), the quasi-deviance is

$$D(\mu, y) = -2\Phi Q(\mu, y) = 2\int_{\mu}^{y} \frac{y-t}{V(t)} dt \tag{11}$$

In an RNA-seq experiment with a simple completely randomized design with two experimental conditions, the quasi-likelihood ratio statistic for the g^{th}gene is

$$QLRT_g = \sum_{i=1}^{2}\sum_{j=1}^{N_i} D(\hat{\mu}_g, y_{ijg}) - \sum_{i=1}^{2}\sum_{j=1}^{N_i} D(\hat{\mu}_{ig}, y_{ijg}) \tag{12}$$

where $\hat{\mu}_g$and $\hat{\mu}_{ig}$are the mean estimates from the quasi-likelihood method for the model with one overall mean (null model) and the model with two group means in two experimental conditions (full model). When the variance function is correct, McCullagh (1983) showed that under the null hypothesis

$$QLRT_g \sim \Phi_g \chi_q^2 \tag{13}$$

where $q = 1$, which is the difference in degrees of freedom between the full model and the null model in a two-group setting.

The dispersion parameter Φ_g can be estimated from the data using deviance or moment approach. The deviance-based estimator is

$$\hat{\Phi}_g^{deviance} = \frac{\sum_{i=1}^{2}\sum_{j=1}^{N_i} D(\hat{\mu}_{ig}, y_{ijg})}{N-p} \tag{14}$$

where $p = 2$, which is the number of group means in a two-group setting.

If Φ_g is known, the quasi-likelihood ratio statistic $QLRT_g/q$ can be used to test the difference between two models. If Φ_g is unknown, the quasi-likelihood ratio statistic is not valid. In this case, the F statistic $QLRT_g/(q\hat{\Phi}_g)$ should be used (Tjur, 1998). The F statistic is also valid when Φ_g is known. The F statistic is compared with an F-distribution with q and $N-p$ degrees of freedom.

Lund et al. (2012) proposed quasi-likelihood methods to analyze RNA-seq data by assuming a linear or quadratic variance function (corresponding to Poisson or NB variance functions). The gene-specific NB dispersion parameters of the quadratic variance functions are estimated through the edgeR method. The dispersion Φ_g is estimated by Equation (14) which is often based on few degrees of freedom. To share information across genes, the empirical Bayes method (Smyth, 2004) to estimate the gene-specific error variance in microarray data can be adapted to estimate the dispersion Φ_g in RNA-seq data. There are two approaches to apply this method:

1) Using the Empirical Bayesian method, the gene-specific dispersion Φ_g is assumed to have a scaled-inverse χ^2 prior distribution with d_0 degrees of freedom and a scale factor Φ_0. It is also assumed that $(N-p)\widehat{\Phi}_g/\Phi_g$ has a χ^2 distribution with $N-p$ degrees of freedom. The hyperparameters d_0 and Φ_0 can be estimated using the moments approach described by Smyth (2004). Denoting these estimates as $\hat{d}_0$ and $\widehat{\Phi}_0$, the first dispersion shrinkage estimator (QLShrink) is as follows:

$$\widehat{\Phi}_g^{shrink} = \frac{\hat{d}_0\widehat{\Phi}_0 + (N-p)\widehat{\Phi}_g}{\hat{d}_0 + N - p} \tag{15}$$

The test statistic $QLRT_g/(q\widehat{\Phi}_g^{shrink})$ is compared with an F-distribution with q and $\hat{d}_0 + N - p$ degrees of freedom.

2) The second shrunken dispersion estimator is called QLSpline. A cubic spline is used to fit $\log(\widehat{\Phi}_g)$ and $\log(\bar{y}_g)$ and denote the predicted value as $\widehat{\Phi}_{0g}$. $\widehat{\Phi}_g/\widehat{\Phi}_{0g}$ is assumed to follow a scaled F-distribution with scale factor τ and degrees of freedom $N-p$ and d_0'. Using the method of moments approach, the hyperparameters d_0' and τ can be

estimated and denote the associated estimates as $\hat{d}_0'$ and $\hat{\tau}$. The second dispersion estimator QLSpline is as follows:

$$\hat{\Phi}_g^{spline} = \frac{\hat{d}_0'\hat{\Phi}_{0g}\hat{\tau} + (N-p)\hat{\Phi}_g}{\hat{d}_0' + N - p} \quad (16)$$

The test statistic $QLRT_g/(q\hat{\Phi}_g^{spline})$ is compared with an F-distribution with q and $\hat{d}_0' + N - p$ degrees of freedom.

Lund et al. (2012) recommended using QLSpline shrinkage estimator under the assumption of a quadratic variance function, which is denoted as NBQLSpline. This approach is implemented in the R package QuasiSeq and is used in this dissertation to compare with the QuasiDE method.

The Poisson or NB assumption is useful, “but can be heavily influenced by outliers” (Li and Tibshirani, 2013). Also the underlying distribution for a real RNA-seq dataset may be different from the Poisson or NB distribution. The goal in this chapter is to develop a statistical method without any distributional assumption. Consequently it has similar sensitivities and FDRs across different distributions with different variance functions. The performance of this method should be at least as good as the existing methods.

2.4 New Quasi-Likelihood Method (QuasiDE)

A novel quasi-likelihood approach (QuasiDE) is proposed to detect differentially expressed genes for RNA-seq data in a completely randomized design with two experimental conditions. In the proposed method, there is no underlying distribution assumed. Instead, a single variance function with gene-specific dispersion is assumed. The variance function is estimated from the normalized count data empirically. In normalization, raw read counts are first divided by their associated effective library sizes

and then multiplied by the mean of the effective library sizes. The test statistic is constructed by quasi-likelihood theory.

The variance of the normalized count for the g^{th} gene has the following form:

$$var(z_{ijg}) = \Phi_g V(\mu_{ig}) \quad (17)$$

where μ_{ig} is the expected normalized count of the g^{th} gene in the i^{th} experimental condition and Φ_g is the dispersion parameter in quasi-likelihood theory. It indicates that each gene has a gene-specific dispersion Φ_g, but all genes share a common variance function V. If there are no DE genes, the within-gene variance function can be estimated using the sample mean ($\bar{z}_g$) and sample variance (S_g^2) pairs of the normalized counts. The mean-variance relationship for those DE genes is quite different from those non-DE genes due to different experimental conditions which cause greater observed variance. To address that, the sample mean-variance pairs within experimental conditions ($\bar{z}_{ig}$ and S_{ig}^2) are used to estimate the variance function V, although the resulting sample mean-variance relationship exhibits larger random variability due to a smaller sample size. For a simple completely randomized design with two experimental conditions, that means $2G$ pairs of sample mean-variance are used to estimate the variance function. After applying the log transformation, Equation (17) changes to the following form:

$$\log[var(z_{ijg})] = \log(\Phi_g) + \log\left[V(\mu_{ig})\right] \quad (18)$$

To estimate the variance function V, the model in Equation (18) fails to be identifiable since multiple parameterizations are possible. To address that, a constraint $\sum_{g=1}^{G} \log(\Phi_g) = 0$ is imposed. A smooth cubic spline is fitted to model $\log(S_{ig}^2)$ as a function of $\bar{z}_{ig}$ and treating $\log(\Phi_g)$ as the error terms with mean 0 and common variance ξ^2. The resulting spline describes the overall biological variability across all

genes. The gene-specific dispersion Φ_g models the gene-specific variability relative to the overall level. The fitted spline function is denoted as $\hat{h}$. The exponential of this spline function is the estimated variance function $\hat{V}$.

$$\hat{V} = \exp(\hat{h}(t)) \quad t > 0 \tag{19}$$

With this estimated variance function $\hat{V}$, a GLM model can be fitted using the quasi-likelihood method. In this GLM model, the variance structure is specified by the estimated variance function and the mean structure is specified based on the experimental design. The GLM model with the identity link function can yield negative values for the predicted normalized counts, which is not a desirable feature. The GLM model with log link can only yield a positive predicted normalized count and has been employed. In a completely randomized design with two conditions, the full model is $\log(\boldsymbol{\mu_g}) = \beta_{1g} + \beta_{2g}X$, where X is an indicator of the experimental condition, $\boldsymbol{\mu_g} = E(\boldsymbol{z_g})$ is the expected normalized count vector for the g^{th} gene, and β_{1g}, β_{2g} are the regression coefficients for the g^{th} gene. In the quasi-likelihood framework, the coefficients are estimated by the quasi-likelihood estimating equations $U_g(\beta_{1g}, \beta_{2g}) = 0$, where

$$U_g(\beta_{1g}, \beta_{2g}) = \boldsymbol{\mathcal{D}_g^T V_g^{-1}}(\boldsymbol{z_g} - \boldsymbol{\mu_g})/\Phi_g \tag{20}$$

$\boldsymbol{z_g}$ is the normalized count vector for the g^{th} gene and $\boldsymbol{\mathcal{D}_g}$ is $N \times 2$ matrix:

$$\boldsymbol{\mathcal{D}_g} = \begin{pmatrix} \frac{\partial \mu_{1g}}{\partial \beta_{1g}} & \frac{\partial \mu_{1g}}{\partial \beta_{2g}} \\ \vdots & \vdots \\ \frac{\partial \mu_{Ng}}{\partial \beta_{1g}} & \frac{\partial \mu_{Ng}}{\partial \beta_{2g}} \end{pmatrix} \tag{21}$$

and $\boldsymbol{V_g}$ is the $N \times N$ diagonal matrix:

$$\boldsymbol{V_g} = \begin{pmatrix} \hat{V}(\mu_{1g}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \hat{V}(\mu_{Ng}) \end{pmatrix} \tag{22}$$

The equations $U_g(\beta_{1g}, \beta_{2g}) = 0$ are solved to obtain the coefficient estimates $\hat{\beta}_{1g}, \hat{\beta}_{2g}$.

The null regression model is $\log(\boldsymbol{\mu_g}) = \beta_{1g}$. Using the same estimated variance function, the β_{1g} estimate $\hat{\beta}_{1g}$ can be similary obtained through the quasi-likelihood estimating equations.

The fitted values for the null model are denoted as $\hat{z}_{ijg,null}$ and the fitted values for the full model are denoted as $\hat{z}_{ijg,full}$. This is implemented by using the glm function in R. The detailed R code is in Appendix A.

The quasi-deviance of the null model is

$$\widetilde{D}_{g,null} = \sum_{i=1}^{2} \sum_{j=1}^{N_i} 2 \int_{\hat{z}_{ijg,null}}^{z_{ijg}} \frac{z_{ijg} - t}{\hat{V}(t)} dt \tag{23}$$

The quasi-deviance of the full model is

$$\widetilde{D}_{g,full} = \sum_{i=1}^{2} \sum_{j=1}^{N_i} 2 \int_{\hat{z}_{ijg,full}}^{z_{ijg}} \frac{z_{ijg} - t}{\hat{V}(t)} dt \tag{24}$$

The difference of Equation (23) and (24) is only on the fitted values used.

The most popular dispersion estimator is the Pearson moment estimator (McCullagh, 1983):

$$\widehat{\Phi}_g^{pearson} = \frac{1}{N-p} \sum_{i=1}^{2} \sum_{j=1}^{N_i} \frac{\left(z_{ijg} - \hat{z}_{ijg,full}\right)^2}{\hat{V}\left(\hat{z}_{ijg,full}\right)} \tag{25}$$

The test statistic $(\widetilde{D}_{g,null} - \widetilde{D}_{g,full}) / (q\widehat{\Phi}_g^{pearson})$ is proposed to compare with an F-distribution with q and $N - p$ degrees of freedom. This method is referred to as "QuasiDE" in this dissertation.

2.5 Multiple-Comparison Procedures

Although genes are not expressed independently, the DE analysis is typically applied one gene at a time. The DE analysis then involves thousands, or even millions of statistical tests of the null hypotheses. With multiple statistical tests, each test has type I and type II error and it is important to control the overall error rate. The first introduced measure of the overall error rate is the family-wise error rate (FWER), which is the probability of making one or more false discoveries among all these hypotheses. Instead of controlling the type I error for each test, the FWER is the target to be controlled. In many situations, controlling the FWER is too strict, especially in the genomic analysis which involves a large number of statistical tests. To address that, another measure of the overall error rate, false discovery rate (FDR), was introduced by Benjamini and Hochberg (1995). FDR is the expected proportion of false positive findings among all the rejected hypotheses. Compared with controlling the FWER, controlling the FDR is found to be more powerful but it has a higher overall type I error. Benjamini and Hochberg (1995) provided a sequential p-value method to control the FDR, which has been adopted in the edgeR and DESeq methods. To adjust the multiple comparisons in the NBQLSpline method, the q-value, which measures the proportion of false positives incurred (Storey and Tibshirani, 2003), is calculated for each test using the method by Nettleton et al. (2006).

In this dissertation, the Benjamini and Hochberg procedure is adopted to adjust the raw p-values produced by the QuasiDE method. Since comparing the methods designed for correcting multiple comparisons is not intended in this dissertation, the multiple comparison procedure used in the NBQLSpline method has been changed to the

Benjamini and Hochberg procedure. This procedure is implemented in function p.adjust in R.

2.6 Simulation

To examine the performance of the QuasiDE method, a series of simulations have been conducted. The data are simulated based on three monotonic increasing variance functions (Table 2). In Table 2, the three simulated variance functions and their associated dispersion parameters are all listed. The dispersion parameters are specified by a lognormal random variable with mean and standard deviation on log scale. Only monotonic increasing variance functions are considered since it was suggested that variance grows with the mean in the RNA-seq data (Love et al., 2014). The first function is in a linear form, the second is a quadratic function and the last function is in a form of $\mu + g(\mu)$. $g(\mu)$ is chosen to be $\ln(\mu + 1)$, which has a different increase speed than the other two functions. The assumption of all these three variance functions satisfies $V(\mu = 0) = 0$. These three types of data are referred to as “QP data”, “QN data”, and “LN data” respectively hereafter.

The gene-specific dispersion parameter is simulated as a log-normal random variable. The parameters for this log-normal random variable are chosen to make the simulated data roughly like the Himes data. Sample sizes are chosen to be 10, 20, and 40, split evenly between two experimental conditions. In a real RNA-seq dataset, the transcripts have at least one copy. Since a gene not expressed is not possible to appear in a real RNA-seq dataset, the simulated genes with zero total count are replaced with a new simulated gene. In each dataset, 20,000 genes are simulated and 5% of them are

Table 2: Variance Functions and Dispersion Parameters used in Simulation

Variance Function	Dispersion Parameter
$V(\mu) = 3\mu$	Log-normal (0, 1.7)
$V(\mu) = 0.5(\mu + \frac{\mu^2}{16})$	Log-normal (0, 1.4)
$V(\mu) = \mu + \ln(\mu + 1)$	Log-normal (0, 1.2)

Note: Variance V is a function of mean μ. The two parameters in the log-normal random variable are the mean and standard deviation on the logarithm scale.

chosen to be DE (1000 DE and 19,000 EE genes). The fold changes of those DE genes were set to be a constant in the dataset. This constant is chosen to be either 1.5 or 3 to represent low and high treatment effect. Half of the DE genes have higher mean expressions in condition 1 and the other half have higher mean expression in condition 2.

In each setting (per sample size and fold change), 200 datasets were simulated. The read counts were simulated as follows. First, the average count for each gene was randomly sampled with replacement from the observed within-group means from the Himes data. Three types of genes are simulated: EE genes, over-expressed DE genes and under-expressed DE genes. For an EE gene, the average count is assigned to all samples. For an over-expressed DE gene, the average count is firstly assigned to all samples; those in the second experimental condition will be multiplied by a constant (1.5 if low treatment effect, 3 if high treatment effect). For an under-expressed gene, the average count is firstly assigned to all samples; those in the second experimental condition will be divided by a constant (1.5 if low treatment effect, 3 if high treatment

effect). These assigned means are then scaled by a random factor which mimics library size effect. This random factor is two to the power of a normal random variable with mean 0 and standard deviation 0.5. Using this scaled average counts and the parameters in Table 2, the corresponding variances are calculated. For each pair of the scaled average count and its variance, the shape parameter and rate parameter for a gamma random variable can be estimated by a method of moments approach. A gamma random variable is then generated using these parameter estimates. This generated continuous random variable is then rounded to an integer and used as raw count data.

The resulting raw count data are first filtered by removing those genes with zero interquartile range or low mean count (≤ 1). The TMM normalization method is applied to the remaining genes. The data are analyzed by different statistical methods (edgeR, DESeq, NBQLSpline and QuasiDE). A nominal FDR we are willing to allow is chosen at level of 0.05 throughout the simulation in this dissertation. All simulation and analyses were conducted using R 3.2.1 and the relevant R code is included in Appendix A.

2.7 Simulation Results

2.7.1 Variance Function Estimation

The smooth cubic spline is used in the QuasiDE method to estimate the variance function. “Spline functions are the most successful approximating functions for practical application” (Rice, 1969). In estimation, the within-group mean-variance pairs of the normalized counts in the simulated dataset from those genes after filtering are used. The estimated variance functions are shown in Figure 1 for the QP, QN and LN simulated datasets with sample size 10, 20 and 40. In Figure 1, the estimated variance

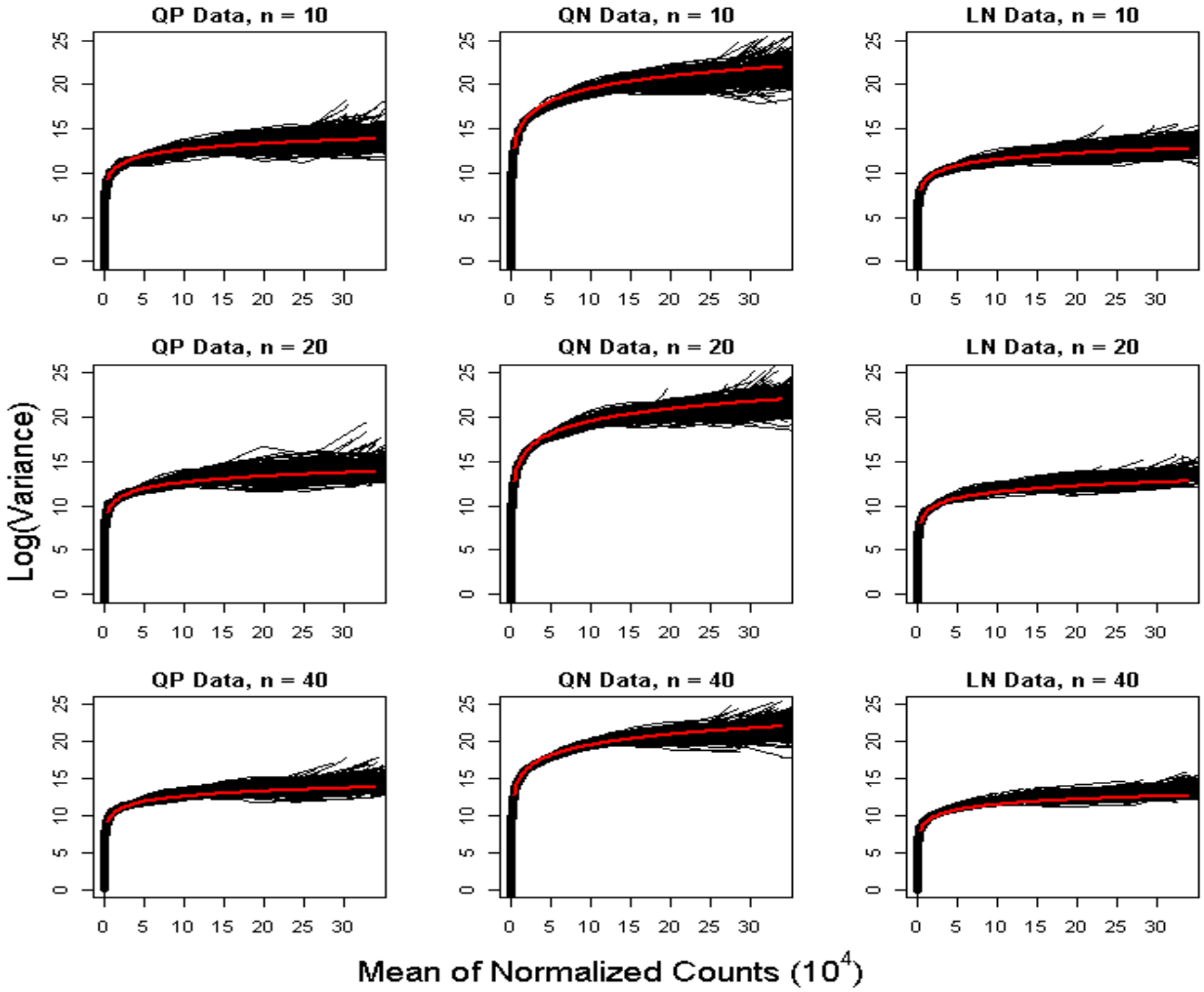

Figure 1. Variance Function Estimation.

The panels in the first, second and third columns show the estimated variance functions for the QP, QN and LN data respectively. The panels in the first, second and third rows show the estimated variance functions for sample size 10, 20 and 40 respectively. In each panel, there are 400 estimated variance functions based on 400 simulated datasets. In each panel, the red line is the true variance function.

functions vary around the true variance function. The true variance functions of the QP and LN data are similarly, which are quite different from the true variance function of the QN data. The variability of the predicted functions at those larger normalized count is expected to be larger and this is also reflected in this figure.

The method of estimating the variance functions using the smooth cubic spline is compared with using the quadratic regression, in which the within-group variances on log scale is modeled as a quadratic function of the average normalized counts. The model fit is measured by mean integrated squared error (MISE) (Härdle, 1986), which is defined as

$$MISE = E\left\{\int \left[h(\mu) - \hat{h}(\mu)\right]^2 f(\mu) d\mu\right\} \quad (26)$$

where h and $\hat{h}$ are the true and estimated functions on log scale. $f(\mu)$ is the density function. The MISE is approximated by using the empirical distribution of mean normalized counts in the simulated dataset. For a specific sample size and variance function, there are 400 simulated datasets (200 with the simulated DE genes having low fold change and 200 with the simulated DE genes having high fold change).

Table 3: MISEs of Estimated Variance Functions in Simulation

Variance Function	Sample Size		
	n = 10	n = 20	n = 40
Smooth Cubic Spline			
QP	0.20	0.09	0.05
QN	0.09	0.03	0.03
LN	0.07	0.02	0.02
Quadratic Regression			
QP	5.56	5.49	5.42
QN	15.47	15.55	15.48
LN	4.87	4.86	4.82

Note: The variance functions are estimated by a smooth cubic spline or a quadratic regression. The detailed form of variance functions are listed in Table 2.

The approximate MISE is calculated by the following equation:

$$MISE \approx \frac{\sum_{d=1}^{400}\sum_{i=1}^{2}\sum_{g=1}^{G_d}\left[h\left(\bar{z}_{ig}^{d}\right)-\hat{h}\left(\bar{z}_{ig}^{d}\right)\right]^2}{\sum_{d=1}^{400}\sum_{i=1}^{2}G_d} \quad (27)$$

where d denotes the d^{th} dataset, $\bar{z}_{ig}^{d}$ is the mean normalized count of the g^{th} gene for the i^{th} experimental condition in the d^{th} dataset and G_d is the total number of genes after filtering in the d^{th} dataset. The estimated MISEs in simulation are summarized in Table 3. In Table 3, the MISEs of the estimated smooth cubic spline are much lower than the estimated quadratic function obtained from quadratic regression. This indicates the estimated cubic splines have a better model fit. Within the method of smooth cubic spline, the model fit gets better in a larger sample size. Also, the model fit of the LN data is the best among three types of data and the model fit of the QP data is the worst. This is due to the designed dispersion parameters, in which the QP data has the largest variability and the LN data has the smallest variability (Table 2). Since the three variance functions on log scale are not in quadratic form, using quadratic regression does not have good performance. The variance function on log scale for the QP and LN data appears to be more close to a quadratic form than that for the QN data. This systematic error is not reduced with larger sample size.

2.7.2 Performance of Different Methods

An ideal statistical method has sensitivity as high as possible while the real FDR is well controlled at the nominal level. In simulation, the sensitivities and real FDRs by different statistical methods are compared in different simulation settings. With larger sample sizes and higher treatment effect, the power to detect the DE genes should increase.

The simulation result for the QuasiDE method is shown in Figure 2. The sensitivity increases with sample size and treatment effect. This suggests that more biological replicates result in higher statistical power. Compared with low treatment effect, QuasiDE has more statistical power to detect those DE genes with high treatment effect. The real FDR of QuasiDE increases slightly from sample size 10 to 20. From sample size 20 to 40, the real FDR becomes stable. For the QP and QN data, the real FDR is best controlled at sample size 10. For the LN data, the real FDR is best controlled at sample size 20 and 40. Moreover, the patterns of sensitivity and FDR are similar across three types of data. All real FDRs are close to the specified nominal level of 0.05. In the setting of sample size 10 and low treatment effect, the QuasiDE method has a bit lower sensitivity and higher variability of real FDR in the QN data compared to the other two types of data. The raw p-values and adjusted p-values produced by the QuasiDE method for those EE and DE genes are shown in Figure 3. The data used in this figure is from the first 10-sample QN simulated dataset in the high fold change setting. The raw p-values for the EE genes roughly have a uniform distribution. Most the raw p-values for those DE genes are very small and the density decreases sharply by the raw p-value. After multiple comparison correction, ideally, most of the adjusted p-values for those EE genes are not small enough to be significant. And most of the adjusted p-values for those DE genes are still small enough to be significant. This is also observed in Figure 3. The distributions shown in Figure 3 are similar in other datasets and simulation settings.

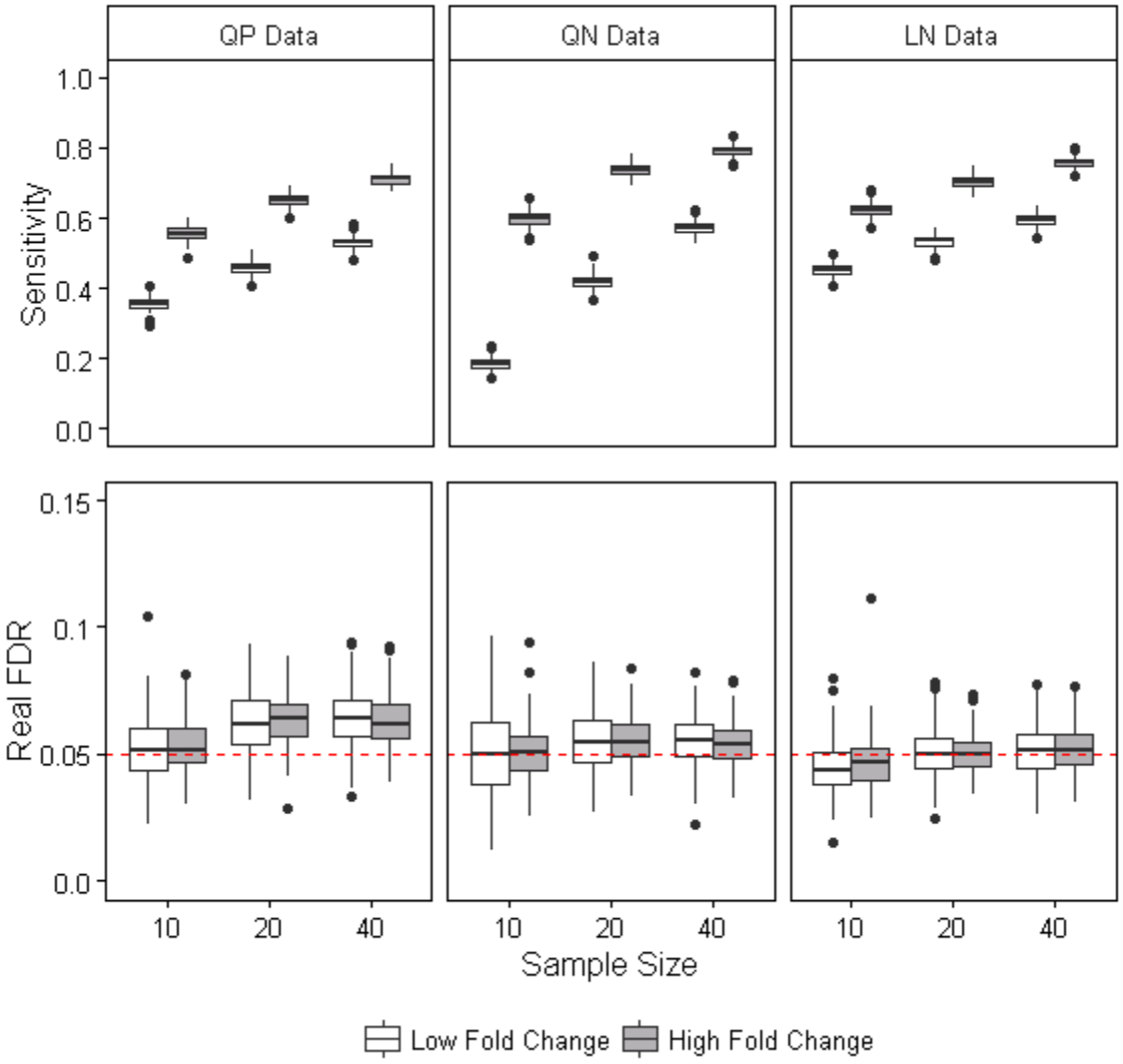

Figure 2. Sensitivity and Real FDR by Sample Size and Treatment Effect for Different Types of Data (QuasiDE)

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

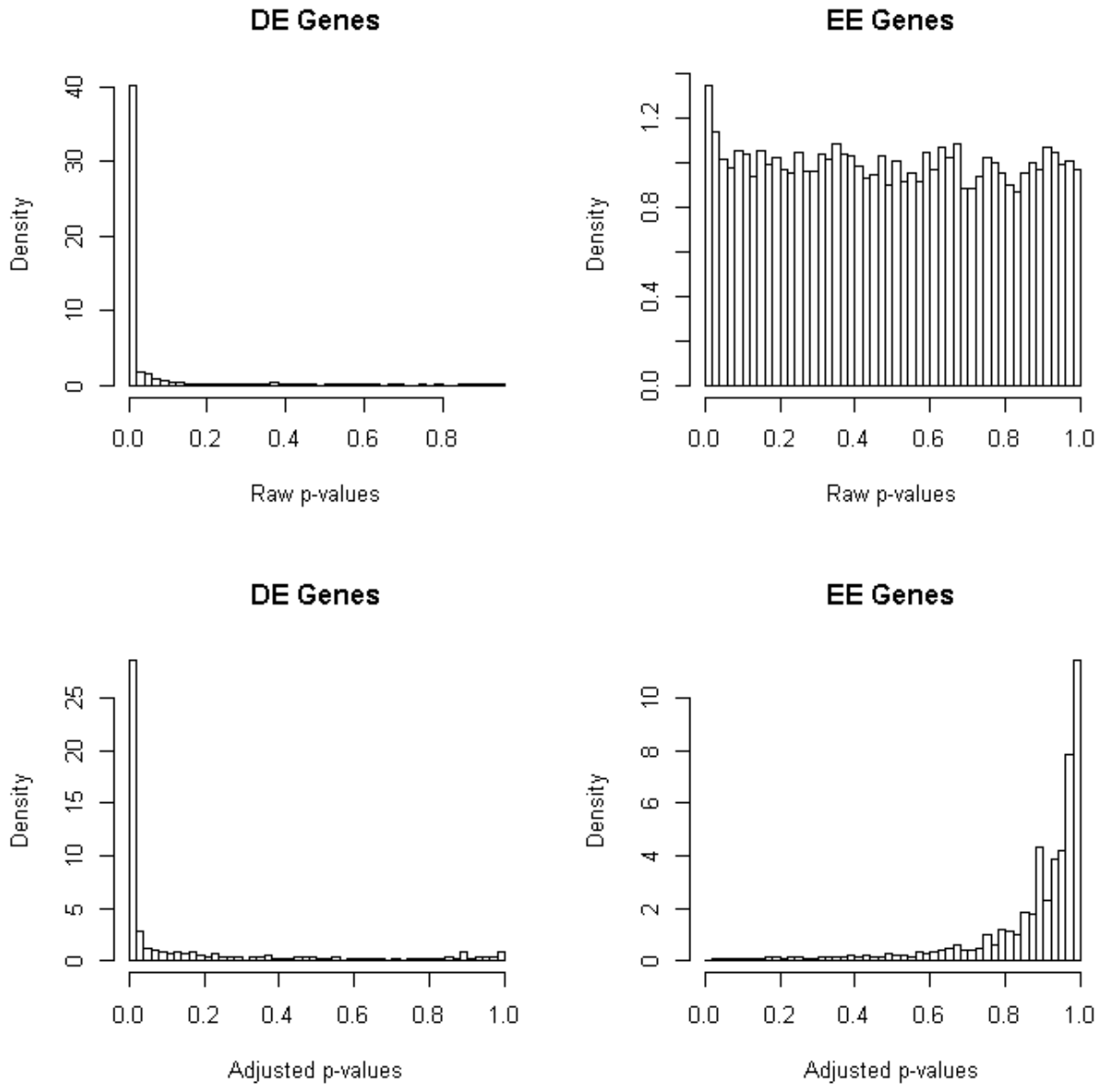

Figure 3. Histograms of Raw p-values and Adjusted p-values for the DE and EE genes in the simulation (QN dataset 1, High Fold Change, n = 10, QuasiDE)

The two panels in the first row show the density of raw p-values for the DE and EE genes. The two panels in the second row show the density of adjusted p-values for the DE and EE genes.

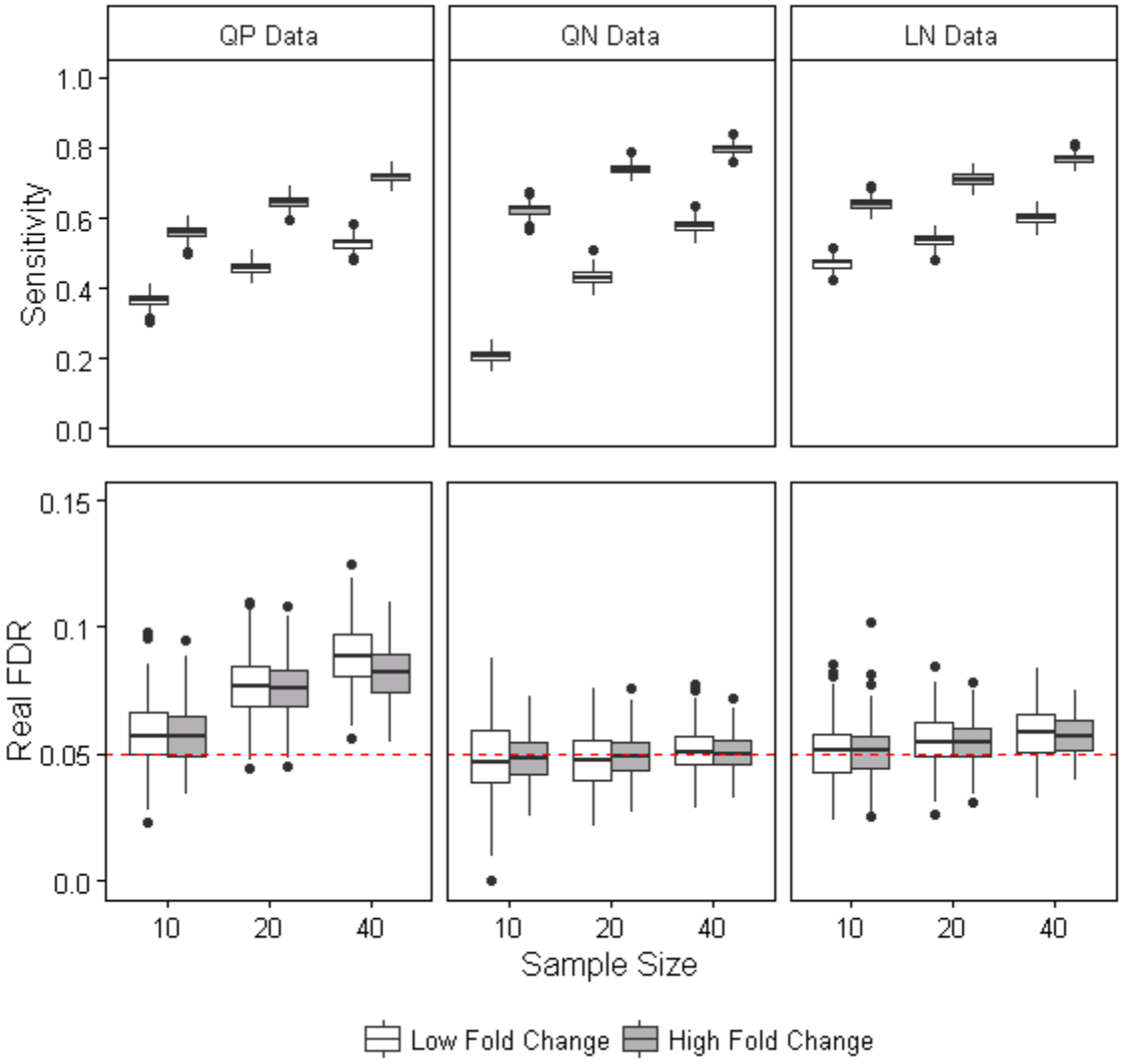

Figure 4. Sensitivity and Real FDR by Sample Size and Treatment Effect for Different Types of Data (NBQLSpline)

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

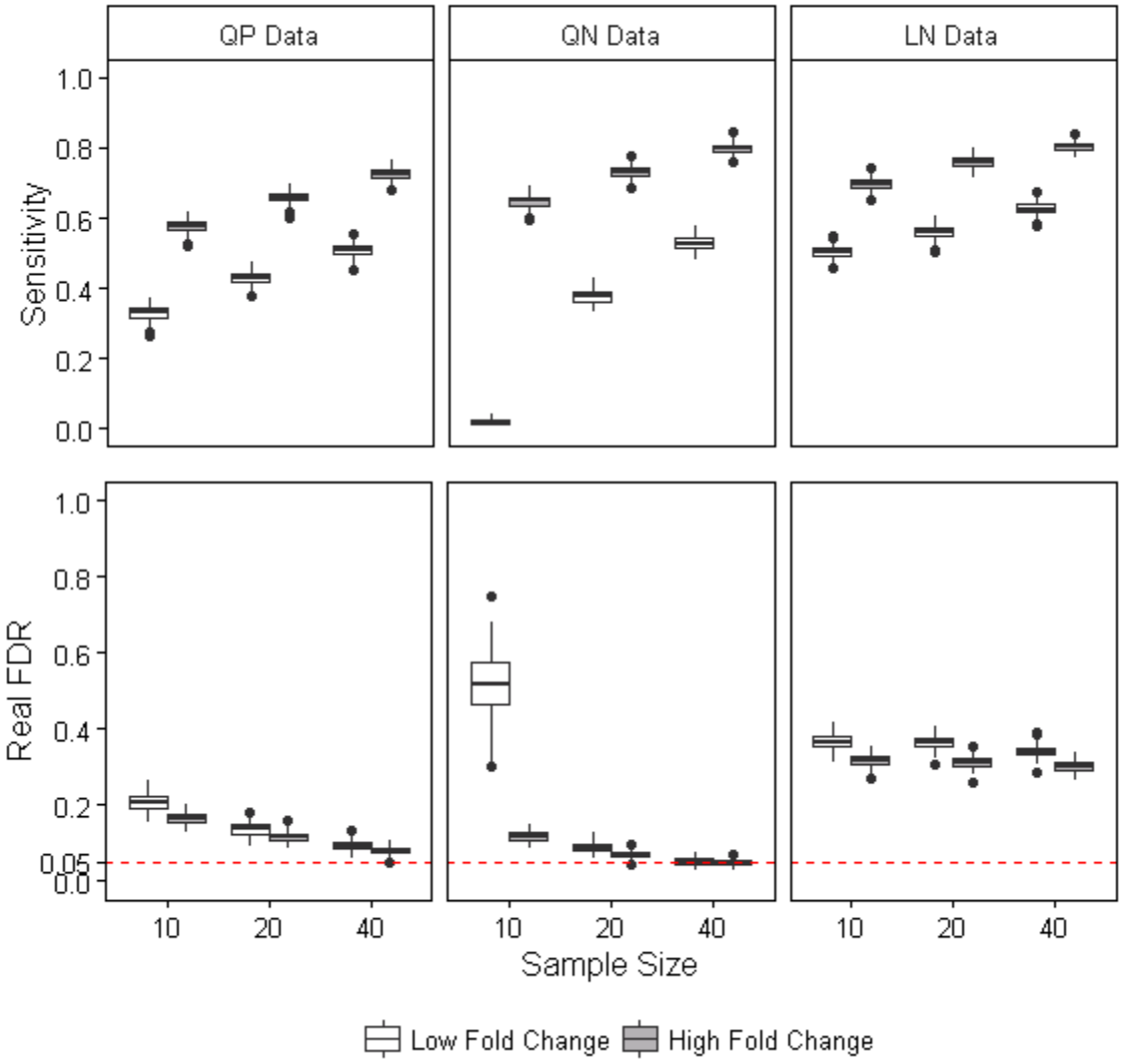

Figure 5. Sensitivity and Real FDR by Sample Size and Treatment Effect for Different Types of Data (edgeR)

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

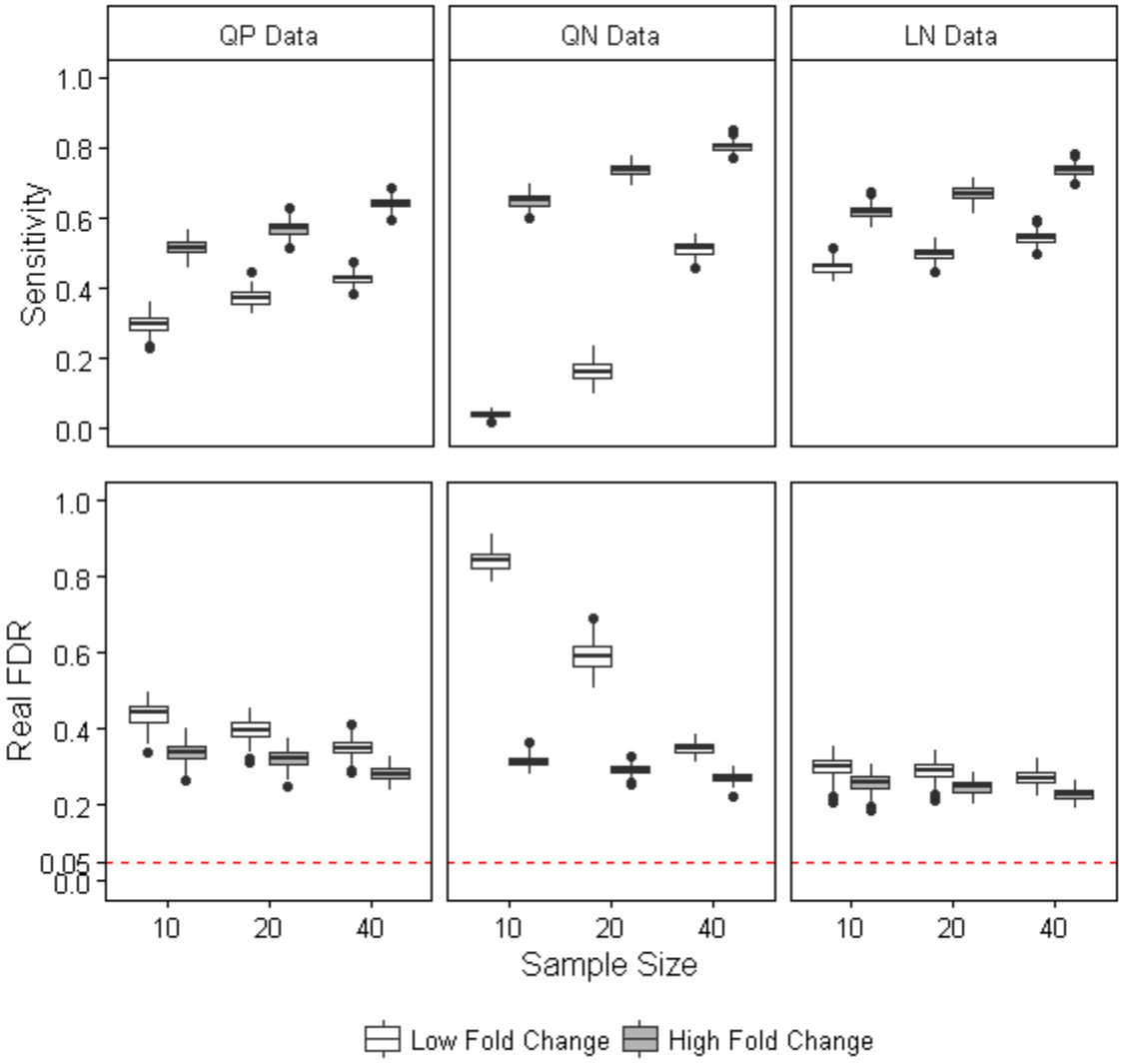

Figure 6. Sensitivity and Real FDR by Sample Size and Treatment Effect for Different Types of Data (DESeq)

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

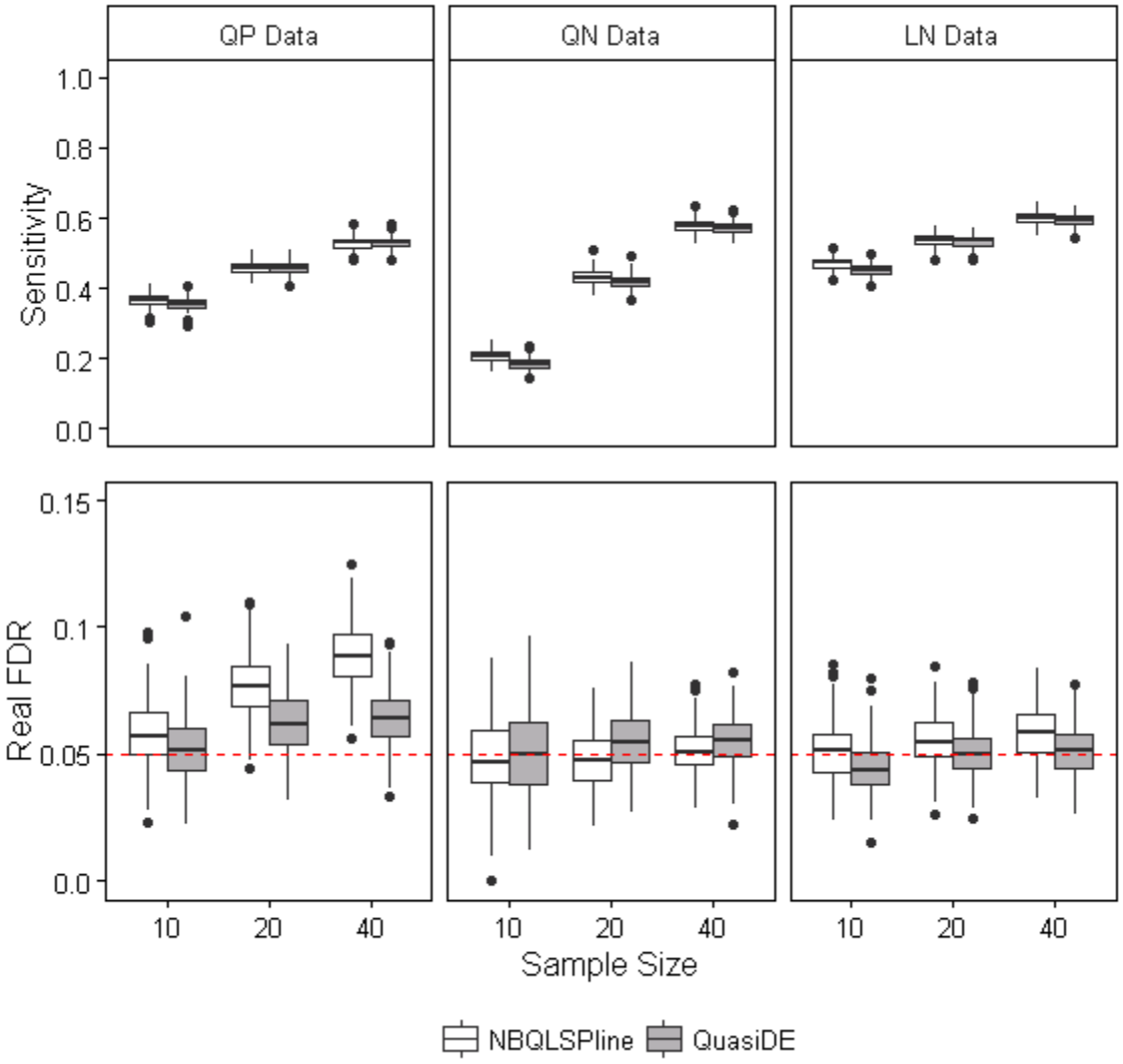

Figure 7. Sensitivity and Real FDR by Sample Size for Different Types of Data in Low Treatment Effect (NBQLSpline vs. QuasiDE)

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

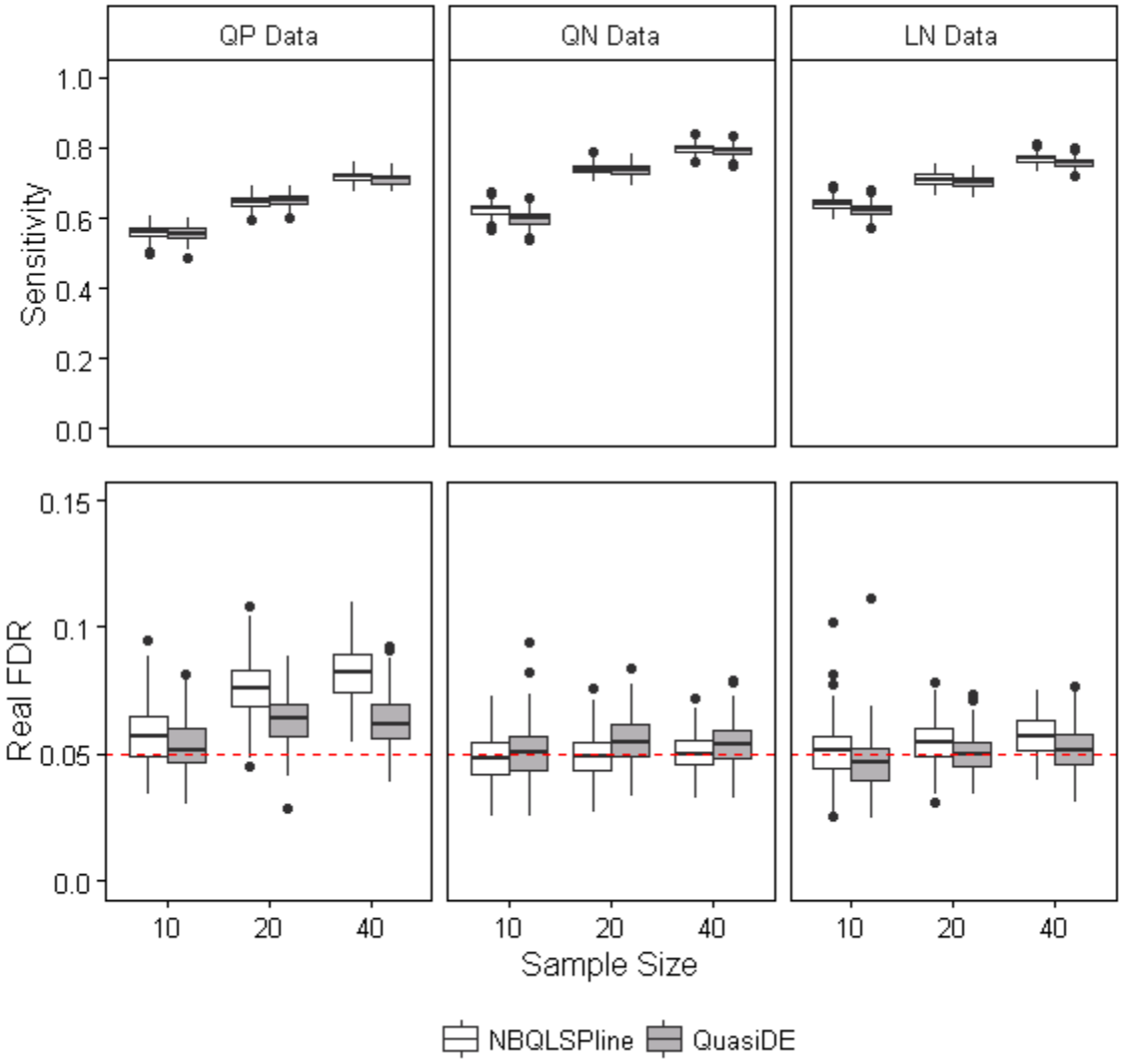

Figure 8. Sensitivity and Real FDR by Sample Size for Different Types of Data in High Treatment Effect (NBQLSpline vs. QuasiDE)

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

The simulation result for the NBQLSpline method is shown in Figure 4. The sensitivity also increases with sample size and treatment effect. The patterns of real FDR are quite different in three types of data. In the QP data, the real FDR increases with sample size. In the QN data, it is stable with sample size and almost perfectly controlled at nominal level of 0.05. In the LN data, it increases slightly with sample size. In the setting of sample size 10 and low treatment effect, NBQLSpline has a bit lower sensitivity which is similar as the QuasiDE method. The NBQLSpline method has the best performance in the QN data, in which the assumption of quadratic variance function is met.

The performance of the edgeR method is shown in Figure 5. The sensitivity has a similar pattern to the QuasiDE and NBQLSpline methods. More severely than the QuasiDE and NBQLSpline methods, the edgeR method loses sensitivity in the setting of sample size 10 and low treatment effect. The real FDR has different patterns across three types of data. In the QP data, the real FDR decreases with sample size; however, all the real FDRs are above the nominal level especially in the case of sample size 10. In the QN data, the real FDR decreases with sample size. High real FDRs are observed in the case of sample size 10 and low treatment effect due to the severe loss of sensitivity. In the LN data, the real FDR appears to decrease with increased treatment effect. All the real FDRs are at an unacceptable high level (above 0.25) in different sample sizes. Overall, only the analyses in the setting of the QN data with sample size 20 or 40 and the QP data with sample size 40 have the real FDR close to the nominal level.

The performance of the DESeq method is shown in Figure 6. The sensitivity has a similar pattern as the edgeR method. Similar to the edgeR method, it also loses

sensitivity severely in the QN data with sample size 10 and low treatment effect. The real FDR seems to be badly controlled in all the simulation settings. For all three types of data, most of the real FDRs are above 0.2.

Compared with the other three methods, the QuasiDE method has the most similar sensitivities and FDRs across the different types of data with different variance functions. Among all four methods, the performance of the NBQLSpline and QuasiDE methods are the best and similar. The edgeR method has the real FDR poorly controlled in most situations except when the sample size is 40 in the QP and QN data. The DESeq method has the real FDR badly controlled in all the simulation settings. The NBQLSpline and QuasiDE methods are further compared in Figure 7 and 8.

In Figure 7, the performances of the NBQLSpline and QuasiDE methods in the simulated datasets with low treatment effect are compared. The sensitivity of the NBQLSpline method is slightly higher than or similar to that of the QuasiDE method. The real FDRs appear to be better controlled in the QP and LN data for the QuasiDE method. In the QN data, the NBQLSpline method has the real FDR better controlled. As mentioned before, the reason for this is that the QN data meet the assumption of quadratic variance function used in the NBQLSpline method. In Figure 8, the performances of the NBQLSpline and QuasiDE methods in the simulated datasets with high treatment effect are compared. The result is similar to that shown in Figure 7. In all simulation settings, the performance of the QuasiDE and NBQLSpline methods is worse in the QP data compared with the QN and LN data. For the NBQLSpline method, this might be due to the violation of the quadratic variance function assumption. For the QuasiDE method, this might be due to the larger designed dispersion for the QP data,

the larger error in approximating the QP variance function using cubic spline and the performance of normalization in the QP data.

Overall, the QuasiDE and NBQLSpline methods have better performance than the edgeR and DESeq methods in simulation. Comparing the QuasiDE method with the NBQLSpline method, the QuasiDE method appears to have more similar sensitivities and FDRs against different variance functions whereas the performance of the NBQLSpline method deteriorates when the underlying assumption is not met.

With a Windows workstation with two processors (CPU speed 2.83 GHz) and 16 G memory, to analyze one 10-sample LN simulated dataset in high fold change setting, it takes about 10 mins, 4 mins, 7 secs and 2 mins on average for the QuasiDE, NBQLSpline, edgeR and DESeq methods respectively (Table 4). The longer time used by the QuasiDE method is mainly due to the numeric evaluation of the quasi-deviance for each gene. The current R program in Appendix A can be optimized and this deserves some further work.

Table 4: Computation Time in Simulation for Different Methods

Method	**Time in Seconds** Mean (SD*)
QuasiDE	578.7 (28.0)
NBQLSpline	245.8 (20.1)
edgeR	7.2 (0.5)
DESeq	149.1 (19.0)

** Standard Deviation*

Note: Based on 200 LN simulated datasets in high fold change setting

2.8 Case Study

The Squadrito data (Squadrito et al., 2014) is used to perform a real data analysis. Bone marrow (BM) cells were obtained from the long bones of eight-week old C57BL/6 mice. C57BL/6 is a common inbred strain of laboratory mouse. BM cells were then plated in macrophage medium with penicillin-streptomycin and CSF-1. After one week, they were polarized by adding interleukin-4 to the medium or left untreated. There are four biological replicates in each group. The research interest is to detect those DE genes between the treated and untreated groups. There are 31,020 genes in the dataset. In the data filtering, there are 11,477 genes having both zero interquartile range and low mean count (≤ 1). There are an additional six and 1,912 genes filtered by zero interquartile range alone and low mean count (≤ 1) alone respectively. After data

Table 5: Top 10 DE Genes by the QuasiDE Method and their Corresponding Results in the NBQLSpline and edgeR Methods (Squadrito Data)

Ensembl Gene ID / Chromosomal Location	Gene	QuasiDE		NBQLSpline		edgeR	
		Rank	Adjusted p-value	Rank	Adjusted p-value	Rank	Adjusted p-value
ENSMUSG00000070942	Il1rl2	1	3.2e-06	6	1.6e-11	16	5.5e-247
Chr16, BP 93140249-93142672		2	3.2e-06	304	7.6e-08	575	7.0e-031
ENSMUSG00000028125	Abca4	3	3.9e-06	4	1.4e-11	4.5	0
ENSMUSG00000000563	Atp5f1	4	8.6e-06	8	1.9e-11	4.5	0
ENSMUSG00000061100	Retnla	5	1.0e-05	1	7.3e-12	4.5	0
ENSMUSG00000016128	Stard13	6	1.0e-05	135	1.5e-08	311	1.3e-046
ENSMUSG00000047250	Ptgs1	7	1.5e-05	10	4.9e-11	15	1.0e-253
Chr14, BP 44396128-44511076		8	1.5e-05	2	7.3e-12	4.5	0
ENSMUSG00000025469	Msx3	9	1.5e-05	5	1.4e-11	19	1.7e-236
ENSMUSG00000074170	Plekhf1	10	1.5e-05	12	7.2e-11	4.5	0

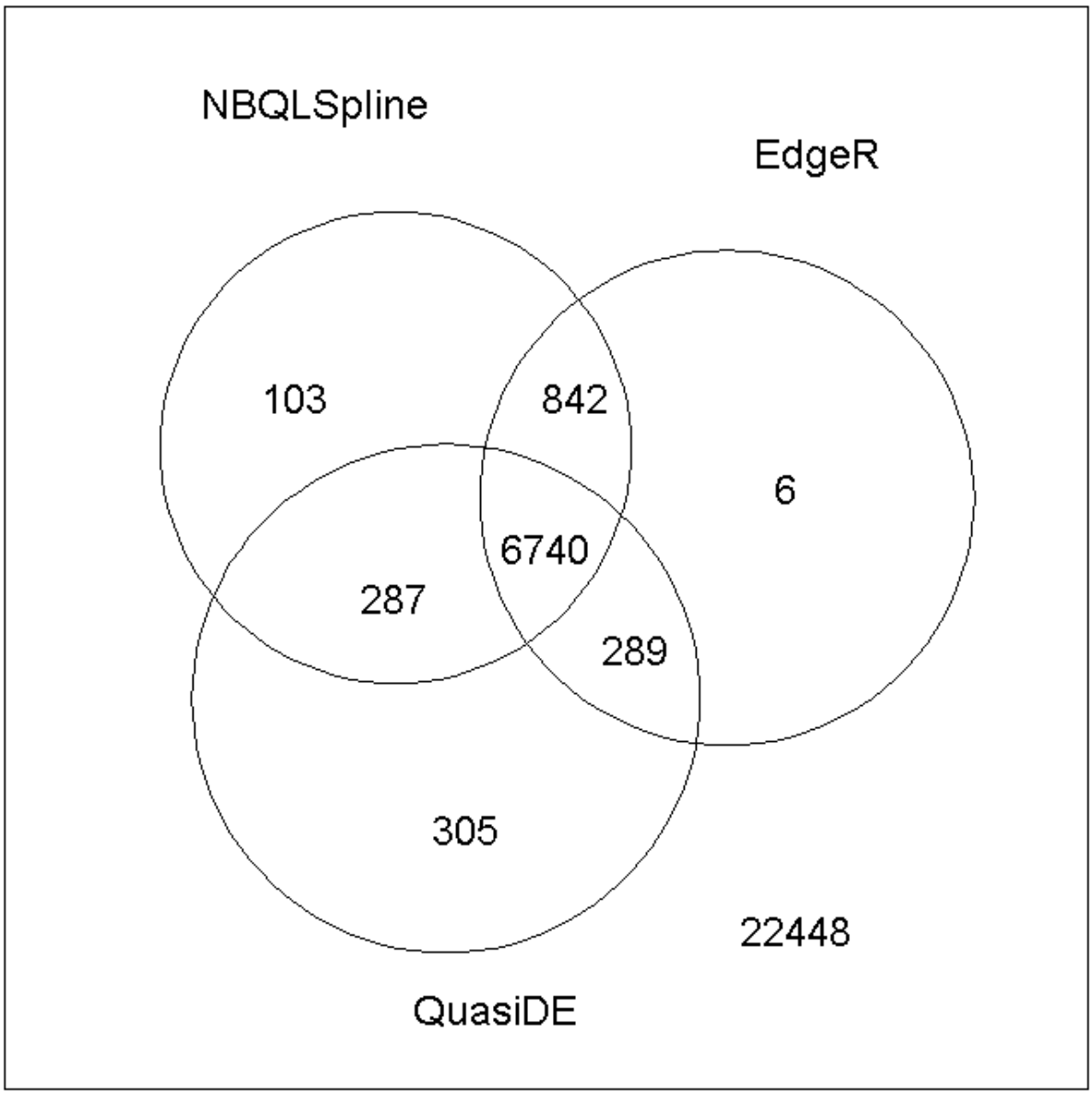

Figure 9. Venn Diagram of DE Gene Counts (Squadrito Data)

This shows the number of DE Genes detected by the QuasiDE, NBQLSpline and edgeR methods, and their relationship.

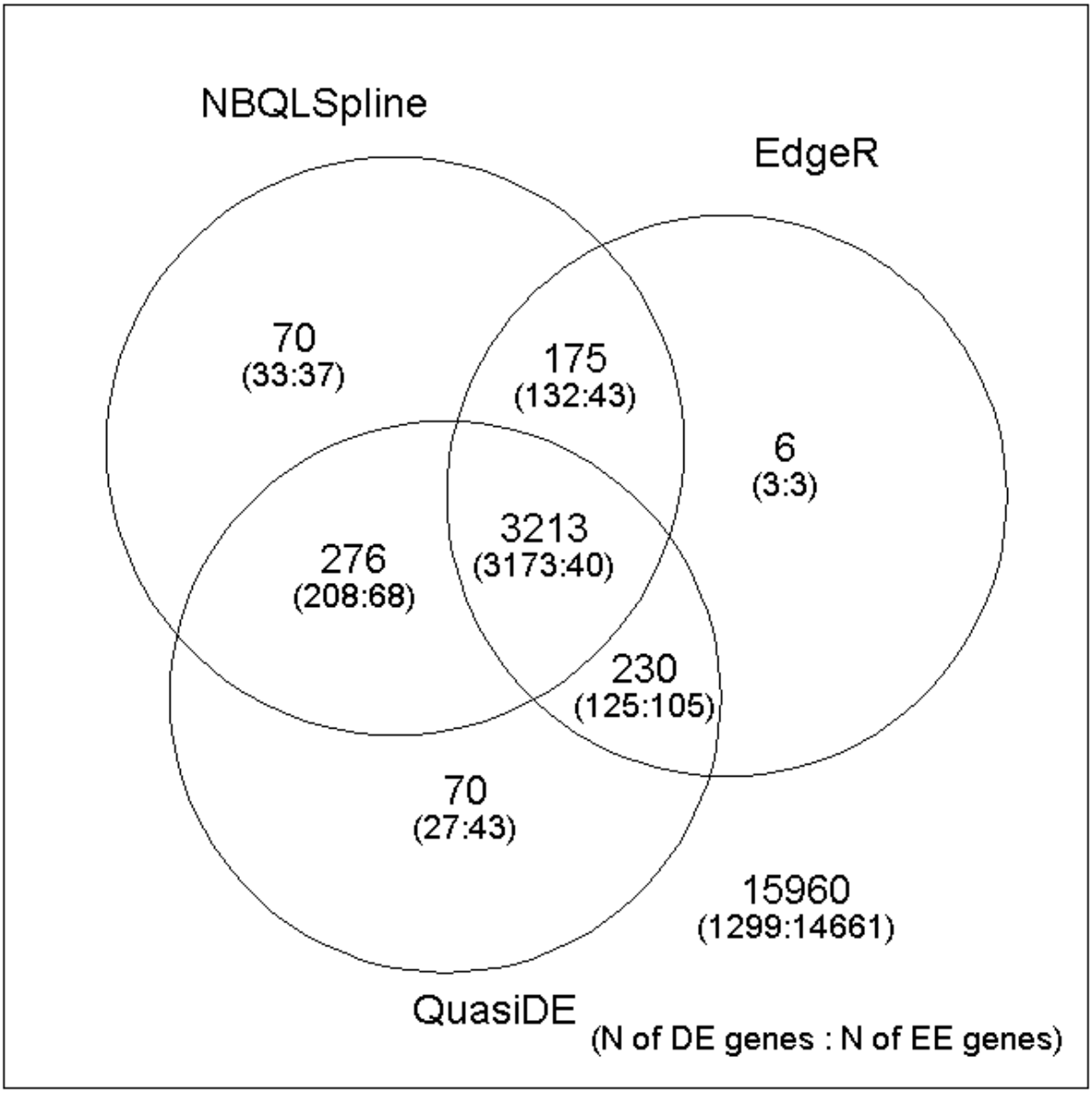

Figure 10. Venn Diagram of DE Gene Counts (results from a randomly chosen simulated dataset with 25% true DE genes)

The simulated dataset has the following settings: QN data, 2 experimental conditions with 5 samples in each condition and 20,000 genes. The diagram shows the number of DE Genes detected by the QuasiDE, NBQLSpline and edgeR methods, and their relationship. The number of true DE genes and true EE genes are shown in parenthesis.

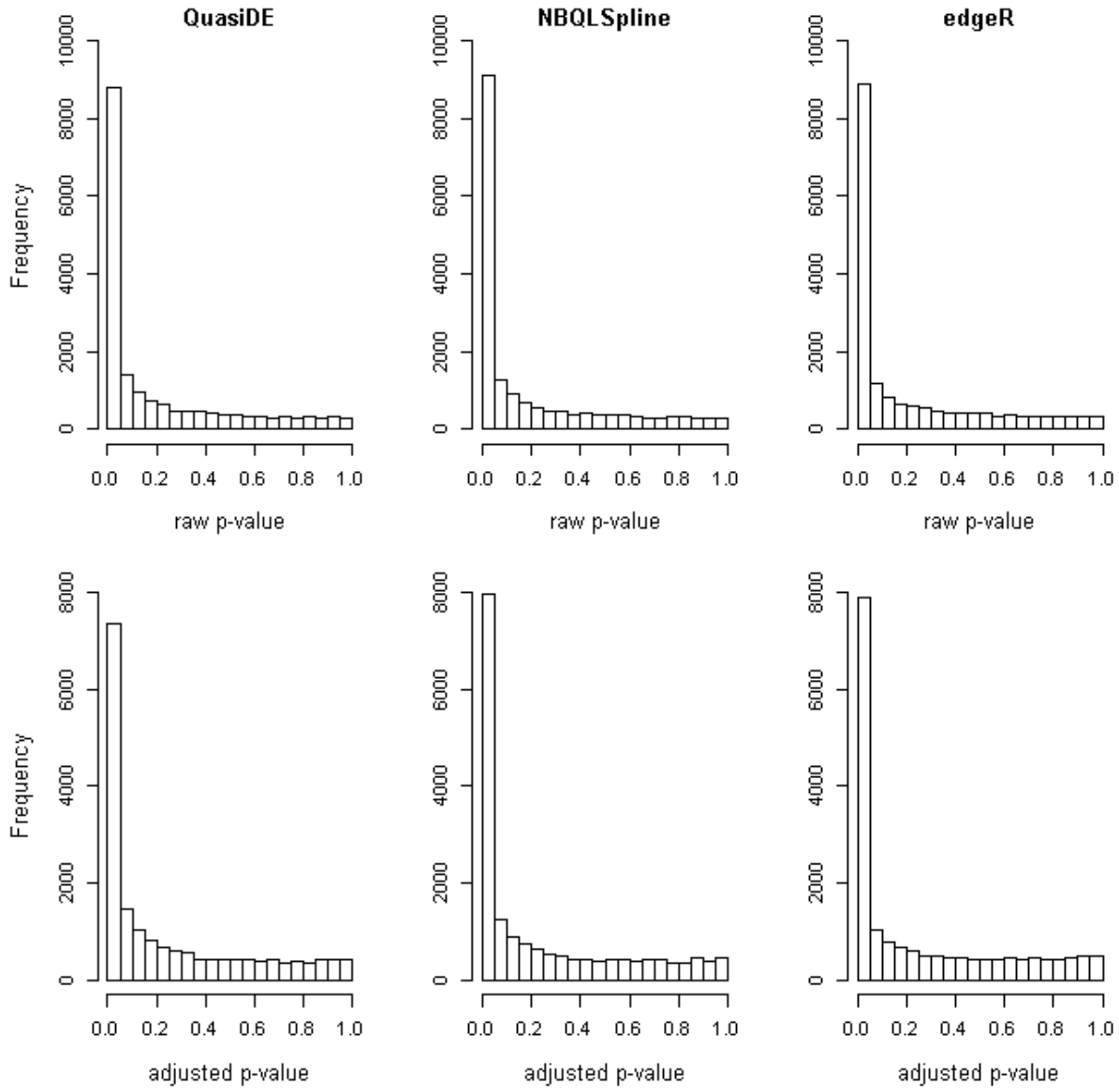

Figure 11. P-values and Adjusted p-values by the QuasiDE, NBQLSpline and edgeR Methods (Squadrito Data)

The panels in the first row show the distributions of the raw p-values by the QuasiDE, NBQLSpline and edgeR methods. The panels in the second row show the distributions of adjusted p-values by the QuasiDE, NBQLSpline and edgeR methods.

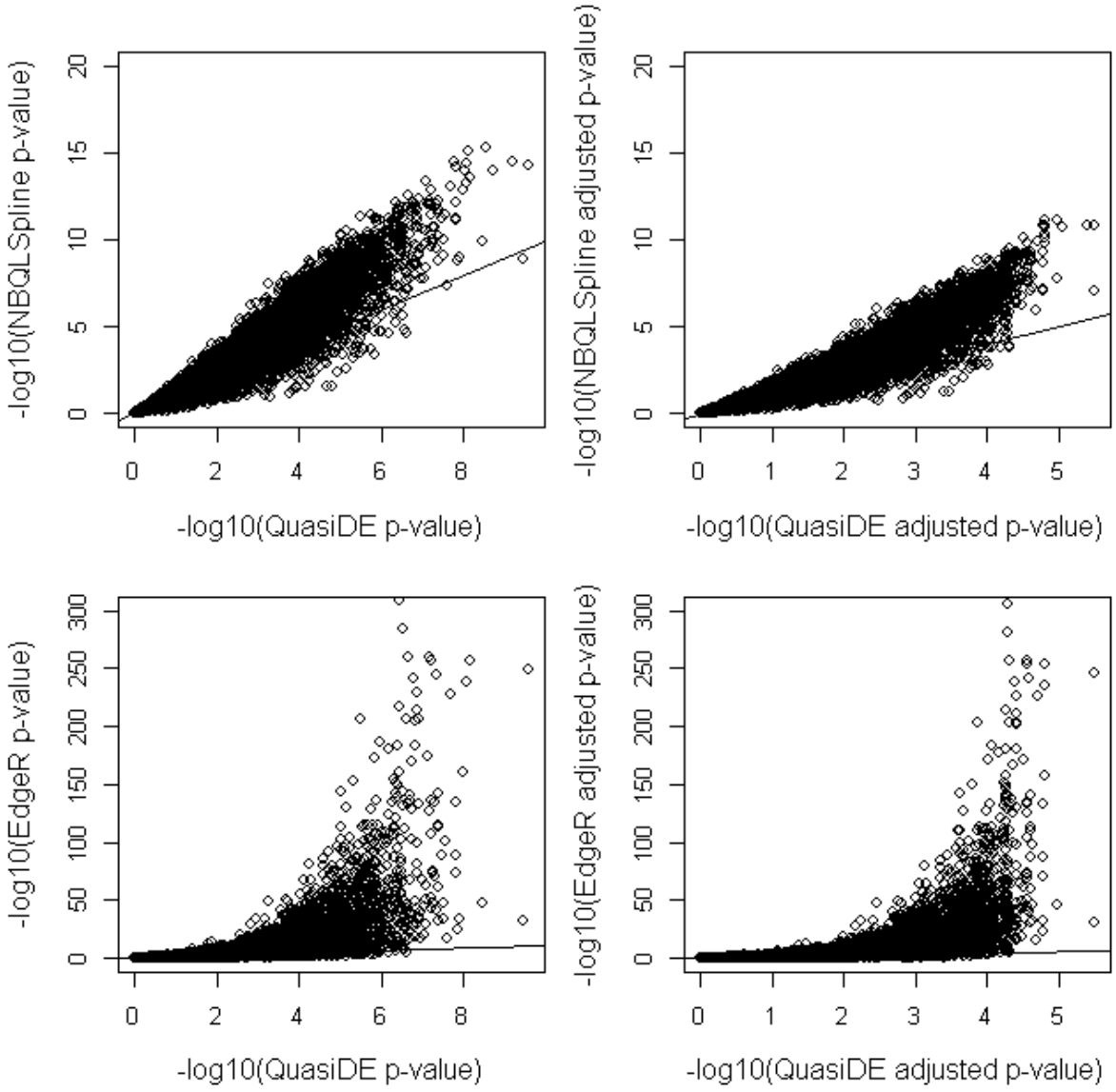

Figure 12. Comparison of the p-values and Adjusted p-values by the QuasiDE, NBQLSpline and edgeR Methods

The panels in the left column show the p-values by the QuasiDE method vs. p-values by the NBQLSpline and edgeR methods on -log10 scale. The panels in the right column show the adjusted p-values by the QuasiDE method vs. adjusted p-values by the NBQLSpline and edgeR methods on -log10 scale.

filtering, there are 17,625 genes remaining for the DE analysis. The TMM normalization is subsequently applied and the data are analyzed by the QuasiDE, NBQLSpline, edgeR and DESeq methods. The result of the DESeq method is not shown since it has badly controlled FDR in simulation.

The DE genes detected by the QuasiDE, NBQLSpline, and edgeR methods are compared in Figure 9. There are 7,972, 7,877, and 7,621 DE genes detected by the NBQLSpline, edgeR and QuasiDE methods respectively. In the original paper, the edgeR method was used and about 7,560 DE genes were identified. It is also interesting to compare this diagram with the corresponding diagram in simulation. For this purpose, 200 10-sample QN datasets are simulated with 25% DE ratio. One of these datasets is randomly chosen to show the DE genes detected by the QuasiDE, NBQLSpline, and edgeR methods. The simulation results of this dataset are shown in Figure 10. In Figure 10, the true status of the detected DE genes is also shown. The set of DE genes claimed by all three methods appears to have the lowest error rate. The sets of DE genes claimed by only one method appear to have the largest error rate. Compared Figure 9 with Figure 10, the DE genes detected by all three methods in Figure 9 are mostly true DE genes. The top 10 DE genes identified by the QuasiDE method are listed in Table 5 with the corresponding rank and adjusted p-values by the other two methods. For these 10 genes, the edgeR method appears to have relatively smaller adjusted p-values. The distributions of p-values and adjusted p-values by the QuasiDE, NBQLSpline, and edgeR methods are shown in Figure 11. The distributions of the raw p-values of all three methods are similar. The NBQLSpline and edgeR methods appear to have more small adjusted p-values (< 0.05) than the QuasiDE

Table 6: Reported DE Genes and their Corresponding Results for the QuasiDE, NBQLSpline and edgeR methods (Squadrito Data)

Gene	QuasiDE		NBQLSpline		edgeR	
	Rank	Adjusted p-value	Rank	Adjusted p-value	Rank	Adjusted p-value
Arg1	85	4.8e-05	16	2.8e-10	12	1.9e-257
Retnla	5	1.0e-05	1	7.3e-12	4.5	0
Chi3l3	1039	4.6e-04	1044	3.6e-06	1055	1.1e-18
Ccl22	1442	8.0e-04	873	1.8e-06	572	5.1e-31
Mrc1	6145	9.6e-03	4194	2.1e-03	3463	1.5e-05

method.

To explore the difference in small p-values (< 0.05) and small adjusted p-values (< 0.05), the p-values and adjusted p-values of the QuasiDE, edgeR and NBQLSpline methods are further compared on -log10 scale between the QuasiDE method and the other two methods in Figure 12. The small p-values for the QuasiDE method are similar to those from the NBQLSpline method. The edgeR method produced more extreme small p-values. The adjusted p-values have a similar pattern.

Those DE genes reported in the original paper (Squadrito et al., 2014) such as Arg1, Retnla, Chi3l3 (Ym1), Ccl22, and Mrc1 are also identified by all three methods. Their corresponding analysis results for the QuasiDE, NBQLSpline and edgeR methods are shown in Table 6.

2.8.1 Impact of DE ratio

The statistical methods have detected about 25% of the genes as DE genes in this study. The ratio is much higher than the 5% DE ratio used in the previous simulation. Here a small simulation has been conducted to understand the impact of DE ratio on

different statistical methods. In this simulation, sample size is chosen to be 10 and those DE genes are simulated to have 3-fold change between experimental conditions. The DE ratio is chosen to be 1%, 10%, 30%, 50%, 70%, and 90%. All the rest of the simulation settings are identical to the previous simulation. The impact on the QuasiDE method is shown in Figure 13. In Figure 13, the sensitivity increases with the DE ratio in all three types of data. In the QP and LN data, the real FDR decreases with the DE ratio and jumps to be above 5% at the DE ratio of 70%. In the QN data, the real FDR decreases with the DE ratio and jumps at the DE ratio of 70%, but not above 5%. When the DE ratio is round 25%, the real FDR appears to be overly controlled. The impact of the DE ratio on the NBQLSpline method is shown in Figure 14. In Figure 14, the sensitivity also increases with the DE ratio in all three types of data. In the QP and LN data, the real FDR has a small increase and then decreases with the DE ratio. At the DE ratio of 70%, the real FDR jumps to be around 10%. In the QN data, the real FDR slightly increases with the DE ratio. At the DE ratio of 70%, the real FDR jumps to be around 10%. When the DE ratio is round 25%, the real FDR appears to be close to 0.05. The impact of the DE ratio on the edgeR method is shown in Figure 15. The sensitivity also increases with the DE ratio in all three types of data. The real FDR decreases with the DE ratio. When the DE ratio is round 25%, the real FDR is at the level of 5% to 10%. This is close to the QuasiDE and NBQLSpline methods. The impact of the DE ratio on the DESeq method is shown in Figure 16. The sensitivity increases with the DE ratio as the other methods. The real FDR roughly decreases with the DE ratio. When the DE ratio is about 25%, the real FDR is around 10%.

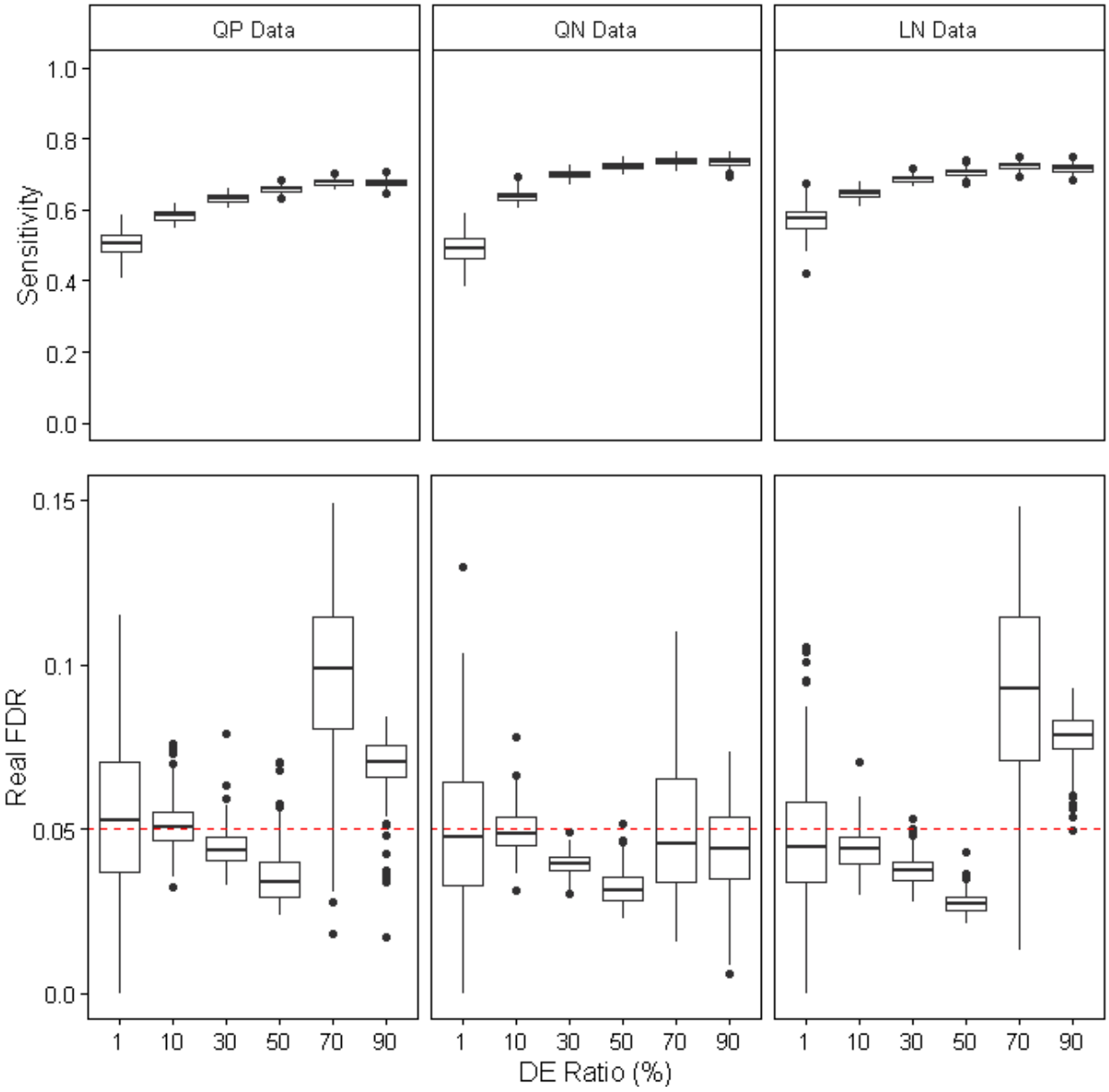

Figure 13. Sensitivity and Real FDR by DE ratio for Different Types of Data (QuasiDE)

The three panels in the first row show the sensitivity varying with DE ratio for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with DE ratio for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

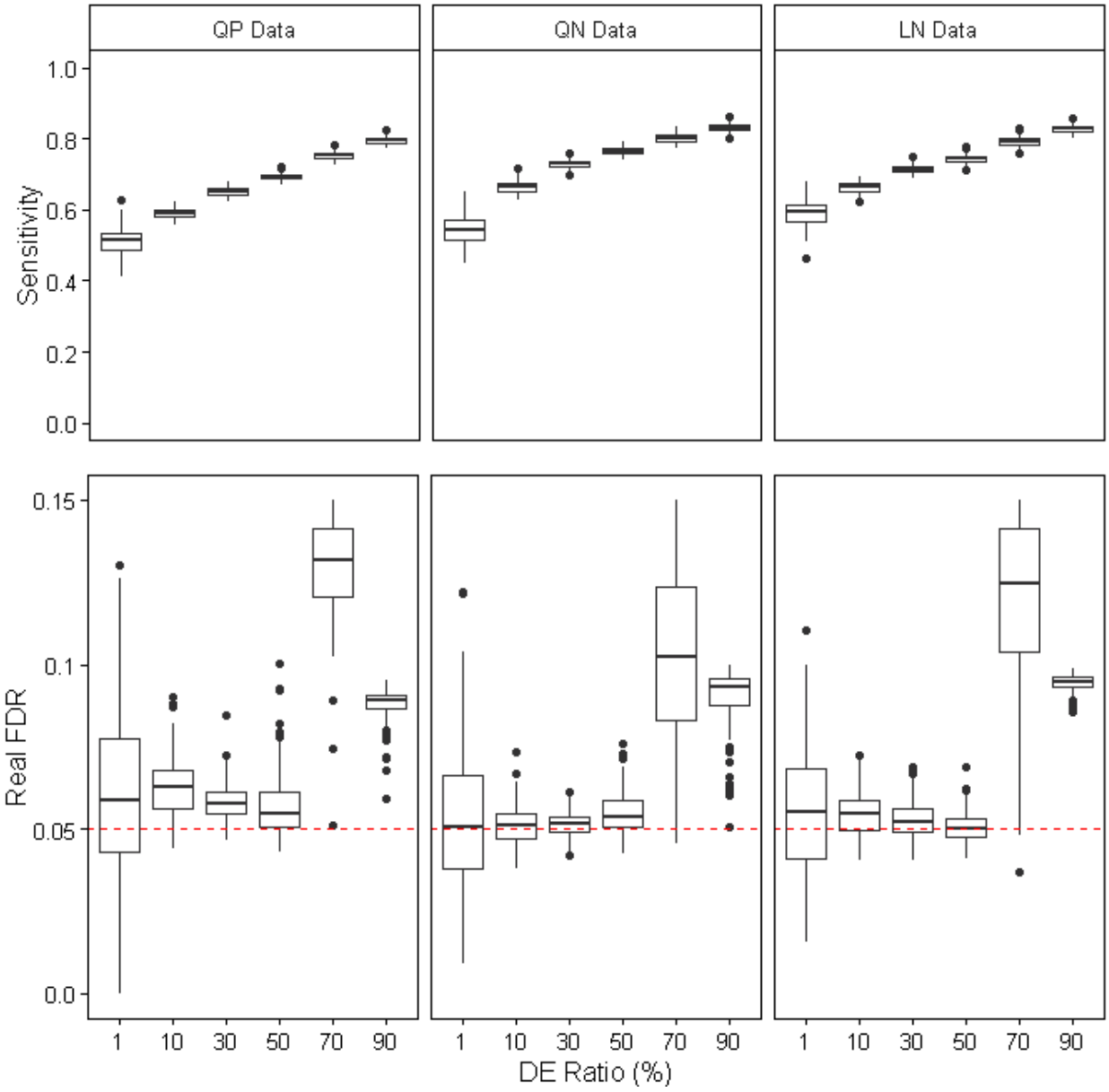

Figure 14. Sensitivity and Real FDR by DE ratio for Different Types of Data (NBQLSpline)

The three panels in the first row show the sensitivity varying with DE ratio for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with DE ratio for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

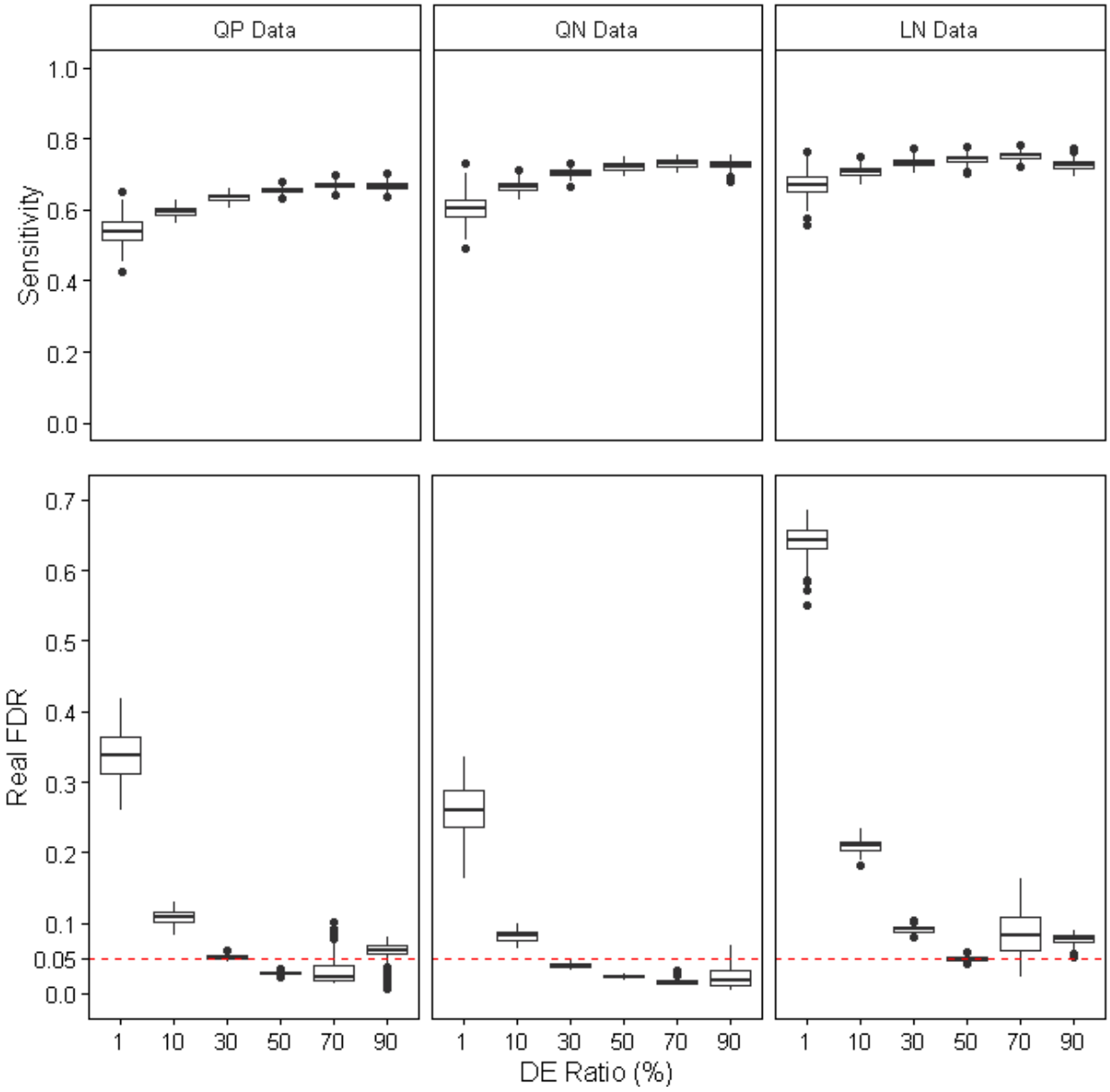

Figure 15. Sensitivity and Real FDR by DE ratio for Different Types of Data (edgeR)

The three panels in the first row show the sensitivity varying with DE ratio for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with DE ratio for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

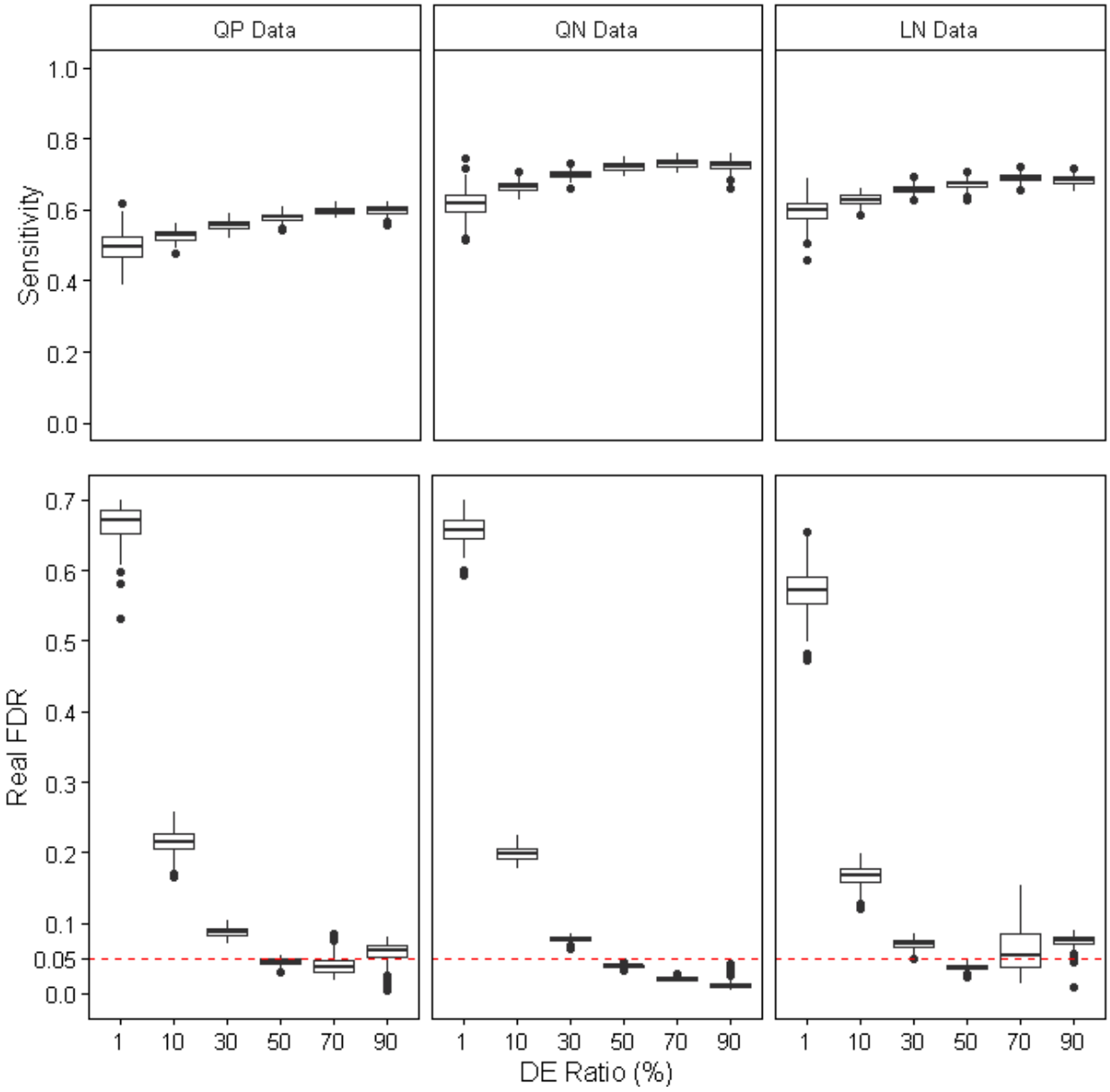

Figure 16. Sensitivity and Real FDR by DE ratio for Different Types of Data (DESeq)

The three panels in the first row show the sensitivity varying with DE ratio for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with DE ratio for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

2.9 Discussion

Analytically, a larger sample size, treatment effect or DE ratio tends to have a larger sensitivity for all four methods. For the real FDR, higher treatment effect tends to have a better controlled FDR for all four methods. The QuasiDE method tends to have a stable and well controlled FDR with a larger sample size. With a larger DE ratio (< 70%), the real FDR tends to be overly controlled. The NBQLSpline method has a worse FDR control when the data deviates from the NB distribution. In the QN data, the FDR is well controlled. The impact from a larger DE ratio is similar as the QuasiDE method. The edgeR method tends to have better controlled FDR in larger sample size in the QN data. In the data not NB distributed, the real FDR is worse controlled. With a larger DE ratio, the FDR tends to be better controlled. The DESeq method has a badly controlled FDR regardless of the sample size and data type. The impact from a larger DE ratio is similar as the edgeR method.

The edgeR and DESeq methods all assume the NB distribution. Their performance is not quite good in the QP and LN data where this assumption is not met. In the QN data, the NBQLSpline method has the best performance and the real FDR is almost perfectly controlled. In the QP and LN data, the real FDR is not controlled as well as in the QN data. The reason is that the QN data meet the assumption of a quadratic variance function.

For a real dataset, the distribution is unknown and may have a different underlying distribution other than the NB distribution. As a result, the assumptions in the edgeR, DESeq and NBQLSpline methods may be violated. The corresponding analysis might

be invalid. The advantage of the QuasiDE method is that it has similar sensitivity and real FDR for different types of data with different variance functions.

Beyond the three variance functions reported, the data were also simulated using a cubic function and a logistic function as the variance function, which gave the similar results. For the simulated data with a more complicated variance function, the QuasiDE method may fail. So do the other methods.

In the edgeR, DESeq and NBQLSpline methods, they all assume the NB distribution. When the data are not NB distributed, one possible way to make these methods robust is to use the robust pseudo-likelihoods, which offer the advantage of preventing the model misspecification effects. This was not explored in this dissertation.

When sample size is very large and most genes are not DE genes, the sensitivities of all four methods will presumably increase. Based on the previous simulation results, the real FDRs for the QuasiDE and NBQLSpline methods may have good control in the QN and LN data. In the QP data, the control of FDR may become stable and need further simulation to explore. The real FDR for the edgeR method may get better control of FDR in the QP and QN data. In the LN data, the real FDR may still be stably high based on the existing simulation results. For the DESeq method, the FDR may still be badly controlled.

The filtering method is important to all these statistical methods. With less stringent criteria, it might result in many false positives from those low expressed genes or genes with less variable expression. In a real analysis with many such positive genes in the result, one should be cautious. If those genes with low expression or less variable expression are of less interest, a more stringent filtering method can be applied.

In the simulation, a balanced design is assumed, that is, the sample size in each experimental condition is same. In RNA-seq experiment, it is very common to have sample imbalance. For example, the Dolatshad data are unbalanced. There are many factors causing the sample imbalance, such as the limit source of samples, budgetary constraints and reducing samples due to bad quality (Yang et al., 2006). In general, the balanced design is more powerful than the unbalanced design. The impact of mild imbalance on the power is negligible whereas the extreme imbalance can cause severe loss of the power.

The data requirement of the QuasiDE method is not limited to the count data. It can be applied to any continuous number, for example, to microarray data or to the RPKM data. Its performance in the RPKM data will be evaluated in Chapter 4.

The NBQLSpline, edgeR and DESeq methods have been extended to a more complicated design (Lund et al., 2012; McCarthy et al., 2012; Anders et al., 2016). The QuasiDE method in this chapter is limited to a two-group design. The methodology is extended to a couple of other common study designs in the next chapter.

2.10 Conclusion

In this chapter, a new quasi-likelihood method is proposed. The proposed method has no distributional assumption. Therefore it has similar sensitivities and FDRs against different types of data with different variance functions compared with other existing methods. Moreover, the performance of the new method is also better than the existing methods in most situations: the sensitivity is approximately the same as other popular methods while it has the real FDR better controlled in most situations.

CHAPTER 3. QuasiDE in Other Study Designs

3.1 Introduction

Any good statistical analysis method for RNA-seq experiments should be sufficiently flexible to analyze a general experimental design such as the design with multiple experimental conditions, block design and factorial design. For example, linear models (Smyth, 2004) have been successfully applied in microarray data, which allows the analysis of a complex design. For the RNA-seq data analysis, extensions of the existing statistical methods have the same capability. This includes extensions of the edgeR, DESeq and NBQLSpline methods through generalized linear models (GLM). In this chapter, the QuasiDE method is extended to achieve similar flexibility.

3.2 Statistical Methods Review

3.2.1 edgeR

McCarthy et al. (2012) extended the edgeR method to a more complex experimental design. Recall P_{ig} is the relative abundance of the g^{th} gene in the i^{th} experimental condition. The associated log-linear model is

$$log[E(y_{ijg})] = X_{ij}^T \beta_g + log(L_{ij}) \quad (28)$$

where $log(P_{ig}) = X_{ij}^T \beta_g$, X_{ij}^T is a row in the design matrix corresponding to the j^{th} sample in the i^{th} experimental condition. β_g is the coefficient vector for the g^{th} gene. If considering normalization, the effective library sizes are used. The NB dispersion parameter is estimated based on a complex design.

To estimate the NB dispersion parameter ϕ_g in a complex design, the adjusted profile likelihood (APL) method (Cox and Reid, 1987) is used. The APL for ϕ_g is the penalized log-likelihood:

$$APL_g(\phi_g) = \ell(\phi_g; y_{ijg}, \hat{\beta}_g) - \frac{1}{2} log[det(\mathfrak{I}_g)] \quad (29)$$

where $\hat{\beta}_g$ is the estimated coefficient vector in the NB GLM, ℓ is the log-likelihood function, $\mathfrak{I}_g$is the Fisher information of β_g evaluated at $\hat{\beta}_g$ and ϕ_g. The regression coefficient β_g is assumed to be orthogonal to ϕ_g. The $\hat{\beta}_g$ is the maximum likelihood estimate of β_g given ϕ_g, which is also a function of ϕ_g.

It has been shown that empirical Bayes methods give more reliable estimates in microarray analysis (Smyth, 2004). In the RNA-seq data analysis, it is not possible to apply the empirical Bayes method directly since there is no conjugate prior distribution for the NB dispersion ϕ_g. A weighted likelihood can be used instead to approximate the empirical Bayes strategy by sharing information across genes.

As mentioned in the previous chapter, the NB dispersion parameter can be estimated from the data by assuming 1) all genes have a constant dispersion parameter; or 2) there is a trend between dispersion parameter and average count; or 3) the dispersion parameter is gene-specific.

A common dispersion is the simplest way to share information across genes. The common dispersion is estimated by maximizing the common APL, which is defined as

$$APL_c(\phi) = \frac{1}{G} \sum_{g=1}^{G} APL_g(\phi) \quad (30)$$

The common APL can be considered as a weighted likelihood where each gene has the same weight.

To estimate the trended dispersion, NB dispersion is modeled as a smooth function of gene-specific average expression level. The gene-specific average expression is calculated from the GLM using the common dispersion and the library size. All genes are then sorted by their average expression values. A locally shared APL for the g^{th} gene is denoted as $APL_s(\phi_g)$, which is a weighted average of APLs of the g^{th} gene and the set of genes which have closest average expression values. The weight for gene a in this gene set is determined by the tricube function $w_a = (1 - |x_a|^3)^3$, where $-1 < x_a < 1$ represents the scaled difference in average expression values for genes g and a. This function assigns larger weight to those genes with expression level closer to the g^{th} gene. The trended dispersion for the g^{th} gene can be estimated by maximizing $APL_s(\phi_g)$.

To esimtate the gene-specific dispersion, the dispersion for the g^{th} gene is estimated by maximizing:

$$APL_g(\phi_g) + G_0 APL_s(\phi_g) \tag{31}$$

where $APL_s(\phi_g)$ is the local shared log-likelihood and G_0 is the weight to be optimized. These three types of dispersion estimation are implemented in functions estimateGLMCommonDisp, estimateGLMTrendedDisp, and estimateGLMTagwiseDisp respectively in R package edgeR. In this dissertation, the gene-specific NB dispersion is used in the edgeR method.

With the Cox-Reid dispersion estimates, Fisher's scoring iteration is used in the NB GLM method to estimate the parameter β_g. This could fail to converge in some datasets since it is not guaranteed to produce an increase in the likelihood function in each iteration. To ensure the convergence for all genes and all datasets, Fisher-scoring

algorithm has been enhanced by Levenberg damping modification (Osborne 1992; McCarthy et al., 2012). The new algorithm forces a reduction in the residual deviance in each iteration. The testing procedure uses the GLM likelihood ratio test, which is based on the NB GLM with the Cox-Reid dispersion estimates.

The edgeR method can also be extended with quasi-likelihood theory. This is considered in Lund's work (2012). In their work, the F statistic based on the quasi-likelihood theory uses the quasi-likelihood dispersion estimated from Equation (14) to test DE genes.

3.2.2 DESeq

Recall that the raw count y_{ijg} is assumed to follow the NB distribution with mean $\eta_{ij}\delta_{ig}$, where η_{ij} denotes the size factors for the j^{th} sample in the i^{th} experimental condition, δ_{ig} is a value proportional to the true concentration of the g^{th} gene in the i^{th} experimental condition. In a complex design, there are two approaches to estimate the NB dispersion parameter (Anders and Huber, 2016). The first approach is to estimate the gene-specific dispersion in each subgroup. The overall gene-specific dispersion is calculated as the weighted average of all these subgroup estimates (weight is the number of samples in each subgroup). The second approach is to estimate the gene-specific dispersion by fitting the NB GLM to maximize the Cox-Reid adjusted profile likelihood. This method was developed by McCarthy et al. (2010) in the edgeR method. In the DESeq method, the expression for the Cox-Reid adjusted profile likelihood has been optimized and the weighted maximum likelihood method is not used. The second approach in dispersion estimation is typically used when there are no replicates in some of the subgroups. The first approach is used in this dissertation to estimate the gene-

specific NB dispersion parameter in the DESeq method. A smooth curve is then fitted between the gene-specific dispersion $\hat{\alpha}_g$ and $\hat{\delta}_g$. The fitted values from this curve are used as dispersion estimates for inference.

After NB dispersion estimation, the NB GLMs are fitted for the full model and the reduced model with log link. The two models are compared by the likelihood ratio test. This is implemented in function fitNbinomGLMs in R package DESeq.

3.2.3 NBQLSpline

The NBQLSpline method is a sub-approach in QuasiSeq R package assuming a quadratic variance function (Lund et al., 2012). Recall that the gene-specific NB dispersion parameters are estimated by the edgeR method. The default setting in the NBQLSpline method is to use the trended dispersion. In a complex design, the dispersion for the g^{th} gene is estimated using local shared adjusted profile likelihood, which is a weighted average of APLs of the g^{th} gene and the set of genes which have closest average expression levels. With the trended dispersion estimate, a NB GLM is fitted for each gene. The full model and the reduced model are compared with an F test, which is constructed using the quasi-deviance of these two models and a shrunken quasi-likelihood dispersion estimate (QLSpline). This is implemented in function QL.fit in R package QuasiSeq.

3.3 QuasiDE Extension to a Complex Design

In the QuasiDE method, the variance function is estimated by the within-group mean-variance pairs. In a complex design, there might be more groups than in a simple two-group design. The variance function is proposed to be estimated by the smooth cubic spline using the mean-variance pairs for the normalized counts in all subgroups. In the

experiment with two conditions, the GLM with the quasi-likelihood method is used. In a complex design, a GLM model can also be used through the quasi-likelihood method with the estimated variance function. Just as the experiment with two conditions, the GLM model with log link is used. The quasi-score function is as follows:

$$U_g\left(\beta_{1g}, \beta_{2g}, \dots, \beta_{Mg}\right) = \boldsymbol{\mathcal{D}}_{\boldsymbol{g}}^{T}\boldsymbol{V}_{\boldsymbol{g}}^{-1}(\boldsymbol{z}_{\boldsymbol{g}} - \boldsymbol{\mu}_{\boldsymbol{g}})/\Phi_g \tag{32}$$

where M denotes the total number of regression coefficients in the GLM, $\boldsymbol{\mu}_{\boldsymbol{g}}$ is the expected normalized count in a specified model (any model includes the full model or the reduced model), $\boldsymbol{\mathcal{D}}_{\boldsymbol{g}}$ is $N \times M$ matrix:

$$\boldsymbol{\mathcal{D}}_{\boldsymbol{g}} = \begin{pmatrix} \frac{\partial \mu_{1g}}{\partial \beta_{1g}} & \frac{\partial \mu_{1g}}{\partial \beta_{2g}} & \cdots & \frac{\partial \mu_{1g}}{\partial \beta_{Mg}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mu_{Ng}}{\partial \beta_{1g}} & \frac{\partial \mu_{Ng}}{\partial \beta_{2g}} & \cdots & \frac{\partial \mu_{Ng}}{\partial \beta_{Mg}} \end{pmatrix} \tag{33}$$

and $\boldsymbol{V}_{\boldsymbol{g}}$ is same as Equation (20) which is a $N \times N$ diagonal matrix.

The estimating equations $U_g\left(\beta_{1g}, \beta_{2g}, \dots, \beta_{Mg}\right) = 0$ are solved to obtain the coefficient estimates $\hat{\beta}_{1g}, \hat{\beta}_{2g}, \dots, \hat{\beta}_{Mg}$. The fitted values for the full model and the reduced model are denoted as $\hat{z}_{ijg,full}$ and $\hat{z}_{ijg,reduced}$ respectively. The implementation uses the glm function in R and the detailed R code is in Appendix A.

The fitted values are then used to calculate the quasi-deviance. Suppose there are S subgroups and the number of samples in the s^{th} subgroup is N_s. The quasi-deviance for a specified model is:

$$\widetilde{D}_{g,model} = \sum_{s=1}^{S}\sum_{j=1}^{N_s} 2\int_{\hat{z}_{sjg,model}}^{z_{sjg}} \frac{z_{sjg} - t}{\hat{V}(t)}\,dt \tag{34}$$

where z_{sjg} is the observed normalized count of the g^{th} gene for the j^{th} sample in the s^{th} subgroup and $\hat{z}_{sjg}, model$ is the fitted value of the g^{th} gene for the j^{th} sample in the s^{th} subgroup for a specified model.

In an experiment with multiple conditions, the ANOVA-like comparison involves comparing the quasi-deviances of two models: the model with group means and the model with an overall mean. In a block design, the treatment effect is tested by comparing the quasi-deviances of two models: the model with both the treatment effect and the block effect and the model with the block effect only. In a factorial design, the effect of interaction is tested by comparing the quasi-deviances of two models: the model with both the main effects and an interaction effect and the model with the main effects only. For all these model comparisons, the test statistic is constructed as an F statistic $\Delta\widetilde{D}_g/(q\widehat{\Phi}_g)$, where $\Delta\widetilde{D}_g$ is the difference of the quasi-deviances between the two models to be compared, q is the difference of degrees of freedom between the two models, and $\widehat{\Phi}_g$ is the dispersion estimated from Equation (25). The test statistic is compared with an F distribution with q and $N-p$ degrees of freedom, where p is the number of parameters in the full model. In an experiment with multiple conditions, the full model is the model with group means. In an experiment with a block design, the full model is the model with both the treatment effect and block effect. In a factorial design, the full model is the model with both the main effects and interaction effect.

3.4 Simulation

To examine the performance of the QuasiDE method in a complex design, a series of simulations have been conducted.

Table 7: DE Patterns in Experiments with Multiple Conditions

DE Pattern Name	Subgroup Means
EMC 1	$\mu, 1.5\mu, 3\mu$
EMC 2	$\mu, 2\mu, 2\mu$

Note: μ represents the average count randomly sampled from the within-group mean normalized counts in the Himes data.

3.4.1 Experiments with Multiple Conditions (EMC)

The data are simulated based on the three types of variance functions listed in Table 2. Sample sizes are chosen to be 15, 30, and 60, split evenly across the three experimental conditions. Simulated genes with zero total count are replaced with a new simulated gene. In each dataset, 20,000 genes are simulated and 5% of them are chosen to be DE (1000 DE and 19,000 EE genes). The DE genes are designed to have one of the patterns listed in Table 7. These two patterns are referred to as “EMC1” and “EMC2” hereafter. Compared with the EMC1, The EMC2 is believed to be more difficult to detect. In simulation, the average counts of three experimental conditions in Table 7 are set in random order. In each scenario (per sample size, variance function, and DE pattern), 200 datasets are simulated. The rest of simulation is done the same as described in Section 2.6.

3.4.2 Block Design

The data are also simulated based on three types of variance functions listed in Table 2. Sample sizes are chosen to be 20, 40, and 80, split evenly across two experimental conditions and two blocks. Simulated genes with zero total count are replaced with a

Table 8: DE Patterns in those Genes with Treatment Effect in Block Design

	Block	Experimental Conditions		Number of Genes
		1	2	
Pattern 1	**1**	μ	2μ	75
	2	2μ	4μ	
Pattern 2	**1**	2μ	μ	75
	2	4μ	2μ	
Pattern 3	**1**	2μ	4μ	75
	2	μ	2μ	
Pattern 4	**1**	4μ	2μ	75
	2	2μ	μ	
Pattern 5	**1**	μ	2μ	350
	2	μ	2μ	
Pattern 6	**1**	2μ	μ	350
	2	2μ	μ	

Note: μ represents the average count randomly sampled from the within-group mean normalized counts in the Himes data.

new simulated gene. In each dataset, 20,000 genes are simulated. Of all the genes, 5% of them are DE genes due to the treatment effect. Of these genes, 30% of them are also DE genes due to the block effect. The treatment effect and block effect are all assumed to be two-fold change. The 5% DE genes having treatment effect are assigned to have one of the following patterns listed in Table 8.

Of those genes with no treatment effect (19,000 genes), 30% of them are simulated as DE genes due to the block effect. Of these 30% of the genes, half of them are over-expressed and half of them are under-expressed for the block effect. The rest of simulation is done the same as described in Section 2.6.

3.4.3 Factorial Design

The data are also simulated based on three types of variance functions listed in Table 2. Sample sizes are chosen to be 20, 40, and 80, split evenly across two levels of factor one and two levels of factor two. Simulated genes with zero total count are replaced with a new simulated gene. In each dataset, 20,000 genes are simulated. Five percent of them are chosen to have true non-zero main effects and interaction; Five percent of them are chosen to have a true non-zero main effect for factor one only; Five percent of them are chosen to have a true non-zero main effect for factor two only; Five percent of them are chosen to have true non-zero main effects for both factors but no interaction. The rest of the genes have zero main effects and interaction. The main effects of Factor one and two are set as 1.5 and 2 fold changes respectively. The genes with both main

Table 9: DE Patterns in those Genes with Main Effects and Interaction in Factorial Design

	Factor Two	Factor One	
		1	**2**
Pattern 1	**1**	μ	1.5μ
	2	2μ	4μ
Pattern 2	**1**	1.5μ	μ
	2	4μ	2μ
Pattern 3	**1**	4μ	2μ
	2	1.5μ	μ
Pattern 4	**1**	2μ	4μ
	2	μ	1.5μ

Note: μ represents the average count randomly sampled from the within-group mean normalized counts in the Himes data.

Table 10: DE Patterns in those Genes with only Main Effects from both Factor One and Factor Two in Factorial Design

	Factor Two	Factor One	
		1	**2**
Pattern 1	**1**	μ	1.5μ
	2	2μ	3μ
Pattern 2	**1**	1.5μ	μ
	2	3μ	2μ
Pattern 3	**1**	3μ	2μ
	2	1.5μ	μ
Pattern 4	**1**	2μ	3μ
	2	μ	1.5μ

Note: μ represents the average count randomly sampled from the within-group mean normalized counts in the Himes data.

effects and interaction are randomly assigned one of the patterns listed in Table 9. The genes with the main effects from both factor one and two are designed to have one of the patterns in Table 10. In simulation, those genes with only one main effect are randomly assigned as over-expressed or under-expressed. The DE genes with the main effects and interaction effect in Table 9 are balanced among all the subgroups. The DE genes with two main effects are also balanced based on the four patterns in Table 10. The rest of simulation is done the same as described in Section 2.6.

3.5 Simulation Results

3.5.1 Experiments with Multiple Conditions

The performance of the QuasiDE method in experiments with multiple conditions is shown in Figure 17. The performance is similar as in the setting of two experimental conditions. The sensitivity increases with sample size in all settings. The sensitivity to

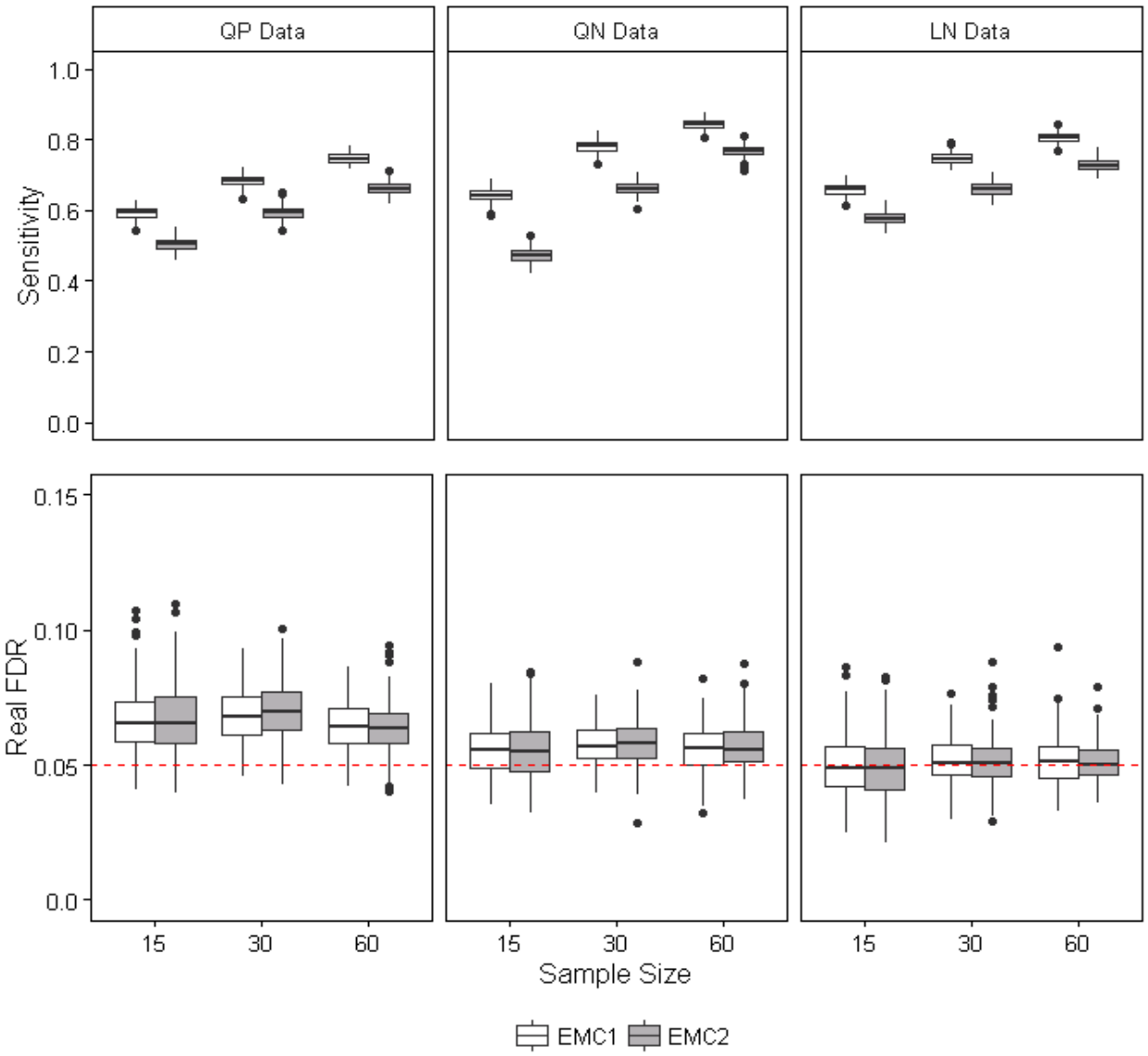

Figure 17. Sensitivity and Real FDR by Sample Size for Different Types of Data (QuasiDE) in Design with Three Experimental Conditions

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR in the same setting. EMC1 and EMC2 are different designed DE patterns (Table 7). In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

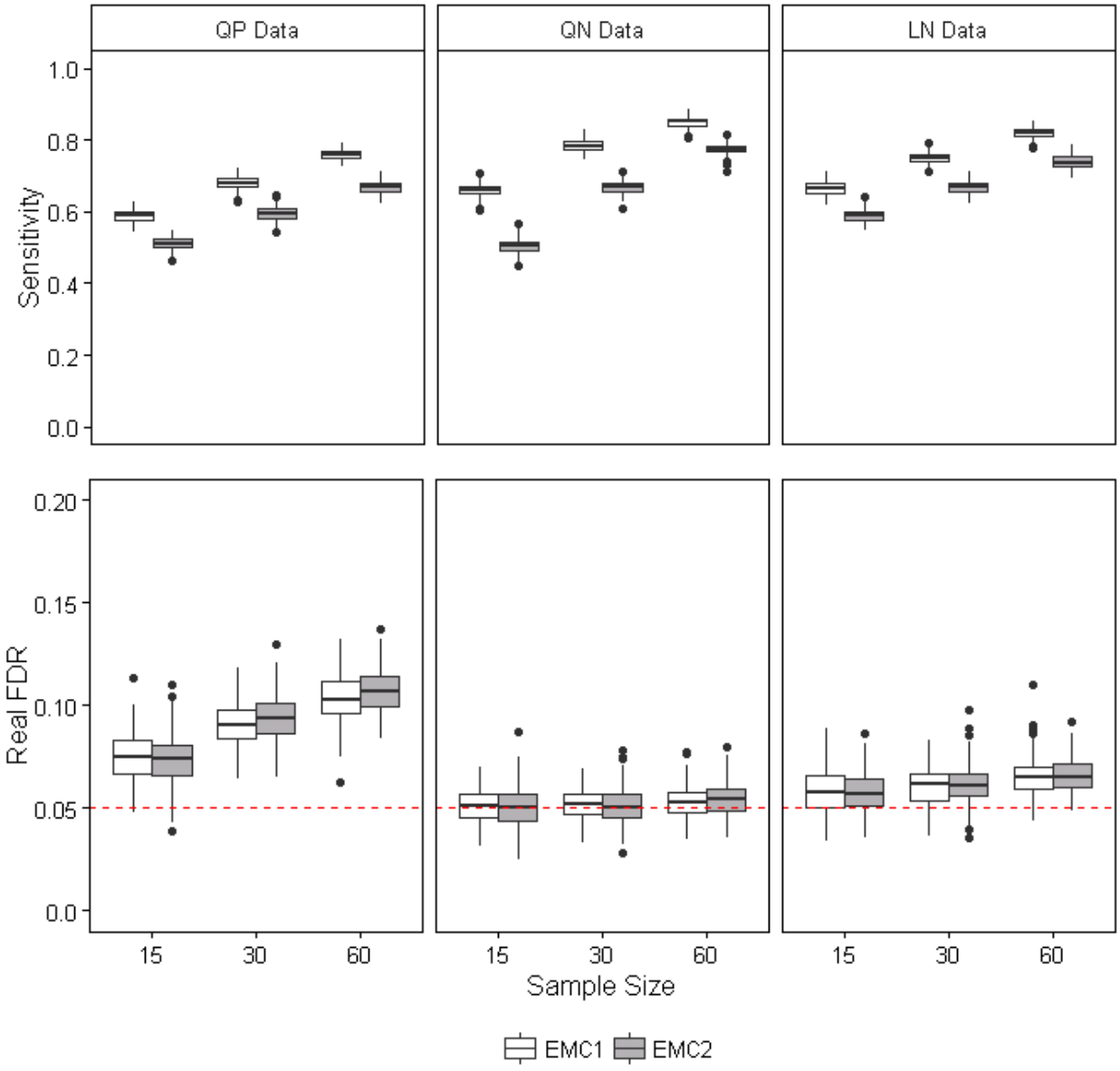

Figure 18. Sensitivity and Real FDR by Sample Size for Different Types of Data (NBQLSpline) in Design with Three Experimental Conditions

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR in the same setting. EMC1 and EMC2 are different designed DE patterns (Table 7). In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

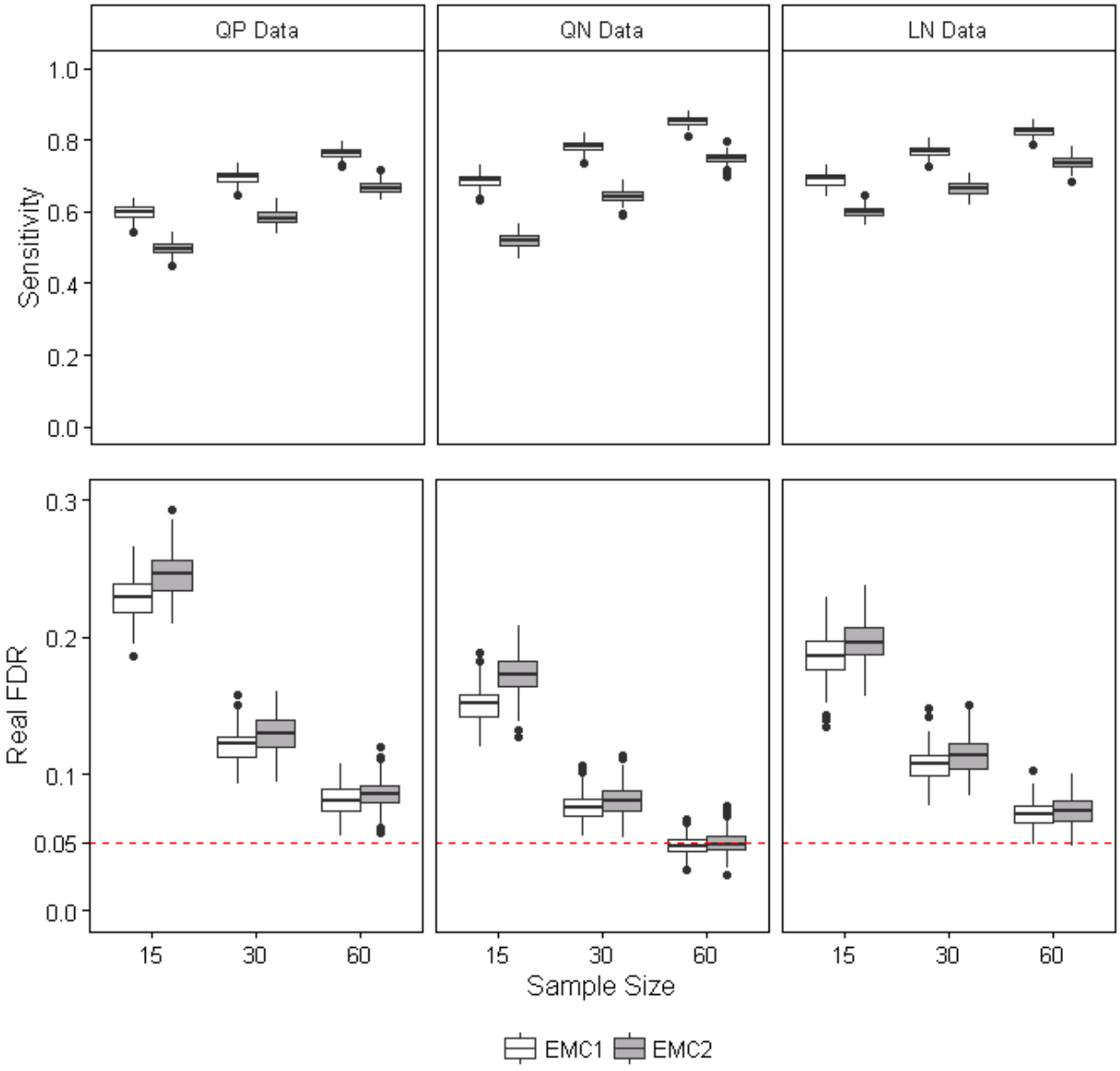

Figure 19. Sensitivity and Real FDR by Sample Size for Different Types of Data (edgeR) in Design with Three Experimental Conditions

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR in the same setting. EMC1 and EMC2 are different designed DE patterns (Table 7). In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

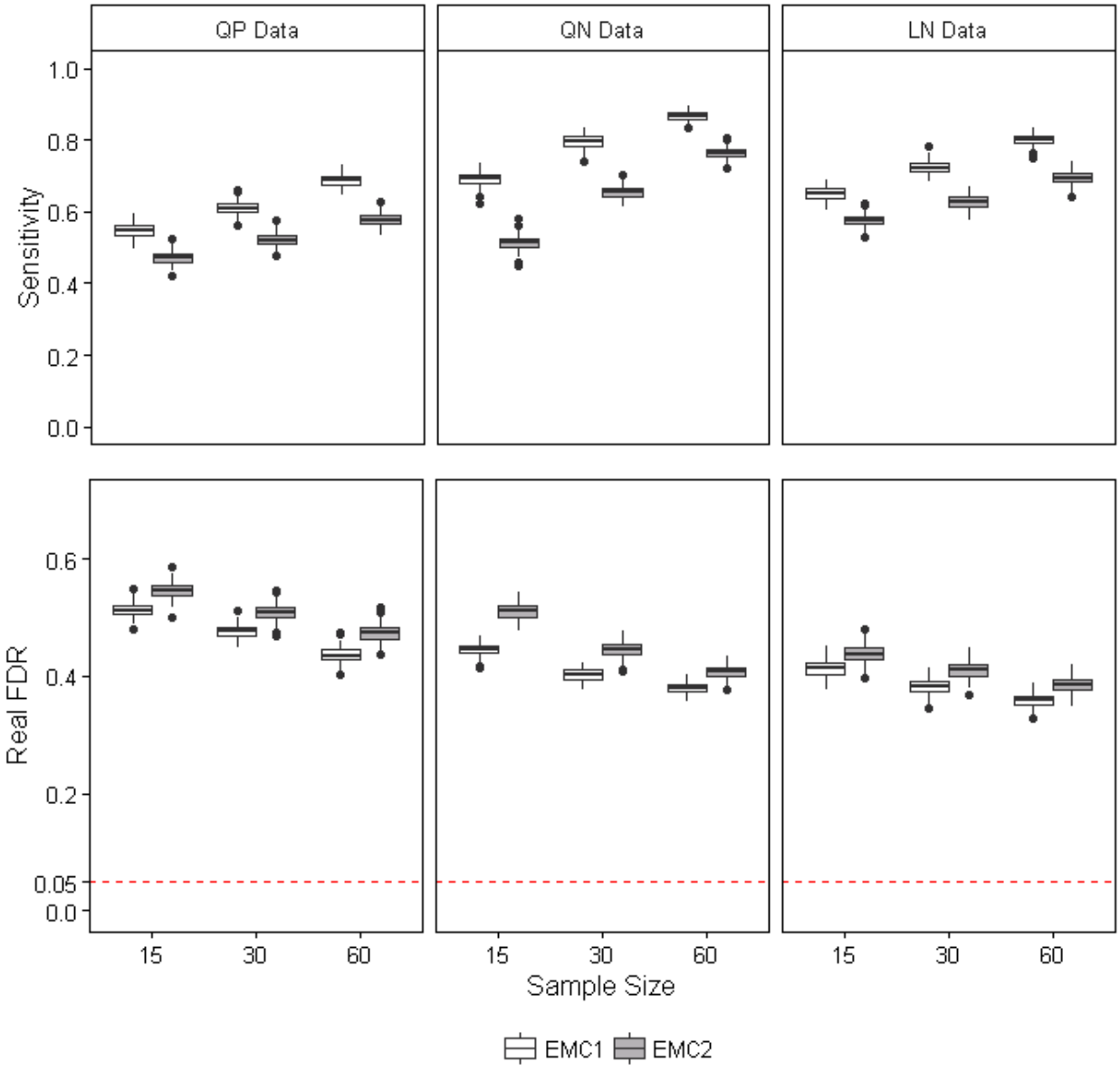

Figure 20. Sensitivity and Real FDR by Sample Size for Different Types of Data (DESeq) in Design with Three Experimental Conditions

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR in the same setting. EMC1 and EMC2 are different designed DE patterns (Table 7). In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

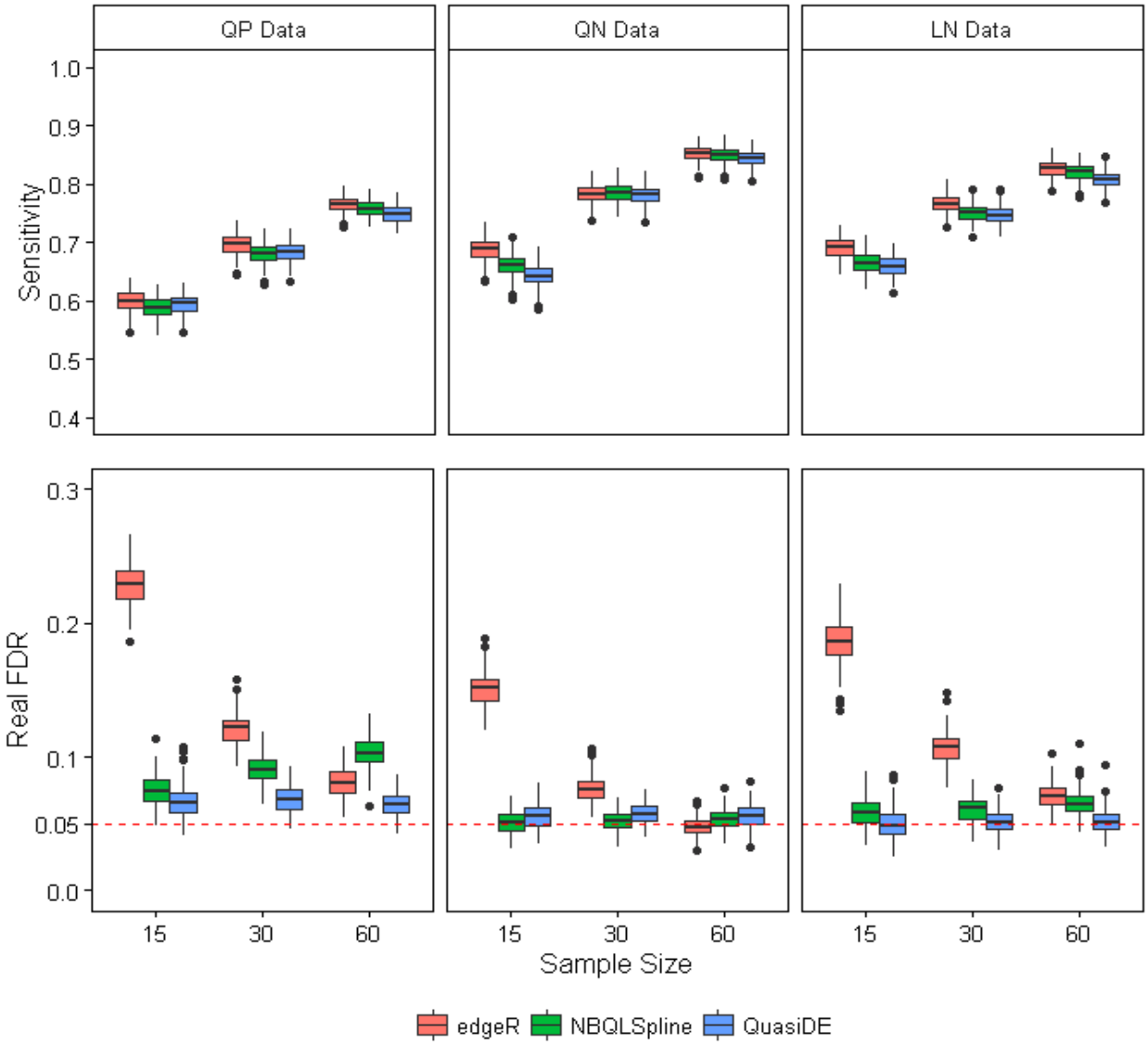

Figure 21. Sensitivity and Real FDR by Sample Size for Different Types of Data and Statistical Methods (edgeR, NBQLSpline and QuasiDE) in EMC1

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR in the same setting. The red, green and blue colors are used for edgeR, NBQLSpline and QuasiDE respectively. EMC1 is the designed DE pattern (Table 7). In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

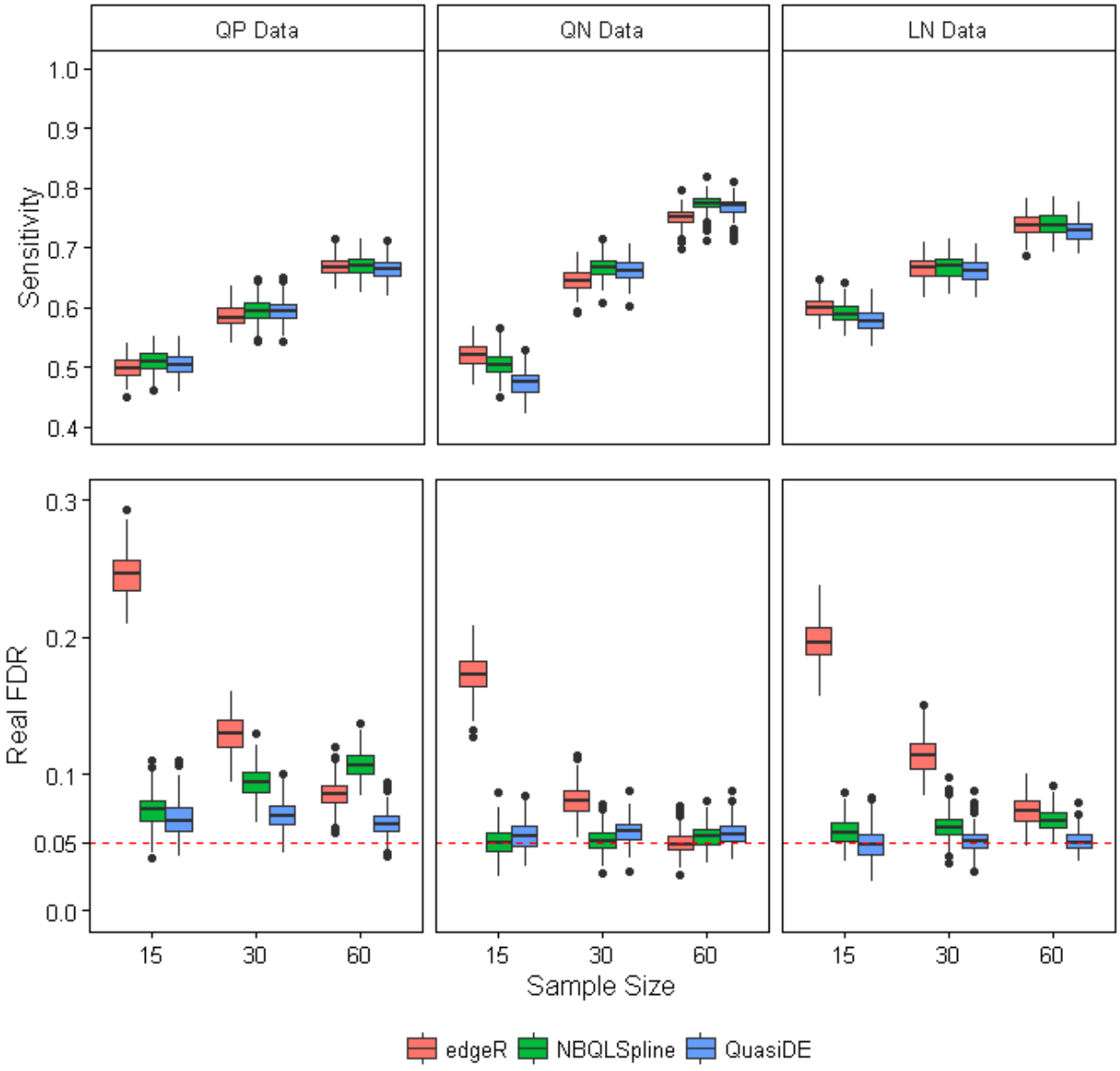

Figure 22. Sensitivity and Real FDR by Sample Size for Different Types of Data and Statistical Methods (edgeR, NBQLSpline and QuasiDE) in EMC2

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR in the same setting. The red, green and blue colors are used for edgeR, NBQLSpline and QuasiDE respectively. EMC2 is the designed DE pattern (Table 7). In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

detect the EMC2 is lower than that to detect the EMC1 since the EMC1 is designed to be easier to detect compared with the EMC2. The real FDR is stable with different sample sizes. The real FDRs in the QN and LN data appear to be similar and the real FDRs in the QP data are slightly higher. In the QP data, the increase of FDR from sample size 15 to 30 is less than that increase from sample size 10 to 20 in simulation of two-condition experiments. This might be due to a larger total sample size.

The performance of the NBQLSpline, edgeR and DESeq methods are shown in Figure 18, 19 and 20. In Figure 18, the sensitivity of the NBQLSpline method has a similar pattern as the QuasiDE method. The real FDR increases with sample size in the QP and LN data. In the QN data, the real FDR is almost perfectly controlled. This is also observed in the simulation of an experiment with two conditions. The performance is still the best in the QN data since the QN data meets the assumption of quadratic variance function in the NBQLSpline method. In Figure 19, the sensitivities of the edgeR method show a similar pattern as the QuasiDE and NBQLSpline methods. The real FDR decreases with sample size in all three types of data. In the setting of sample size 60 in the QN data, the edgeR method has the real FDR almost perfectly controlled. In all other settings, it has worse controlled real FDRs than those from the QuasiDE and NBQLSpline methods. In Figure 20, the real FDRs of DESeq are very badly controlled with all observed FDRs greater than 0.3 despite the criterion that the FDR should be controlled at 0.05.

The edgeR, NBQLSpline and QuasiDE methods are further compared in Figure 21 and 22 for EMC1 and EMC2 respectively. In Figure 21, the QuasiDE method appears to have a slightly smaller sensitivity compared with the other two approaches. The real

FDRs of the QuasiDE method appear to be the best controlled among these three approaches in the QP and LN data. In the QN data, the NBQLSpline method has a better controlled real FDR. In Figure 22, the three approaches have similar patterns as shown in Figure 21. Overall, the QuasiDE method has the most similar sensitivities and FDRs across three types of data, which is similar as the results in experiments with two conditions.

3.5.2 Block Design

The performance of the QuasiDE method is compared with the NBQLSpline method in Figure 23. The sensitivity is about the same for these two approaches. The real FDR is better controlled for the QuasiDE method in the QP and LN data. In the QN data, the real FDRs of the NBQLSpline method are slightly better. This is not a surprise since the QN data perfectly meet the quadratic variance function assumption in the NBQLSpline method. The performance of the QuasiDE method is compared with the edgeR method in Figure 24. The sensitivities are similar for these two approaches. The QuasiDE method has better controlled real FDRs in almost all situations except when the sample size is 40 in the QN data. In that setting, the real FDRs of these two approaches are competitive. In the QN data with sample size 80, the edgeR method has overly controlled FDRs. The performance of the QuasiDE method is compared with the DESeq method in Figure 25. The sensitivity of the two approaches is about the same. The DESeq method has very badly controlled real FDR in all the simulation settings. Overall, the performance of the QuasiDE method is at least as good as the other three approaches in the block design simulation. The sensitivities are similar to those from the other three approaches in all settings. In most simulation settings, the QuasiDE method

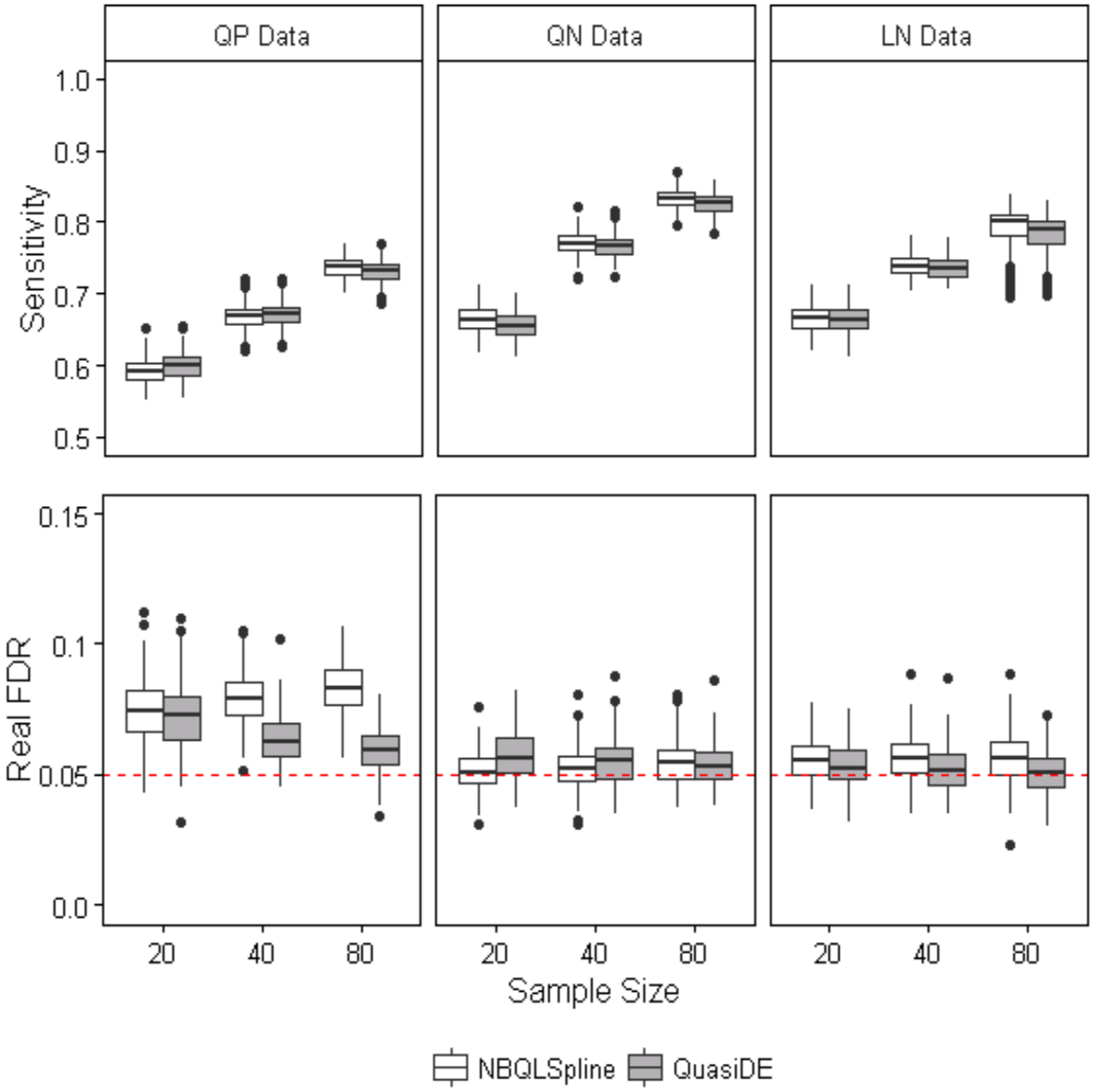

Figure 23. Sensitivity and Real FDR by Sample Size for Different Types of Data (NBQLSpline vs. QuasiDE) in Block Design

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

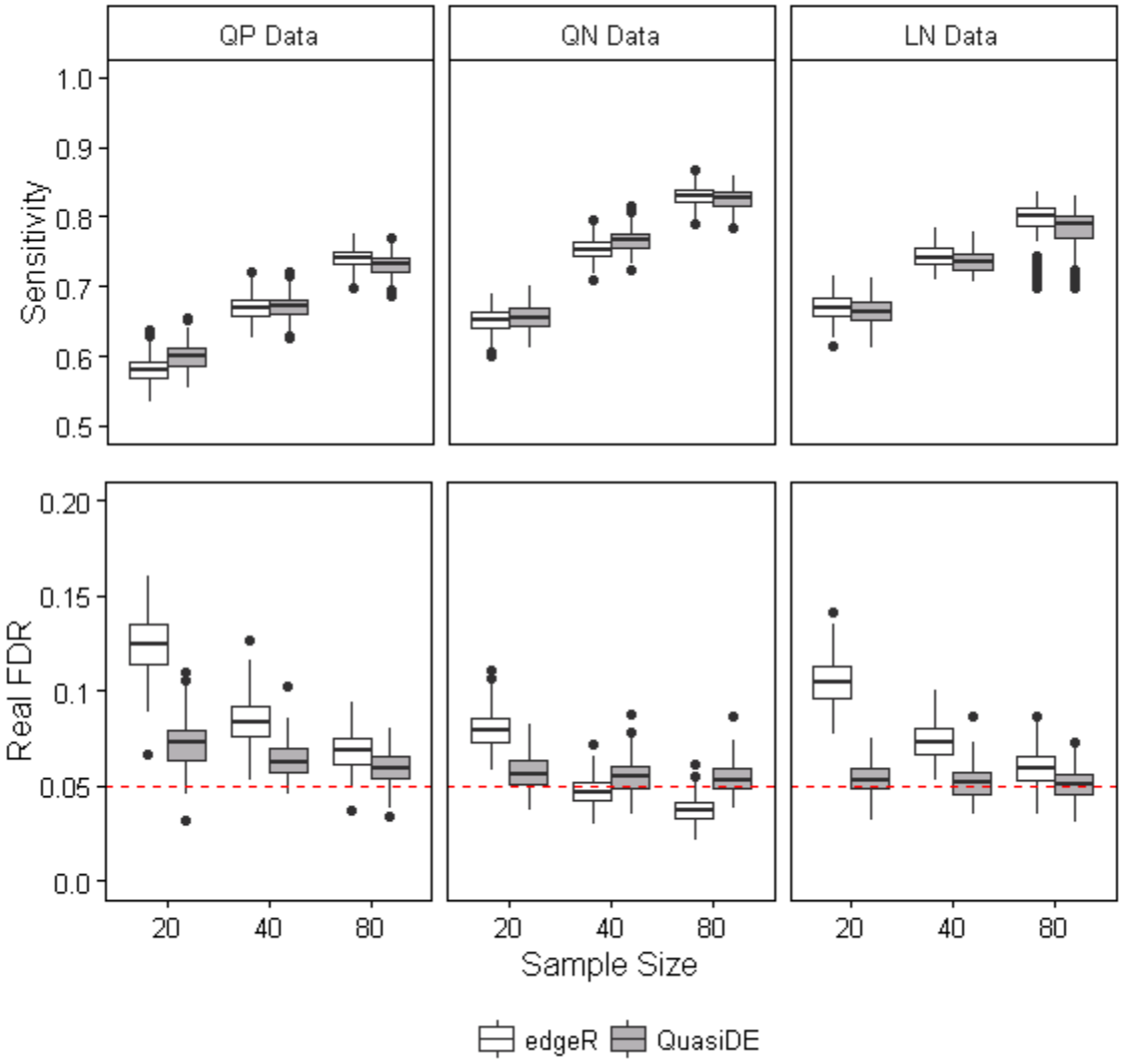

Figure 24. Sensitivity and Real FDR by Sample Size for Different Types of Data (edgeR vs. QuasiDE) in Block Design

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

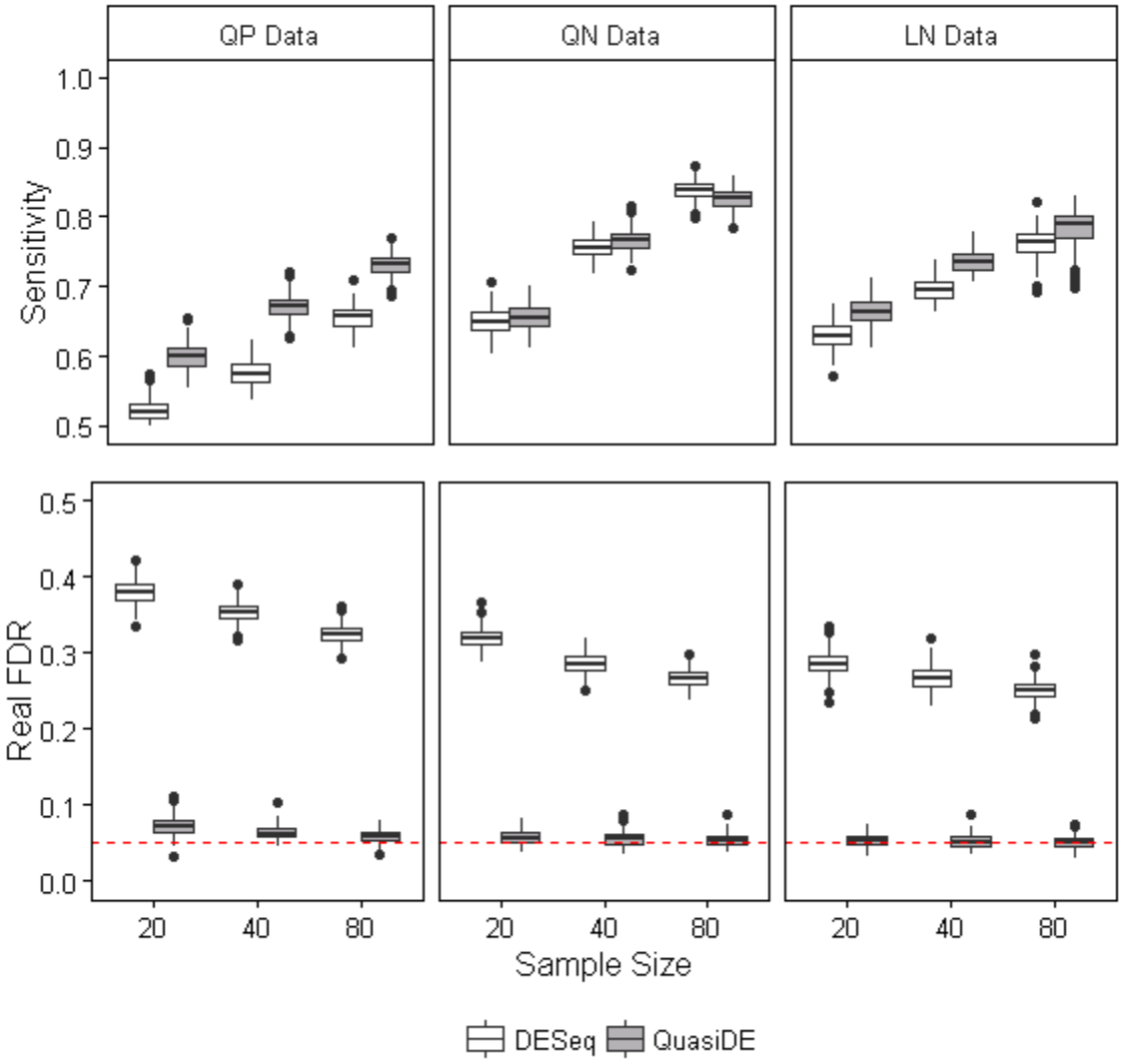

Figure 25. Sensitivity and Real FDR by Sample Size for Different Types of Data (DESeq vs. QuasiDE) in Block Design

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

has a better controlled real FDR. In the rest simulation settings, the real FDR of the QuasiDE method is also competitive to those from the other three approaches. Just as the results in experiments with two conditions, the QuasiDE method has similar sensitivities and FDRs across the data with three types of variance functions.

3.5.3 Factorial Design

The performance of the QuasiDE method is compared with the NBQLSpline method in Figure 26. The sensitivities of the NBQLSpline method are slightly higher than those produced by the QuasiDE method in the QP and LN data. In the QN data, it is slightly complicated. The sensitivities of the NBQLSpline method are lower when the sample size is 20. When the sample size is 40, the sensitivities of the two approaches are about the same. When the sample size is 80, the QuasiDE method has a slightly lower sensitivity. The real FDRs of the QuasiDE method are better controlled in the QP and LN data. In the QN data, the NBQLSpline method appears to have similarly controlled FDR to those from the QuasiDE method. Similarly, as with the simulation in other designs, the NBQLSpline method has the best performance in the QN data since it meets the quadratic variance function assumption.

The performance of the QuasiDE method is compared with the edgeR method in Figure 27. The edgeR method loses sensitivity in QN data with sample sizes 20 and 40. The QuasiDE method has slightly higher sensitivities in almost all the simulation settings except in the LN data with sample size 20. The real FDRs of the QuasiDE method are consistently better controlled than those from the edgeR method in all the simulation settings.

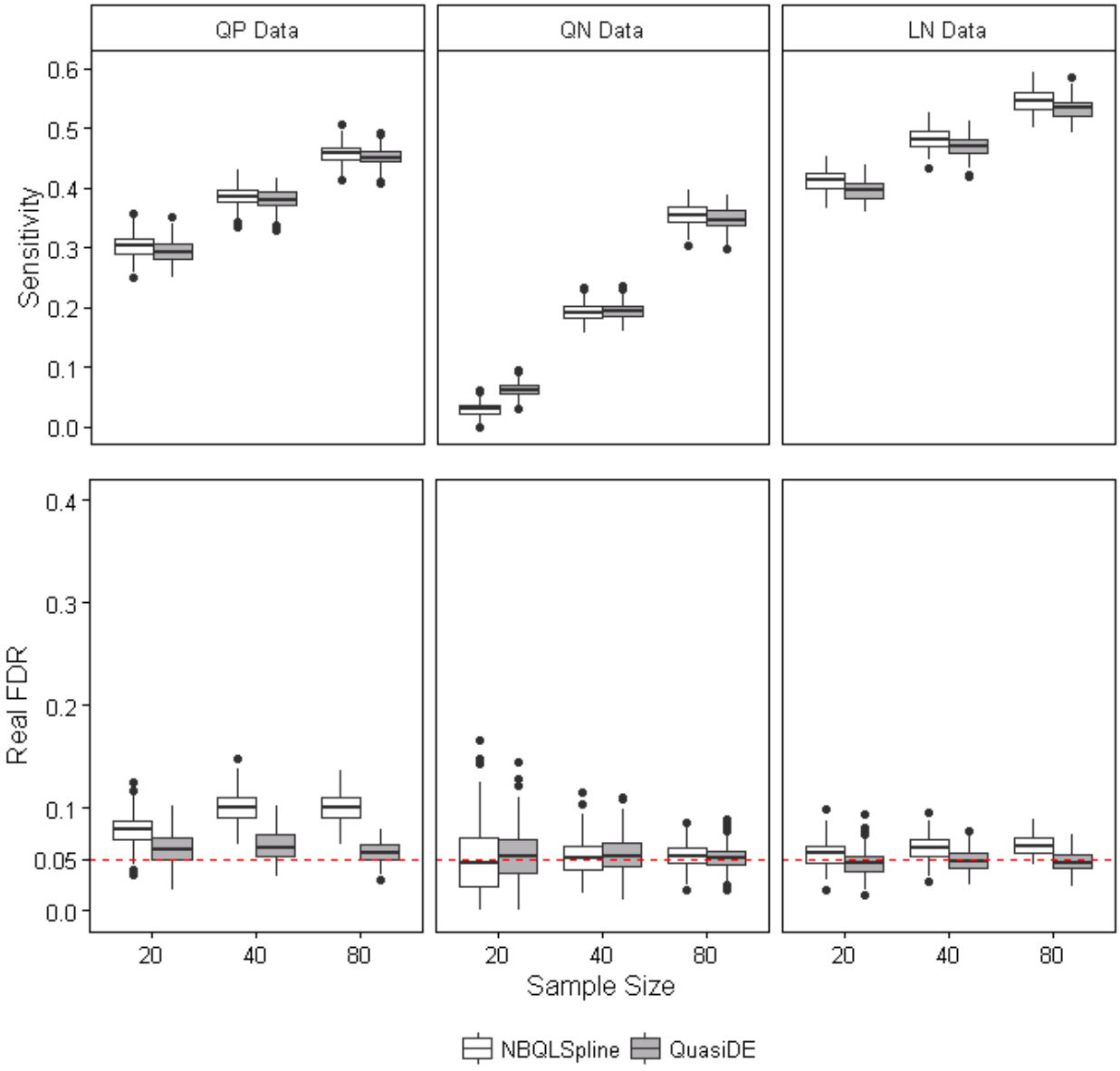

Figure 26. Sensitivity and Real FDR by Sample Size for Different Types of Data (NBQLSpline vs. QuasiDE) in Factorial Design

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

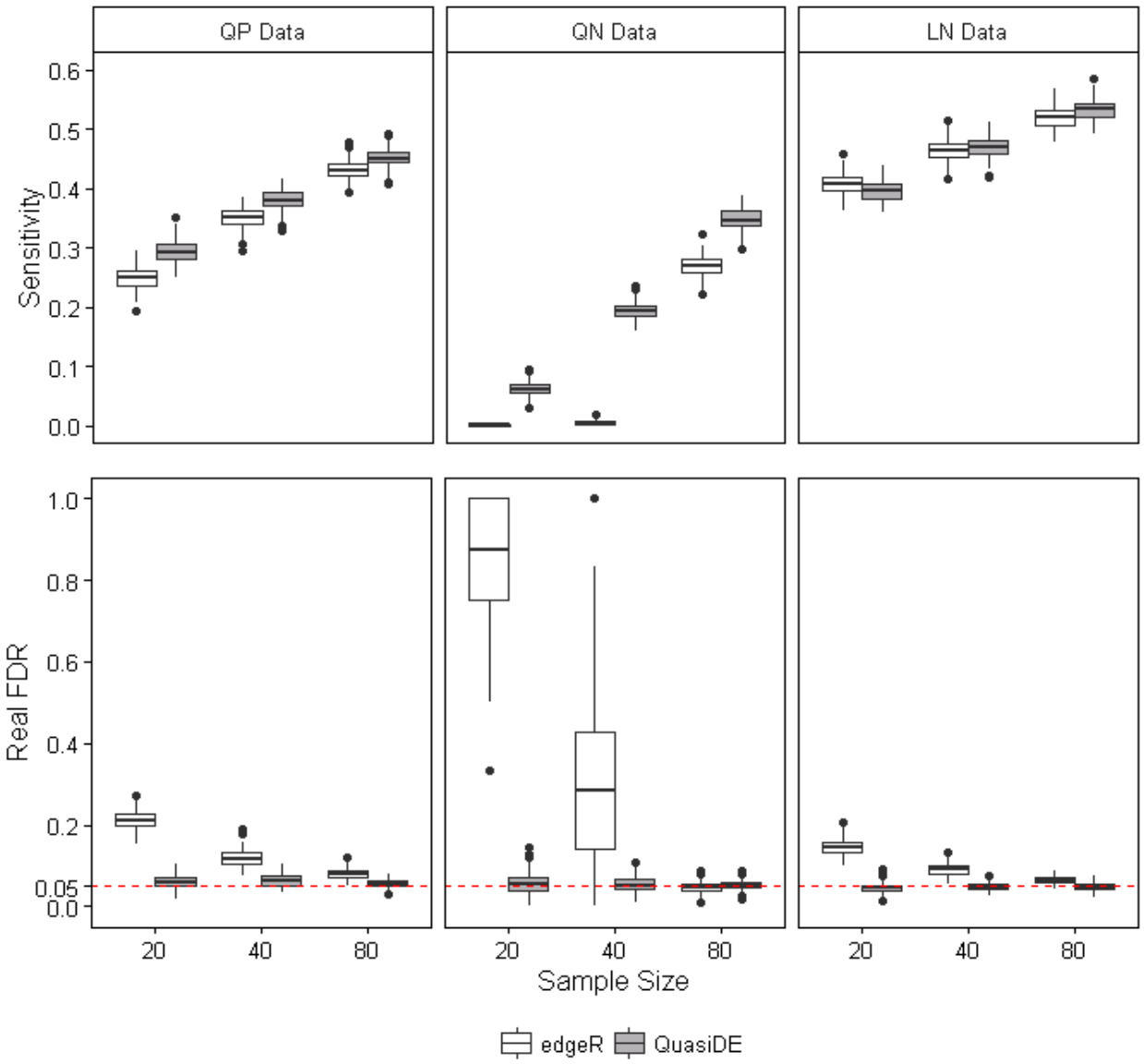

Figure 27. Sensitivity and Real FDR by Sample Size for Different Types of Data (edgeR vs. QuasiDE) in Factorial Design

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

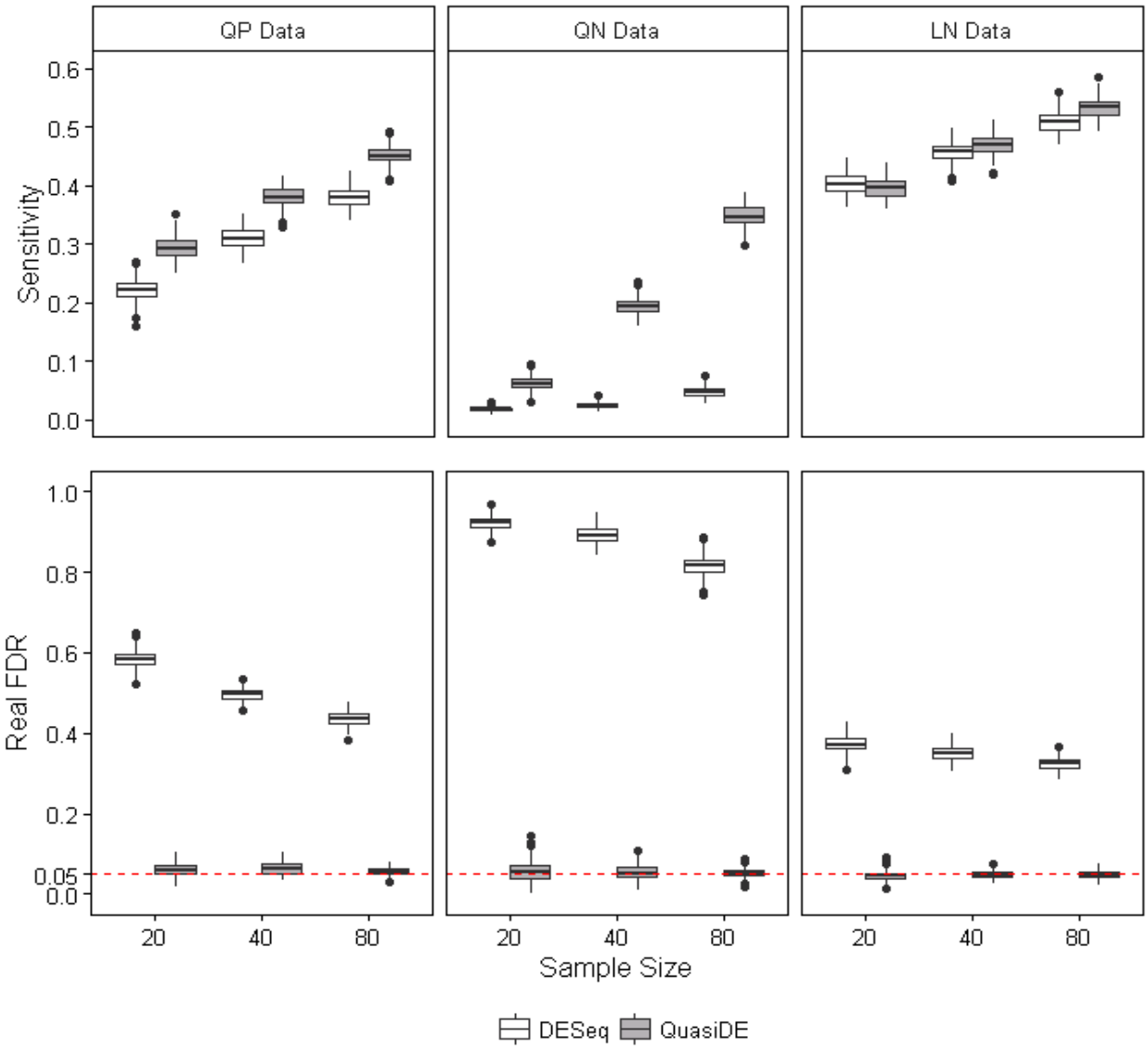

Figure 28. Sensitivity and Real FDR by Sample Size for Different Types of Data (DESeq vs. QuasiDE) in Factorial Design

The three panels in the first row show the sensitivity varying with sample size for the QP, QN and LN data respectively. The three panels in the second row show the real FDR varying with sample size for the QP, QN and LN data respectively. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

The performance of the QuasiDE method is compared with the DESeq method in Figure 28. The DESeq method also loses sensitivity in the QN data with sample size 20 and 40. The QuasiDE method has slightly higher sensitivities in almost all the simulation settings except in the LN data with sample size 20. The DESeq method has very badly controlled FDRs even though they are aimed to be controlled at 0.05.

Overall, the QuasiDE method has similar sensitivities and FDRs across the data with three types of variance functions. The performance is also at least as good as the other three methods.

3.6 Case Studies

There are three case studies in this section. The first case study is an experiment with three conditions. The second is an experiment with treatment and technical effects. The technical effect is treated as block effect. The third case study is in a factorial design and the research interest is to test the interaction between two binary factors. The QuasiDE, NBQLSpline, edgeR, and DESeq methods are used to analyze these three real studies and the results of the DESeq method are not shown since it has very badly controlled FDRs in simulation.

3.6.1 Case Study 1

The Dolatshad data (Dolatshad et al., 2015) are used to perform a real data analysis by comparing the gene expressions of bone marrow CD34+ cells among three groups: eight myelodysplastic syndrome patients with SF3B1 mutation, four myelodysplastic syndrome patients with no known splicing mutation and five healthy control patients. The research interest is to detect those genes differentially expressed at least in one group. The original dataset has 58,023 genes. After filtering, there are 29,375 genes to

be analyzed. There are 27,020 genes filtered due to both zero interquartile range and low mean count (≤ 1). There are 1,002 and 626 additional genes filtered due to zero interquartile range alone and low mean count (≤ 1) alone respectively.

The DE genes detected by the QuasiDE, NBQLSpline, and edgeR methods are compared in Figure 29. The numbers of DE genes detected by the QuasiDE, NBQLSpline, and edgeR methods are 526, 339 and 863 respectively. The top 10 DE genes identified by the QuasiDE method are listed in Table 11 with the corresponding rank and adjusted p-values by the other two methods. In these 10 genes, the adjusted p-values by the NBQLSpline and the edgeR method are smaller than the QuasiDE method. The p-values and the adjusted p-values by all three methods are shown in Figure 30. The number of small p-values (< 0.05) by the QuasiDE method is much

Table 11: Top 10 DE Genes Detected by the QuasiDE Method and their Corresponding Results in the NBQLSpline and edgeR Methods (Dolatshad Data)

Ensembl Gene ID	Gene	QuasiDE		NBQLSpline		edgeR	
		Rank	Adjusted p-value	Rank	Adjusted p-value	Rank	Adjusted p-value
ENSG00000113108	APBB3	1	9.4e-05	1	1.5e-08	50	1.5e-06
ENSG00000092621	PHGDH	2	3.9e-03	2	4.3e-08	7	3.0e-09
ENSG00000238608		3	7.7e-03	157	1.4e-04	523	1.5e-02
ENSG00000231476		4	9.8e-03	3	4.0e-07	172	4.2e-04
ENSG00000232931	LINC00342	5	9.8e-03	29	6.9e-06	228	1.2e-03
ENSG00000143839	REN	6	9.8e-03	60	2.3e-05	28	1.4e-07
ENSG00000006210		7	9.8e-03	72	3.2e-05	47	9.4e-07
ENSG00000224070	HMGN1P6	8	9.8e-03	95	4.9e-05	227	1.2e-03
ENSG00000124942	AHNAK	9	9.8e-03	4	4.3e-07	26	1.3e-07
ENSG00000230176	LINC01433	10	9.8e-03	17	4.1e-06	10	2.6e-08

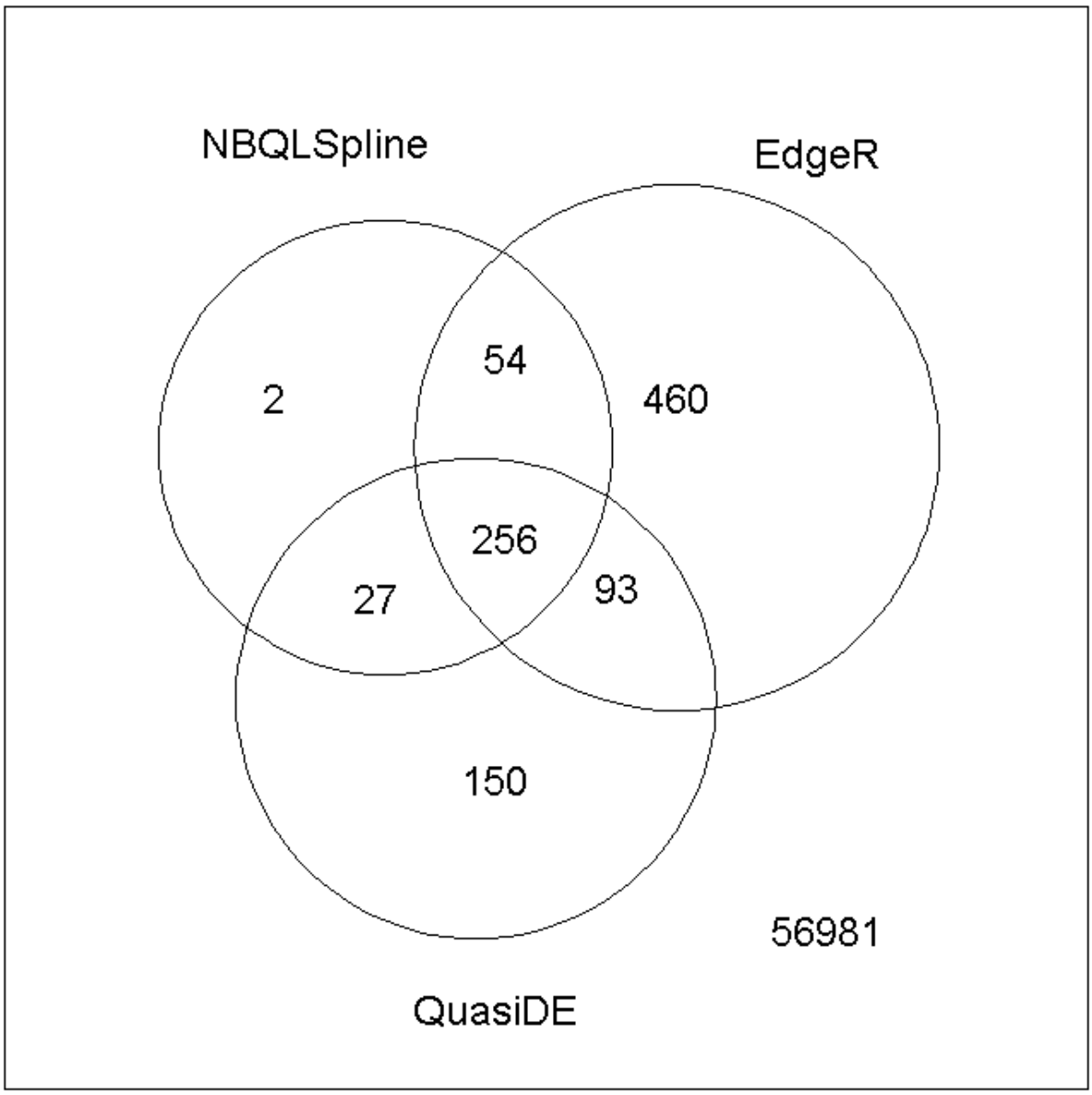

Figure 29. Venn Diagram of DE Gene Counts (Dolatshad Data)

This shows the number of DE Genes detected by the QuasiDE, NBQLSpline and edgeR methods, and their relationship.

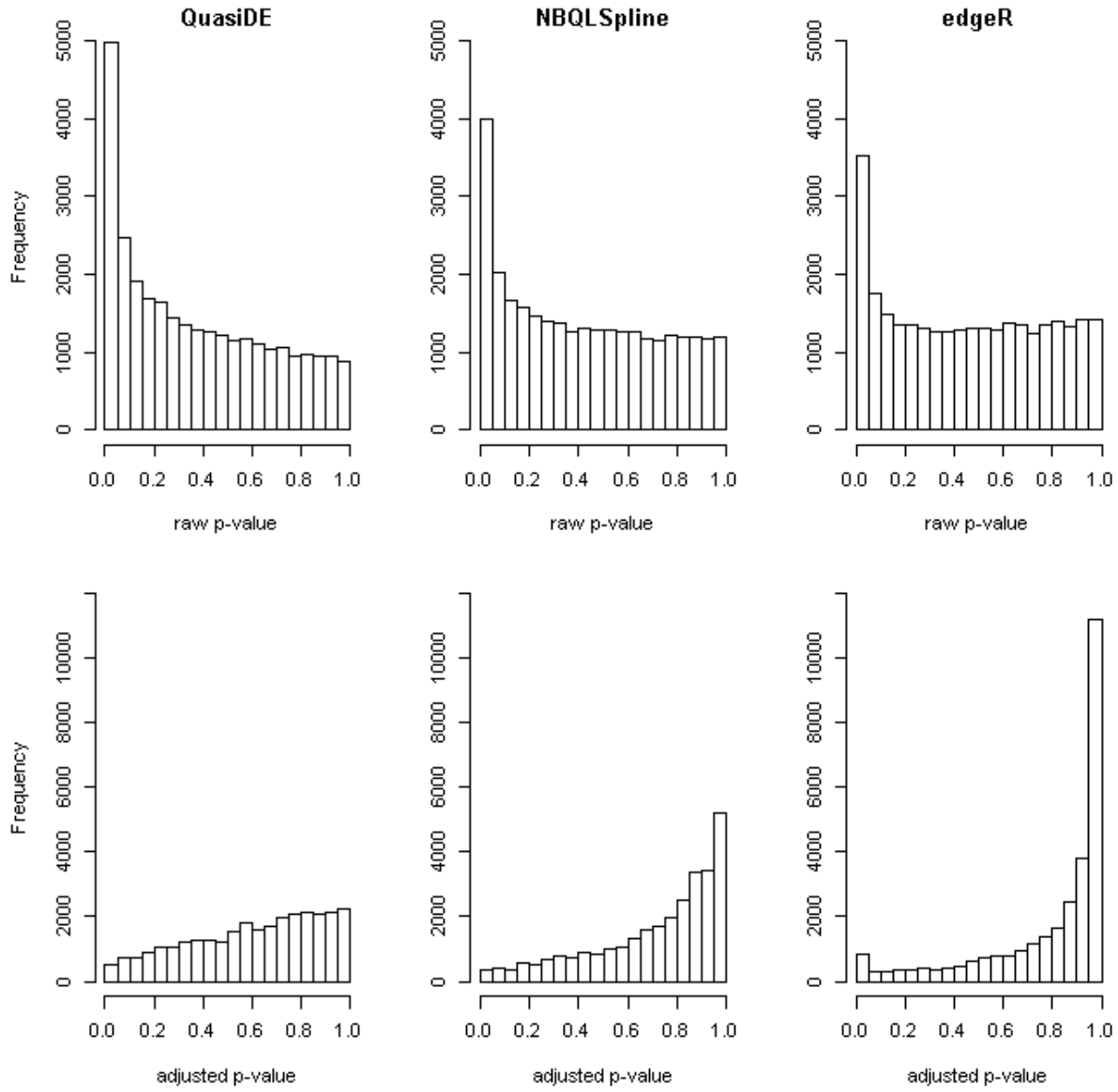

Figure 30. P-values and Adjusted p-values by the QuasiDE, NBQLSpline and edgeR Methods (Dolatshad Data)

The panels in the first row show the distributions of the raw p-values by the QuasiDE, NBQLSpline and edgeR methods. The panels in the second row show the distributions of adjusted p-values by the QuasiDE, NBQLSpline and edgeR methods.

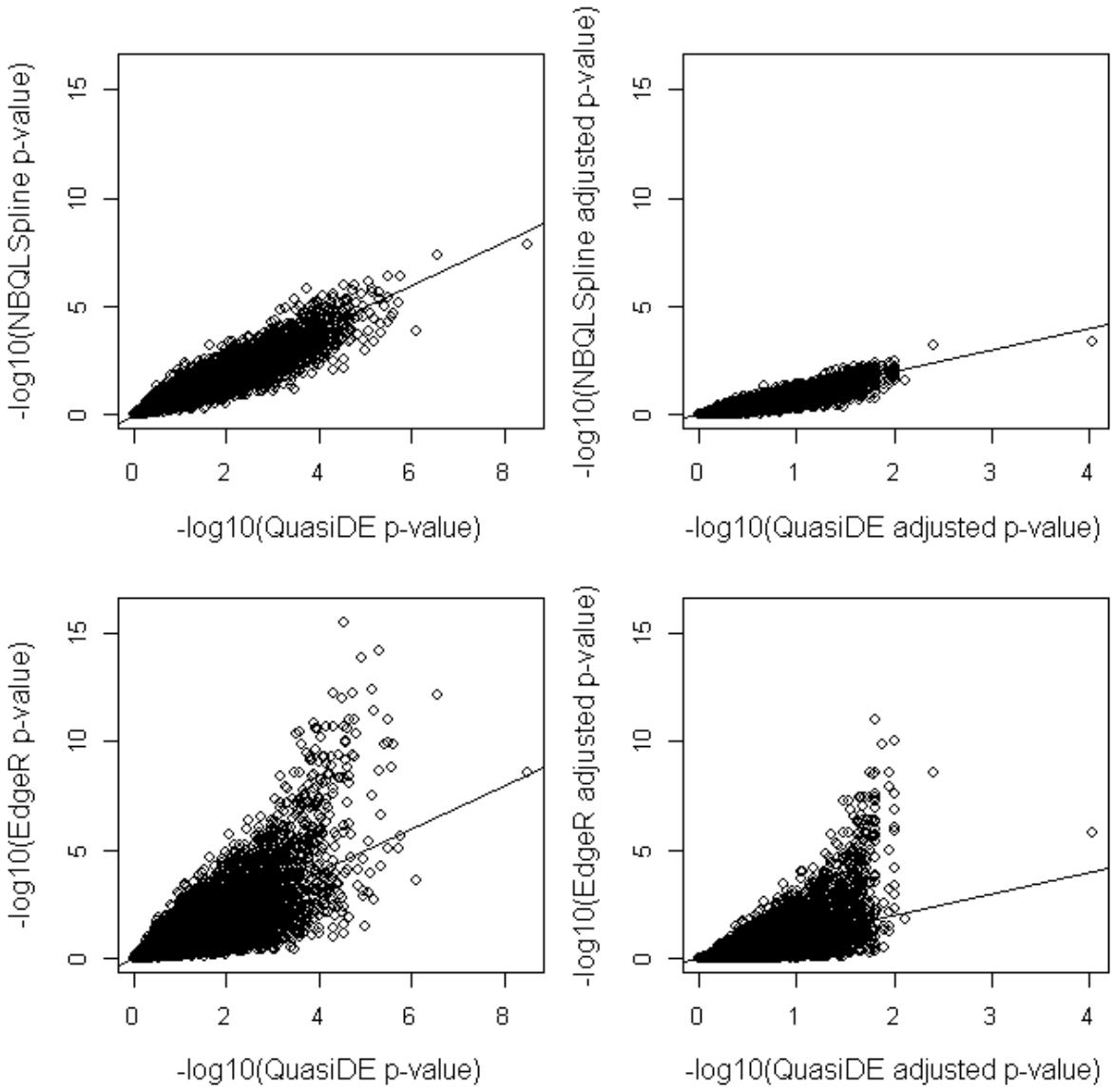

Figure 31. Comparison of the p-values and Adjusted p-values by the QuasiDE, NBQLSpline and edgeR Methods (Dolatshad Data)

The panels in the left column show the p-values by the QuasiDE method vs. p-values by the NBQLSpline and edgeR methods on -log10 scale. The panels in the right column show the adjusted p-values by the QuasiDE method vs. adjusted p-values by the NBQLSpline and edgeR methods on -log10 scale.

Table 12: Reported DE Genes and their Corresponding Results for the QuasiDE, NBQLSpline and edgeR methods (Dolatshad Data)

Gene	QuasiDE		NBQLSpline		edgeR	
	Rank	Adjusted p-value	Rank	Adjusted p-value	Rank	Adjusted p-value
ALAS2	133	2.2e-02	87	1.3e-02	43	6.2e-07
ABCB7	42	1.5e-02	91	1.4e-02	390	6.2e-03

larger than the other two approaches. After multiple comparison correction, the number of small adjusted p-values (<0.05) by the edgeR method is the largest. The small p-values and adjusted p-values are further compared on the -log10 scale between the QuasiDE method and the other two methods in Figure 31. The small p-values (< 0.05) from the QuasiDE method appear to be similar to those from the NBQLSpline method. The edgeR method produced more extreme small p-values. The adjusted p-values have a similar pattern.

In the original paper (Dolatshad et al., 2015), the edgeR method was used to detect DE genes. The DE genes reported such as ALAS2 and ABCB7 are also identified by all these three methods. Their corresponding analysis results for the QuasiDE, NBQLSpline and edgeR methods are shown in Table 12.

3.6.2 Case Study 2

The Bottomly data (Bottomly et al., 2011) are used to perform a real data analysis by comparing the gene expression between two mouse inbred strains. This dataset contains RNA-seq profiling of 10 B6 and 11 D2 inbred mice. There are three separate experiments: two experiments both include three B2 and four D2 inbred mice and the third experiment includes four B2 and three D2 inbred mice. The data have been

Table 13: Top 10 Genes with Treatment Effect Detected by the QuasiDE Method and their Corresponding Results in the NBQLSpline and edgeR Methods (Bottomly Data)

Ensembl Gene ID	QuasiDE		NBQLSpline		edgeR	
	Rank	Adjusted p-value	Rank	Adjusted p-value	Rank	Adjusted p-value
ENSMUSG00000028009	1	8.7e-10	47	1.9e-09	79	1.4e-16
ENSMUSG00000015484	2	8.7e-10	4	5.7e-14	4	4.7e-73
ENSMUSG00000035775	3	9.5e-10	1	5.7e-14	2	2.8e-77
ENSMUSG00000037461	4	9.5e-10	35	2.7e-10	40	7.6e-26
ENSMUSG00000024248	5	3.2e-09	5	1.2e-13	5	4.2e-66
ENSMUSG00000023236	6	4.1e-09	3	5.7e-14	6	6.9e-66
ENSMUSG00000030532	7	5.8e-09	10	4.0e-12	13	2.7e-43
ENSMUSG00000067235	8	5.9e-09	44	1.4e-09	94	1.6e-15
ENSMUSG00000027855	9	6.9e-09	8	3.6e-12	23	2.0e-36
ENSMUSG00000068299	10	8.5e-09	16	6.8e-12	18	1.7e-40

analyzed by taking the experiment number as the block effect. The RNA-seq data in the original study is used mainly to compare the RNA-seq platform with the microarray platform. The original dataset has 36,536 genes. After filtering, there are 10,908 genes to be analyzed. The majority 24,567 genes are filtered by both zero interquartile range and low mean count (≤ 1). There are an additional 1,061 genes filtered by low mean count (≤ 1) alone.

The genes with treatment effect detected by the QuasiDE, NBQLSpline, and edgeR methods are compared in Figure 32. The NBQLSpline, edgeR, and QuasiDE methods have detected 1,781, 1,983 and 1,640 genes with the treatment effect respectively. The top 10 genes with the treatment effect detected by the QuasiDE method are listed in Table 13 with the corresponding rank and adjusted p-values by the other two

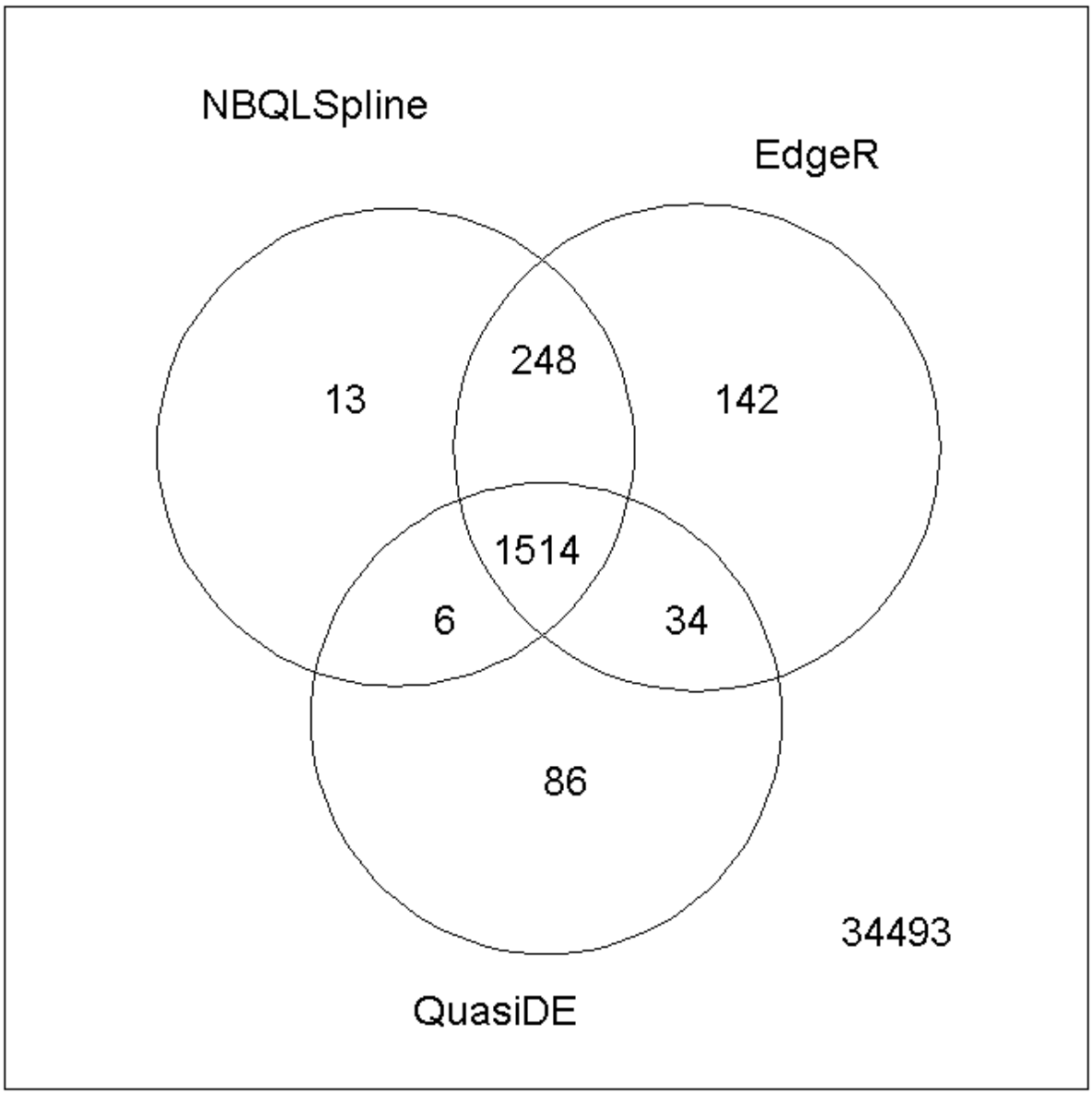

Figure 32. Venn Diagram of DE Gene Counts (Bottomly Data)

This shows the numbers of DE Genes detected by the QuasiDE, NBQLSpline and edgeR methods, and their relationship.

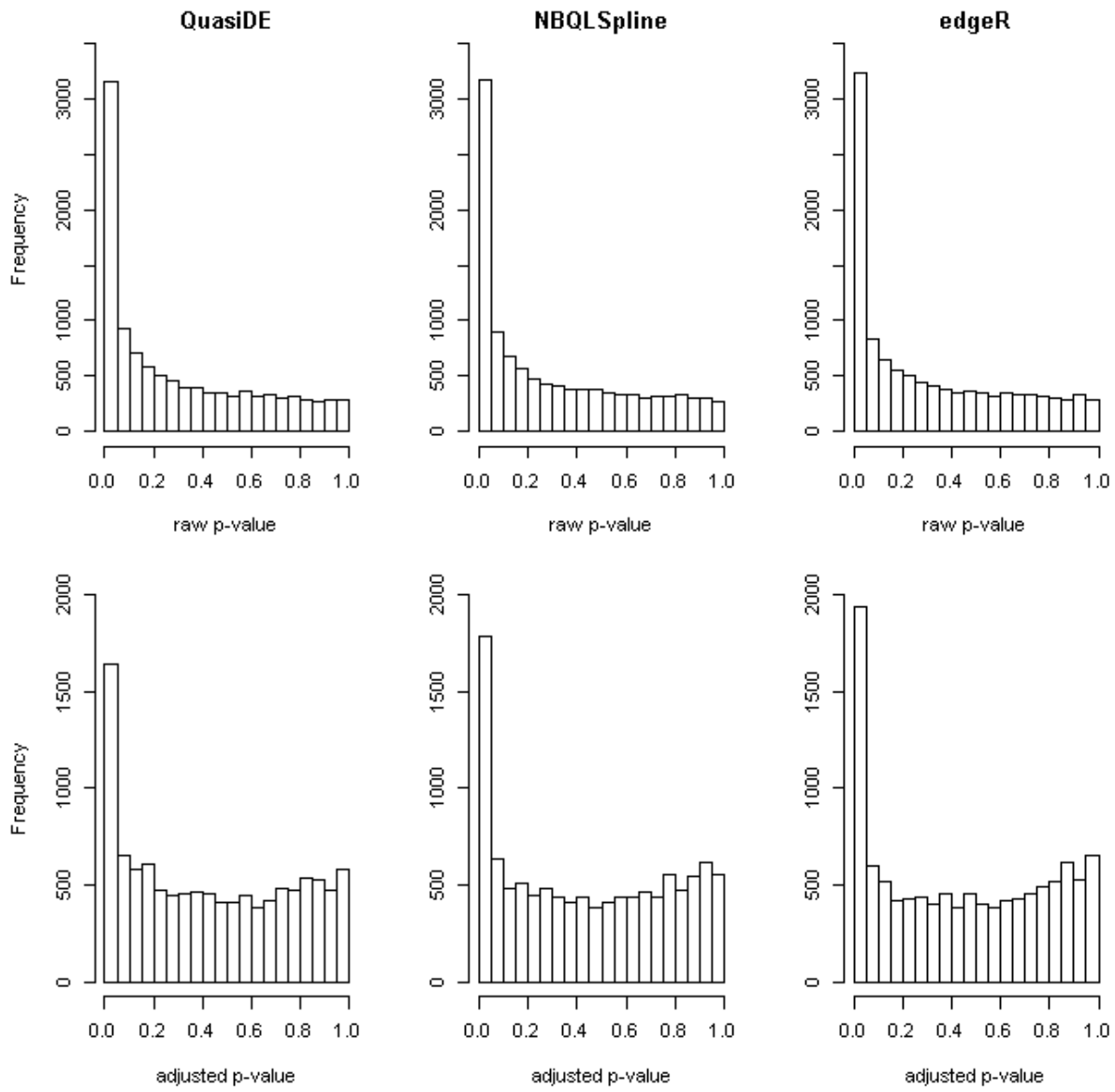

Figure 33. P-values and Adjusted p-values by the QuasiDE, NBQLSpline and edgeR Methods (Bottomly Data)

The panels in the first row show the distributions of the raw p-values by the QuasiDE, NBQLSpline and edgeR methods. The panels in the second row show the distributions of adjusted p-values by the QuasiDE, NBQLSpline and edgeR methods.

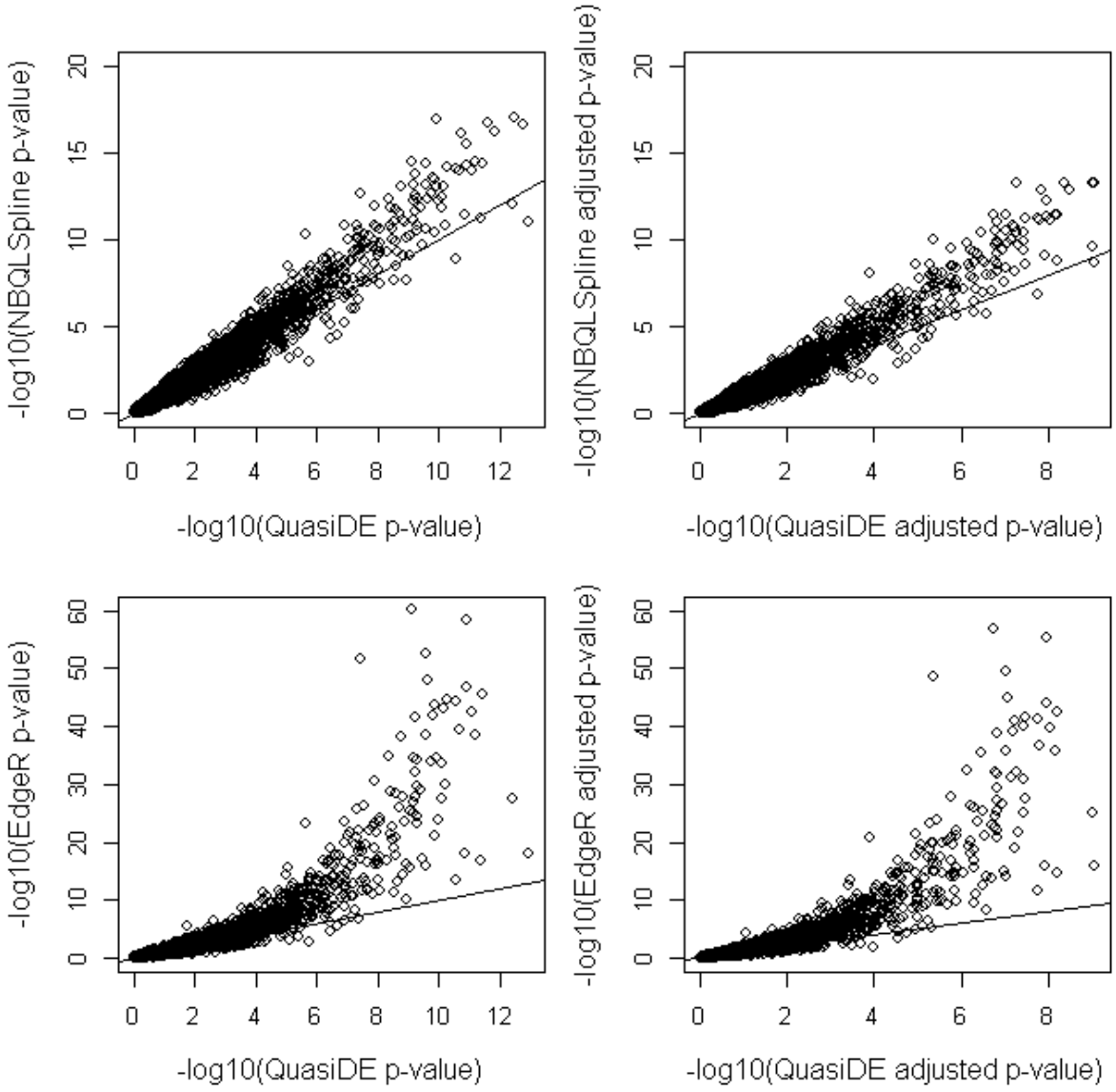

Figure 34. Comparison of the p-values and Adjusted p-values by the QuasiDE, NBQLSpline and edgeR Methods (Bottomly Data)

The panels in the left column show the p-values by the QuasiDE method vs. p-values by the NBQLSpline and edgeR methods on -log10 scale. The panels in the right column show the adjusted p-values by the QuasiDE vs. adjusted p-values by the NBQLSpline and edgeR methods on -log10 scale.

approaches. For these 10 genes, the edgeR method has much smaller adjusted p-values than the other methods. The p-values and the adjusted p-values by all three methods are shown in Figure 33. The numbers of small p-values (< 0.05) by all three methods are similar. After multiple comparison correction, the edgeR method has the highest number of small adjusted p-values (< 0.05). The small p-values and adjusted p-values are further compared on -log10 scale between the QuasiDE method and the other two methods in Figure 34. The small p-values (< 0.05) from the QuasiDE method also appear to be similar to those from the NBQLSpline method. More extreme small p-values are produced by the edgeR method. The adjusted p-values have a similar pattern.

3.6.3 Case Study 3

The Zhang data (Zhang et al., 2015) are used to perform a real data analysis by

Table 14: Top 10 Genes with Interaction Detected by the QuasiDE Method and their Corresponding Results in the NBQLSpline and edgeR Methods (Zhang Data)

Ensembl Gene ID	QuasiDE		NBQLSpline		edgeR	
	Rank	Adjusted p-value	Rank	Adjusted p-value	Rank	Adjusted p-value
ENSRNOG00000001205	1	6.6e-05	5	2.4e-04	29	2.8e-06
ENSRNOG00000007060	2	6.6e-05	1	6.6e-06	3	2.2e-21
ENSRNOG00000025167	3	1.6e-04	13	8.5e-04	79	4.9e-04
ENSRNOG00000056457	4	3.4e-04	3	5.1e-05	4	1.3e-15
ENSRNOG00000039862	5	4.9e-04	184	2.8e-02	800	3.2e-01
ENSRNOG00000011789	6	4.9e-04	7	2.4e-04	20	9.9e-07
ENSRNOG00000019077	7	4.9e-04	26	1.4e-03	120	2.4e-03
ENSRNOG00000019372	8	4.9e-04	29	1.8e-03	174	9.2e-03
ENSRNOG00000009867	9	4.9e-04	23	1.1e-03	85	7.0e-04
ENSRNOG00000005589	10	4.9e-04	12	8.4e-04	63	2.5e-04

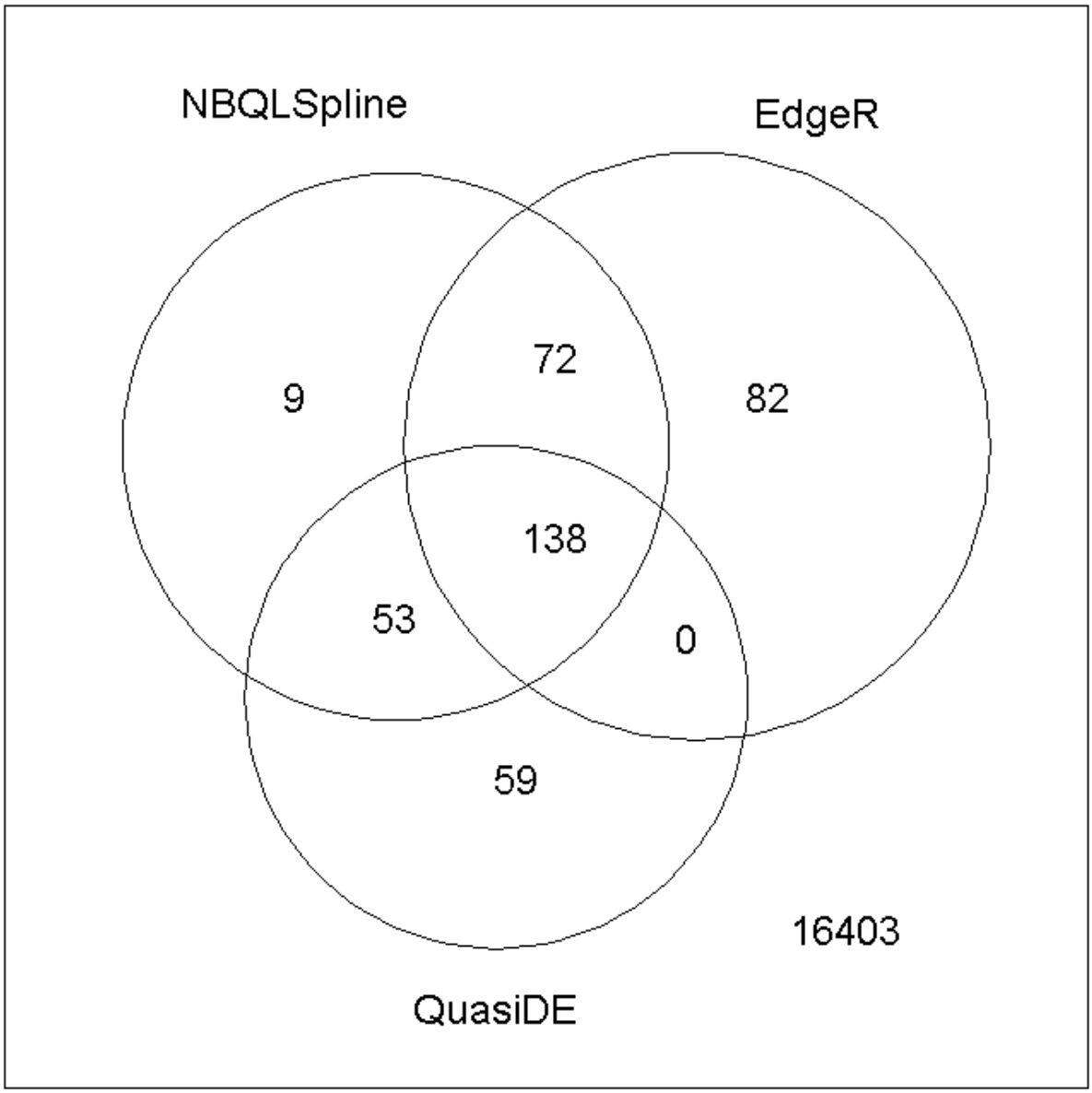

Figure 35. Venn Diagram of Gene Counts with Interaction (Zhang Data)

This shows the numbers of genes with interaction detected by the QuasiDE, NBQLSpline and edgeR methods, and their relationship.

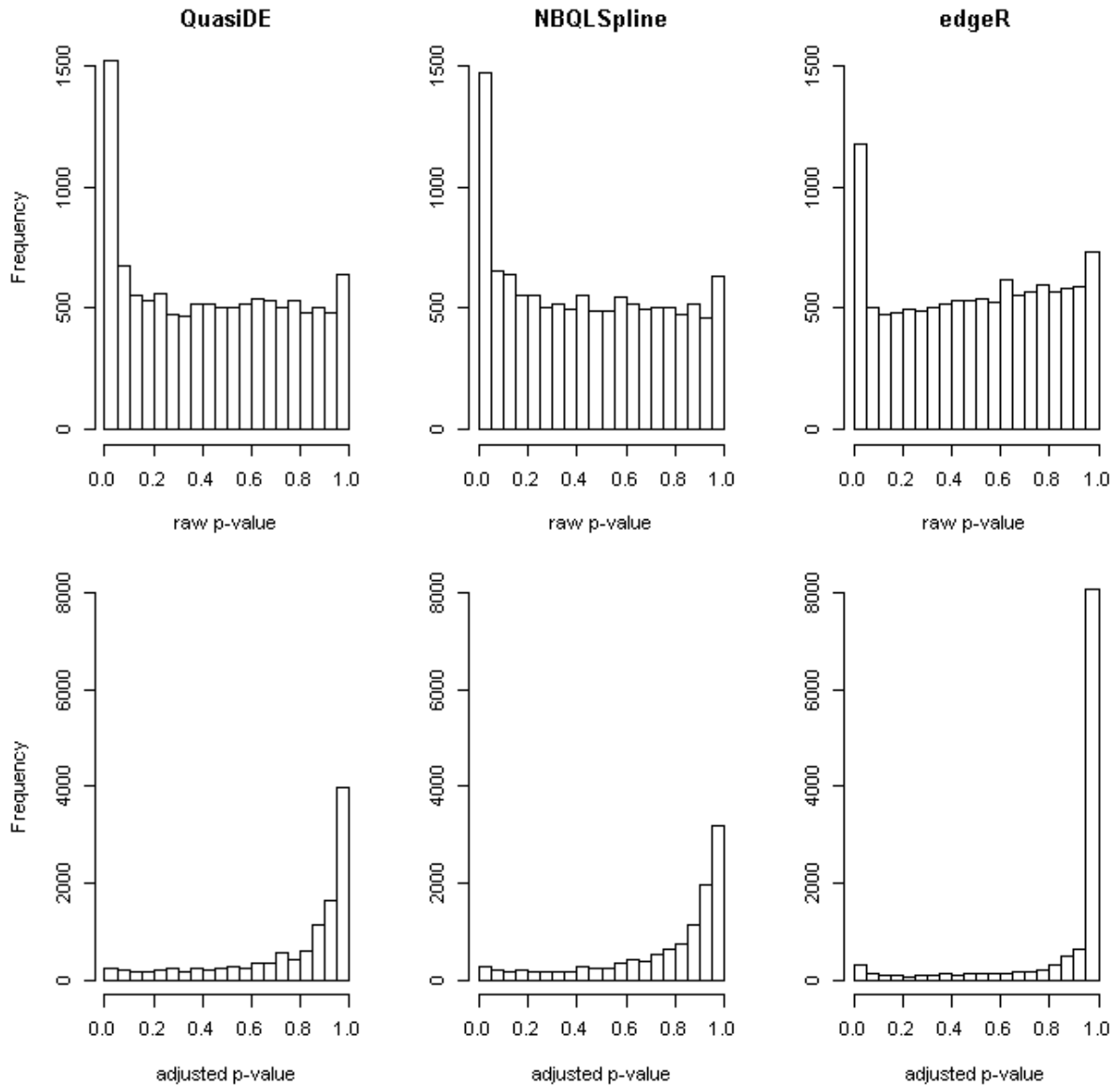

Figure 36. P-values and Adjusted p-values by the QuasiDE, NBQLSpline and edgeR Methods (Zhang Data)

The panels in the first row show the distributions of the raw p-values by the QuasiDE, NBQLSpline and edgeR methods. The panels in the second row show the distributions of adjusted p-values by the QuasiDE, NBQLSpline and edgeR methods.

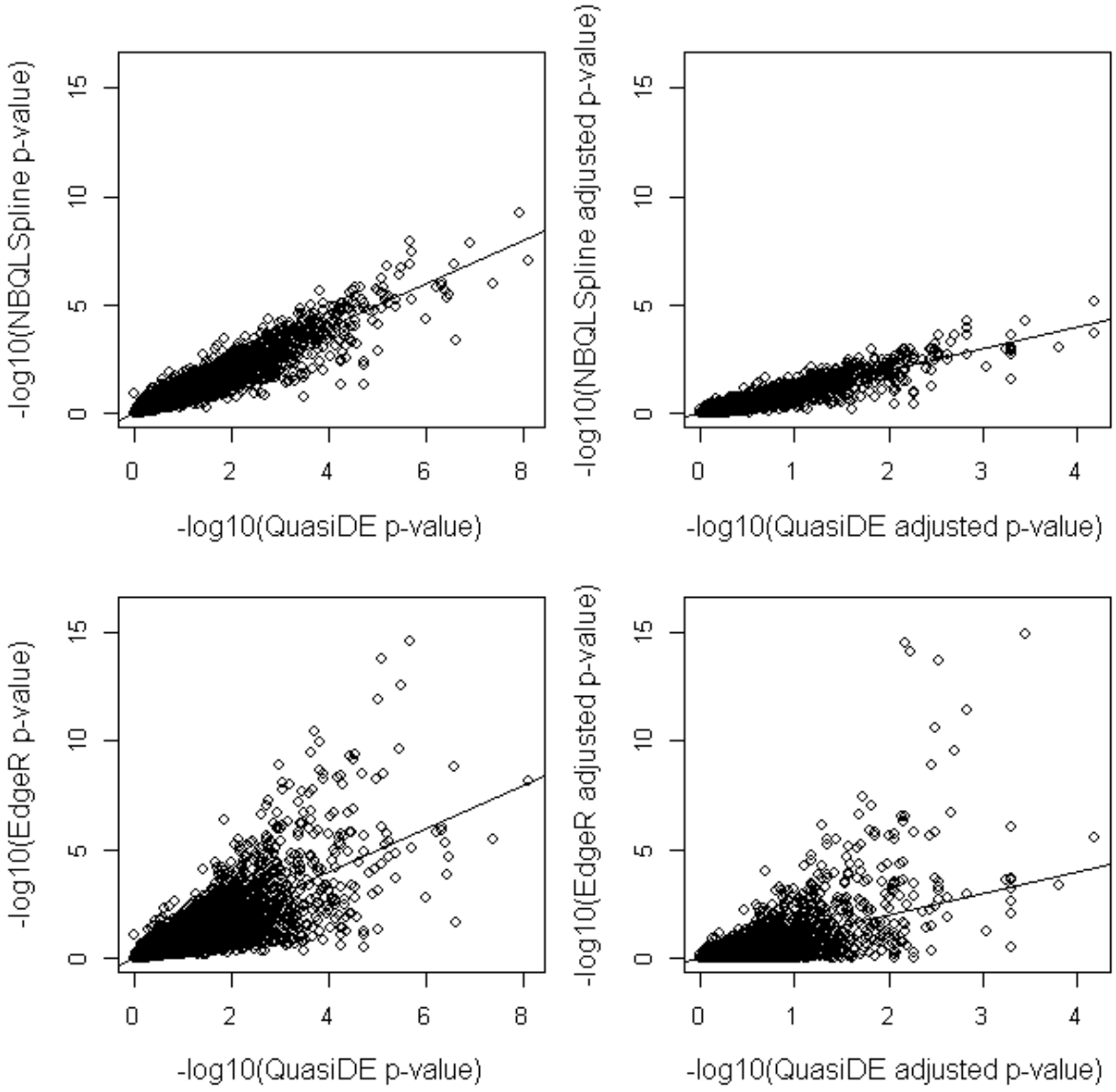

Figure 37. Comparison of the p-values and Adjusted p-values by the QuasiDE, NBQLSpline and edgeR Methods (Zhang Data)

The panels in the left column show the p-values by the QuasiDE method vs. p-values by the NBQLSpline and edgeR methods on -log10 scale. The panels in the right column show the adjusted p-values by the QuasiDE method vs. adjusted p-values by the NBQLSpline and edgeR methods on -log10 scale.

comparing the gene expression between two different cells treated with or without pioglitazone. The oncogenic PAX8-PPARG fusion protein (PPFP) in thyroid carcinomas is caused by a chromosomal translocation. In the study, PPFP was expressed in PCCL3 rat thyroid cells. The PPFP-dependent gene expression was studied. The comparison is between six PPFP cells and six empty cells. Half of the six PPFP cells and half of the six empty cells are treated with pioglitazone. The research interest is to detect those genes which have the interaction between cell type and treatment. The original dataset has 16,816 genes. After filtering, there are 11,586 genes to be analyzed. There are 4,333 genes filtered due to both zero interquartile range and low mean count (≤ 1). In addition, there are 897 genes filtered by low mean count (≤ 1) alone. The genes with interaction detected by the QuasiDE, NBQLSpline, and edgeR methods are compared in Figure 35. The edgeR, NBQLSpline, and QuasiDE methods have detected 292, 272, and 250 genes with interaction respectively. The top 10 DE genes with interaction identified by the QuasiDE method are listed in Table 14 with the corresponding rank and adjusted p-values by the other two approaches. Of these 10 genes, the fifth gene has been detected as a gene with interaction by the QuasiDE and NBQLSpline methods but not by the edgeR method. The p-values and adjusted p-values by all the three methods are shown in Figure 36. The number of small p-values (< 0.05) by the QuasiDE and NBQLSpline methods are higher than the edgeR method. After multiple-comparison correction, the numbers of small adjusted p-values (< 0.05) of all three methods are similar. The small p-values and small adjusted p-values (< 0.05) are further compared on -log10 scale between the QuasiDE method and the other two methods in Figure 37. The p-values of the QuasiDE method appear to be similar to

those from the NBQLSpline method. The edgeR method produced more extreme small p-values. The adjusted p-values have a similar pattern.

The analysis in the original paper (Zhang et al., 2015) focused on pair-wise comparisons between subgroups. There are no DE genes found between the empty cells treated with or without pioglitazone. There are 250 DE genes detected between PPFP cells treated with and without pioglitazone.

3.7 Discussion

In this chapter, the QuasiDE method is extended to three types of designs: design with multiple experimental conditions, block design and factorial design. The QuasiDE method can also potentially adapt to other complex designs. There are numerous other complex designs in the literature such as matched pairs design and time course design. In matched pairs design, the differences in expression between pairs are typically analyzed to test if the difference is zero. The number of groups in this case reduces to one group. The variance function can be estimated by the mean-variance pairs of these gene expression differences. The quasi-likelihood F statistic can be constructed accordingly. In the time course design, the time covariate can be considered as a continuous covariate by the QuasiDE method. Using the QuasiDE method in a complex design, the linear predictor is $\log(\boldsymbol{\mu_g}) = \beta_{1g} + \beta_{2g}X_1 + \cdots + \beta_{Mg}X_M$, where $X_1, X_2, \ldots, X_M$ are the covariates. The covariates can be either categorical or continuous. There is no problem to include a continuous covariate in the QuasiDE method. However, the way to estimate the variance function adjusting for a continuous covariate may need careful consideration. The options might be to develop an estimation procedure to adjust for a continuous covariate. The generalized additive model can be considered. The variance

function can possibly be estimated using smooth cubic spline adjusting for a continuous covariate. If the time can be modeled as a factor, the impact from time in this situation can be easily taken into account in estimating variance function, in which the mean-variance pairs in all the subgroups can be used.

With numerous possible designs, the current extension of the QuasiDE method to three common designs is just a start.

3.8 Conclusion

In this chapter, the QuasiDE method is adapted to some common study designs: design with multiple experimental conditions, block design and factorial design. In simulation, the same advantages have been found as in experiments with two conditions: the sensitivity is about the same as other popular methods while the real FDR is better controlled in most situations. The sensitivities and FDRs are more similar across different types of data with different variance functions than those from the other methods. Also, it is promising to be applied in some other complex designs.

CHAPTER 4. Statistical Analysis of the RPKM Data

4.1 Background

The RNA-seq read count is associated with two important factors: the gene length and the library size. It is obvious that more reads will be produced from longer transcripts. To adjust for the gene length, a simple gene-wise normalization is to divide the number of reads by the gene length and then multiply it by a constant 10^3. On the other hand, if a larger number of reads are obtained in the experiment for a specific biological sample (library size), the number of reads for each gene will be proportionally larger. To adjust for this factor, a simple sample-wise normalization is to divide the number of reads by the associated library size and then multiply it by a constant 10^6. Reads per kilobase per million mapped reads (RPKM) combines the above two adjustments and facilitates comparisons between samples (Dillies et al., 2012). The RPKM of the g^{th}gene for the j^{th} sample in the i^{th} experimental condition is defined as

$$RPKM_{ijg} = y_{ijg}\frac{10^3}{\lambda_{ijg}}\frac{10^6}{L_{ij}} = \frac{y_{ijg}}{\lambda_{ijg}}\frac{10^9}{L_{ij}} \tag{35}$$

where λ_{ijg} denotes the gene length of the g^{th} gene for the j^{th} sample in the i^{th} experimental condition. Wagner et al. (2012) found that RPKM is not a consistent measure of mRNA abundance. With this finding, researchers focused more on developing a statistical method using the raw read counts. These include two methods implemented in R packages edgeR (Robinson et al., 2012) and DESeq (Anders et al., 2010) and a quasi-likelihood approach implemented in R package QuasiSeq (Lund et al., 2012). The first two methods assume a NB distribution and the QuasiSeq method assumes a linear or quadratic variance function. When the comparison is between samples in different experimental conditions and the gene length is fixed, the effect of

gene length will usually cancel out in statistical analysis. However, there are cases where researchers are interested in making comparisons between similar genes in different species (Reid et al., 2012). In this situation, the gene length is different for most orthologs. Even when the comparison is among biological samples of the same species, the gene length may be varied for the same gene among these samples due to mutation, insertion or other mechanisms. In this case, the gene length may be different to a low or moderate extent. When the gene length is different among samples, normalization using the gene length becomes necessary. The RPKM value is able to correct the impact from both the library size and the gene length and facilitates the comparisons in this situation. In addition, the RPKM value is a popular choice in practical applications (Mortazavi et al., 2008) especially for biologists. The proper methods to conduct the DE analysis based on the RPKM data remain unclear in the literature.

An example of RPKM data from Reid et al. (2012) is shown in Table 15. In this dataset, the samples are from two different, but closely related, species of parasites: Toxoplasma gondii and Neospora caninum. There are eight independent Toxoplasma gondii samples and six independent Neospora caninum samples. In cell culture, the parasites were grown asynchronously for a period of six days. RNAs were taken from each two Toxoplasma gondii samples at day 2, 3, 4 and 6. RNAs were taken from each two samples of Neospora caninum at day 3, 4 and 6. These RNA samples were sequenced on an Illumina GAIIx machine. The research interest is to detect those orthologs which have an interaction effect between species and time.

Table 15: Example of an RNA-seq RPKM Dataset

(Reid et al., 2012)

Day	Species	Sample No	Orthologs		
			NCLIV_000010* TGME49_092920**	NCLIV_000020 TGME49_092930	… …
2	**Toxoplasma gondii**	1	59.8	0.4	…
		2	37.1	0.1	…
3	**Neospora caninum**	1	30.9	0	…
		2	22.7	0	…
	Toxoplasma gondii	1	52.9	0.4	…
		2	45.8	0.3	…
4	**Neospora caninum**	1	28.1	0	…
		2	22.3	0.1	…
	Toxoplasma gondii	1	24.9	1.3	…
		2	24.9	0.5	…
6	**Neospora caninum**	1	33.8	0	…
		2	25.1	0	…
	Toxoplasma gondii	1	37.9	0.3	…
		2	32.7	0.1	…

***** *Neospora caninum gene ID*

****** *Toxoplasma gondii gene ID*

The distributions of the raw count data and RPKM data on log scale are compared in Figure 38 using one sample of Neospora caninum and one sample of Toxoplasma gondii at day 3. The percent of zero values for these two samples are all lower than 2%. The ranges of the raw count data and the RPKM data are quite different.

Theoretically, RPKM can also be thought of as a normalization method. Preprocessing RPKM data may include filtering and further normalization. The filtering method under consideration is to remove those genes with zero interquartile range in the RPKM data

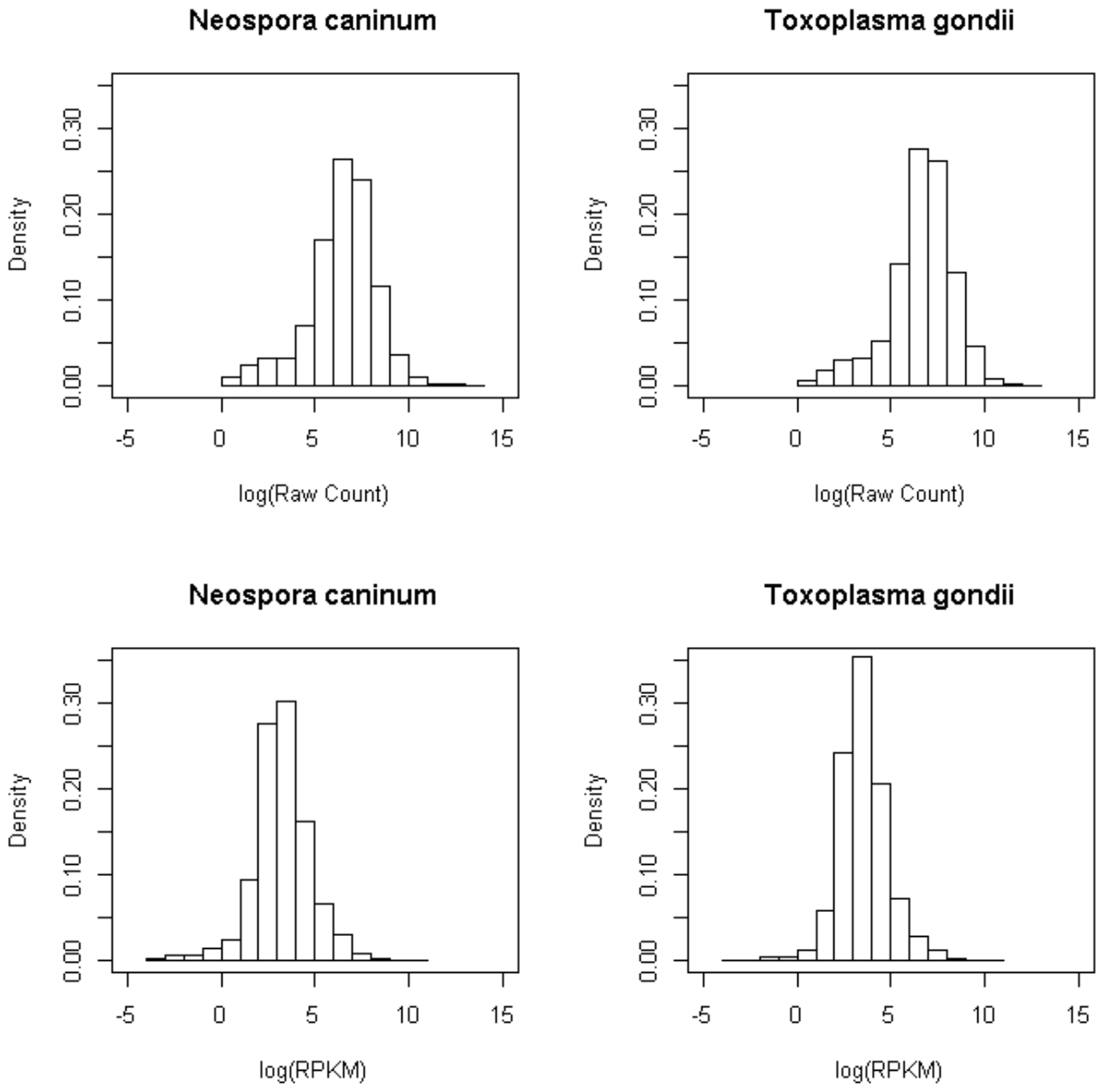

Figure 38. Histogram of Raw Count Data and RPKM Values on log scale (One sample in each of two Species on Day 3, Reid Data)

or with low mean RPKM value (≤ 1). Since the range of RPKM values is quite different from the range of the raw count data, the filtering method is chosen to only remove those genes with zero interquartile range in the RPKM data. For further normalization, the RPKM data are either used directly without further normalization or are normalized by the existing methods not originally developed for the RPKM data. In the next section, the normalization methods applicable to the RPKM data are described as well as the applicable statistical methods. The comparison among these normalization and statistical methods is made in a simulation. In this chapter, a completely randomized design with two experimental conditions is assumed.

4.2 Normalization Methods

Many normalization methods have been proposed in the literature for the RNA-seq data and microarray intensity data. For the RPKM data, at least seven methods were found applicable to the RPKM data regardless of whether they are designed for the RNA-seq data or for microarray intensity data. These normalization methods are introduced below:

4.2.1 TMM

In this method, normalization factors are calculated as weighted Trimmed Mean of M-values. This method is described in Section 2.2.

4.2.2 RLE

In this method (Anders et al., 2010), a reference sample is calculated as the geometric mean of each gene across all samples. Normalization factors are the median of the ratios between the sample to be normalized and the reference sample. Since the genes

with zero count will result in zero geometric mean for those genes in the reference sample, the ratios between the sample to be normalized and the reference sample for those genes become not evaluable. To calculate the normalization factors, genes with zero counts are not used in the calculation. This method is referred to as Relative Log Expression (RLE).

4.2.3 UQ

In this method (Bullard et al., 2010), the raw count data are scaled by their associated library size first. Normalization factors are the upper quartiles of the scaled data for each sample. This normalization method is referred to as Upper Quartile (UQ).

4.2.4 Scaling

In this method (Affymetrix, 2001), all samples are scaled to have the same median. The median of all genes for each sample is calculated first. The geometric mean of these sample-specific medians is used as common median. The scale factor in each sample is calculated as the sample-specific median divided by the common median. The genes in each sample are then divided by this sample-specific scale factor.

4.2.5 Quantile

The quantile normalization (Bolstad et al., 2003) imposes the same empirical distribution to each sample. Within each sample, the raw data are sorted first. The empirical distribution is obtained by the arithmetical means of the sorted values with the same rank across all samples. To impose this distribution to each sample, the raw data of each sample are replaced by the values in the empirical distribution with the same rank.

4.2.6 Cyclic Loess

In each cycle, the loess normalization (Yang et al., 2002) is applied to all possible pairs of samples. The raw values of two samples are first log transformed. The M-values and A-values of the raw values are calculated. A loess curve is then fitted between M-values and A-values. The differences between the fitted M-values and raw M-values ΔM_g are calculated. The log transformed raw values for the first sample are adjusted by adding $\Delta M_g/N$. The log transformed raw values for the second sample are adjusted by reducing $\Delta M_g/N$. The adjusted values are reverted to the original scale. The procedure iterates until convergence or the maximum number of iterations is reached. The convergence is measured by how much additional adjustment on that iteration.

4.2.7 Invariant Set

This method (Li and Wong, 2001) also works in a pairwise manner. In a pair of samples, one sample is used as reference. The raw values in each sample are ranked. Those genes with similar rank across the two samples are selected. A cubic smooth spline is fitted using the raw values of these selected genes between the sample to be normalized and the reference sample. The raw values of the sample to be normalized are replaced by the fitted values of the estimated spline.

The TMM, RLE and UQ methods are devised for the RNA-seq count data and applicable to the RPKM data. They are implemented in R package edgeR. To apply these methods to the RPKM data, the RPKM data are divided by their normalization factors. The scaling, quantile, cyclic loess, and invariant set methods were initially devised for microarray intensity data and are also applicable to the RPKM data. They are implemented in R package limma.

4.3 Statistical Methods

Two methods are found applicable to analyze the continuous RPKM data directly. They are the LIMMA (Smyth, 2004) method and the QuasiDE method. It is also of interest to understand the difference in analyzing the RPKM data and analyzing the raw count data with different normalization methods other than RPKM if RPKM is considered as a normalization method. When the gene length is the same across samples, the gene length effect will cancel out by using these two statistical methods. One of the proper preprocessing methods on the raw count data uses the filtering method and the normalization method in Chapter 2. The raw count data are filtered by zero interquartile range or low mean count (≤ 1). The data after filtering is normalized by the TMM method. When the gene length is not fixed across samples, the proper preprocessing method on raw count data has not been studied. In this setting, two approaches are proposed to do the preprocessing. The first approach is to do the same preprocessing as if the gene length is fixed across samples. Using this preprocessing, the consequence of ignoring the gene length can be studied. In the second approach, the raw count data will be filtered by zero interquartile range or low mean count (≤ 1). After filtering, the raw count data is firstly corrected by the gene length (the raw count data is divided by the associated gene length and multiplied by a constant 10^3). The data is then normalized by the TMM method. The performance of the second approach is studied in simulation later in this chapter.

The analysis of the raw count data with the normalization other than RPKM is only possible when the raw count data is available. This is used to compare with the analysis of the RPKM data, which assumes only the RPKM data available. The edgeR method is

not able to analyze the continuous RPKM data but is able to analyze the raw count data adjusting for the impact from the library size and the gene length in a GLM model. It is also included for comparison purpose.

4.3.1 LIMMA

The method was originally developed in Smyth (2004) for the continuous intensity data in microarray experiments but is also applicable to the RPKM data. For the RPKM data in two experimental conditions, the ordinary t statistic is

$$t = \frac{\overline{RPKM}_{1g} - \overline{RPKM}_{2g}}{s_{g,RPKM}/\sqrt{N}} \tag{36}$$

where $\overline{RPKM}_{1g}$and $\overline{RPKM}_{2g}$ are the mean RPKM values of the g^{th} gene in experimental condition 1 and 2, and $s_{g,RPKM}$ is the pooled standard deviation for the g^{th} gene. In the case of small sample size, the estimate of $s_{g,RPKM}$ may not be reliable. To come up with a more reliable estimate, $s_{g,RPKM}$ is moderated by the empirical Bayes approach which borrows information from the other genes.

The statistic proposed in the LIMMA method is the moderated t statistic:

$$Moderated\ t = \frac{\overline{RPKM}_{1g} - \overline{RPKM}_{2g}}{\tilde{s}_{g,RPKM}/\sqrt{N}} \tag{37}$$

where $\tilde{s}_{g,RPKM}$ is the gene-specific posterior standard deviation, in which $\tilde{s}^2_{g,RPKM}$ is a weighted average of $s^2_{g,RPKM}$ and a global prior variance estimate $s^2_{0,RPKM}$. The raw variances of each gene are shrunken to the global prior estimate. The assumptions in the LIMMA method are that the data follow the normal distribution and the moderated t statistics follows the t distribution with the degrees of freedom estimated from the data. To make the LIMMA method applicable to the RPKM data, the RPKM data are assumed

to have a normal distribution. To adjust for multiple comparisons, the Benjamini and Hochberg procedure is used in the LIMMA method (Benjamini and Hochberg, 1995). The performance will be evaluated in simulation later in this chapter. In simulation, the LIMMA method is used as a method to analyze the RPKM data. In addition, the LIMMA method will also be used to compare the analysis of the RPKM data and the analysis of the raw count data with normalization other than RPKM.

4.3.2 QuasiDE

This approach was introduced in Chapter 2. It was initially designed for the normalized count data but is also applicable to the RPKM data. In this chapter, it is used to analyze the RPKM data. Just as the LIMMA method, it will also be used to compare the analysis of the RPKM data and the analysis of the raw count data with normalization other than RPKM.

4.3.3 edgeR

To adjust for the library size L_{ij} and gene length λ_{ijg} in the edgeR method, the following log-linear model is used:

$$\log[E(y_{ijg})] = X_{ij}^T\beta_g + \log(L_{ij}) + \log(\lambda_{ijg}) - \frac{\sum_{g=1}^{G}\log(\lambda_{ijg})}{G} \quad (38)$$

where X_{ij}^T is the design matrix and β_g is a vector of regression coefficients for the g^{th} gene. Only the TMM normalization is considered for the edgeR method in this chapter. To apply the TMM normalization, the effective library sizes, which are the TMM normalization factors time the corresponding library sizes, are used in Equation (38) instead of the original library sizes. The raw count data are assumed to follow the NB distribution specified in Equation (2).

In the GLM framework, the NB dispersion parameter ϕ_g can be estimated by assuming: 1) all genes have a constant dispersion parameter through Equation (30); or 2) there is a trend between dispersion parameter and average count; or 3) the dispersion parameter is gene-specific through Equation (31). The Cox-Reid profile-adjusted likelihood method is used in estimating these dispersion parameters. The statistical test is the likelihood ratio test, which is obtained from fitting NB GLM models. In this dissertation, the dispersion parameter is considered gene-specific when using the edgeR method. The edgeR method is not designed for the RPKM data but is able to analyze the raw count data adjusting for the gene length and the library size in a GLM framework. On the contrary, the two other statistical methods are able to analyze the RPKM data directly.

4.4 Simulation

A simple balanced study design with two experimental conditions is assumed in simulation. To examine the performance of the normalization and statistical methods, a series of simulations have been conducted. The data are also simulated using the three variance functions (Table 2). Sample sizes were chosen to be 10, 20, and 40, split evenly between two treatment groups. Simulated genes with zero total count were replaced with a new simulated gene. Gene length is randomly selected from the human gene lengths from the University of California Santa Cruz (UCSC) database (Kent et al., 2002; Rosenbloom et al., 2015). In each dataset, there are 20,000 genes simulated and 5% of them are randomly chosen as DE genes. The treatment effects of DE genes were set to be constant as the fold change of 1.5 or 3 which represent low and high treatment effect respectively. Gene length is set to be either fixed or varied across experimental

conditions. In each scenario (3 variance functions x 3 sample sizes x 2 treatment effects x 2 gene length settings = 36), 200 datasets were simulated. For the DE genes, half of them are simulated as over-expressed and half of them are simulated as under-expressed. The RPKM data were simulated as follows: First, the average count for each gene was randomly sampled from the within-group mean normalized count in the Himes data. The average count was then assigned to all samples. If the gene is over-expressed, the average count multiplied by the treatment effect will be assigned to the samples in the second treatment group. If the gene is under-expressed, the average count divided by the treatment effect will be assigned to the samples in the second treatment group. The next step is to modify the average count further by multiplying a random scale factor which mimics the library size effect. This random factor is two to the power of a normal random variable with mean 0 and standard deviation 0.5. The average count is further modified by multiplying the randomly selected gene length. Only one gene length is randomly selected for each gene. In a varied gene length setting, the gene lengths of the orthologs are assumed to be different to a low or moderate extent. The gene length in the second condition is modified by a random scale factor, which is two to the power of a normal random variable with mean 0 and standard deviation $1/3$. The modified average count is then divided by a constant 10^3. With this scaled average count, the corresponding variance is calculated using the variance functions in Table 2. With the mean and the calculated variance, the shape and rate parameters of the gamma distribution can be estimated by a method of moments approach. A gamma random variable is then generated using these parameter estimates. This continuous random variable is then rounded to a discrete

number. This discrete number is used as raw count data. From the raw count data, the corresponding RPKM values are found using Equation (35).

4.5 Simulation Results

Using the QuasiDE or LIMMA methods, ten ways to analyze the RPKM dataset are compared in simulation: 1) analyzing the RPKM data without further normalization; 2) analyzing the RPKM data with the TMM normalization; 3) analyzing the RPKM data with the RLE normalization; 4) analyzing the RPKM data with the UQ normalization; 5) analyzing the RPKM data with the scaling normalization; 6) analyzing the RPKM data with the quantile normalization; 7) analyzing the RPKM data with the cyclic loess normalization; 8) analyzing the RPKM data with the invariant set normalization; 9) analyzing the raw count data with the TMM normalization; 10) analyzing the raw count data with the gene length and TMM normalization: the raw count data is first divided by the gene length and multiplied by a constant 10^3, and the resulting data is normalized by the TMM method. Method 10 is used only when the gene length is varied across samples. When gene length is fixed across samples, Method 10 reduces to Method 9. Using the QuasiDE method, the first nine methods used in the fixed gene length setting are compared in Figure 39 in the simulation setting of sample size 10 and low fold change. In Figure 39, the sensitivities are similar across different approaches. In the QP and LN data, the real FDRs of Method 1, 2, 3, 4 have many large outliers. In the QN data, there is one case of the real FDRs in Method 1, 2, 3, 4 is zero since their sensitivities are zero. By definition of FDR, it is defined as zero when there is no DE genes found. In all three types of data, only Method 5 and 9 have the real FDRs reasonably controlled. Method 5 is to analyze the RPKM data with the scaling

normalization and Method 9 is to analyze the raw count data with the TMM normalization. Comparing Method 5 and 9, Method 9 has slightly higher sensitivity and better controlled real FDR with fewer outliers. Method 9 is superior to Method 5, which suggests that analyzing the raw count data with proper normalization when raw count data is available is a better choice. This pattern remains the same for the other simulation settings in fixed gene length setting (different sample sizes and treatment effects). Detailed results for the other simulation settings are presented in Appendix B (Figure B1-B5). Overall, when the gene length is fixed across samples, analyzing raw count data with the TMM normalization is the best method. If there is only RPKM data available, the scaling normalization is the best choice.

Using the QuasiDE method, the ten methods are compared in Figure 40 in the simulation setting of sample size 10, low fold change and varied gene length. In Figure 40, the sensitivity of Method 9 is the highest in all three types of data but with a very badly controlled FDR. In the QP and LN data, the sensitivities of Method 1, 2, 3, 4 and 10 are in the second largest group. Method 5, 6, 7 and 8 have similar sensitivities but are lower than the other methods. In the QN data, Method 9 has the second largest sensitivity. The methods other than Method 9 and 10 have similar sensitivities. For the real FDR, it is best controlled in Method 5 and 10. Comparing Method 5 and 10, Method 10 has FDR better controlled, which indicates that analyzing the raw count data with proper normalization when raw count data is available is superior to analyzing the RPKM data with the scaling normalization. In addition, Method 9 has very badly controlled FDR, which indicates the gene length effect must be adjusted to have a valid analysis. This pattern remains the same for the other simulation settings in the varied

gene length setting (different sample sizes and treatment effects), which are presented in Appendix B (Figure B6-B10). In summary, when the gene length is varied across experimental conditions, analyzing the raw count data normalized by the gene length and the TMM method is a better choice compared with the analysis of the RPKM data. It not only has a higher sensitivity but also a better controlled FDR without many outliers. If there is only the RPKM data available, the scaling normalization is the best choice. The simulation results by the LIMMA method are similar to the QuasiDE method. The details are presented in Appendix C (Figure C1-C12). To understand the impact of the statistical methods, they are compared in ten scenarios: in the setting of fixed gene length, 1) the raw count data are analyzed by the edgeR method using the TMM normalization; 2) the raw count data are normalized by the TMM method and analyzed by the LIMMA method; 3) the RPKM data are normalized by the scaling method and analyzed by the LIMMA method; 4) the raw count data are normalized by the TMM method and analyzed by the QuasiDE method; 5) the RPKM data are normalized by the scaling method and analyzed by the QuasiDE method; in the setting of varied gene length, 6) the simulated raw count data are analyzed by the edgeR method adjusting for the gene length and the library size with the TMM normalization; 7) the raw count data are normalized by the gene length and the TMM method and then analyzed by the LIMMA method; 8) the simulated RPKM data are normalized by the scaling method and analyzed by the LIMMA method; 9) the raw count data are normalized by the gene length and TMM method and then analyzed by the QuasiDE method; 10) the simulated RPKM data are normalized by the scaling method and analyzed by the QuasiDE

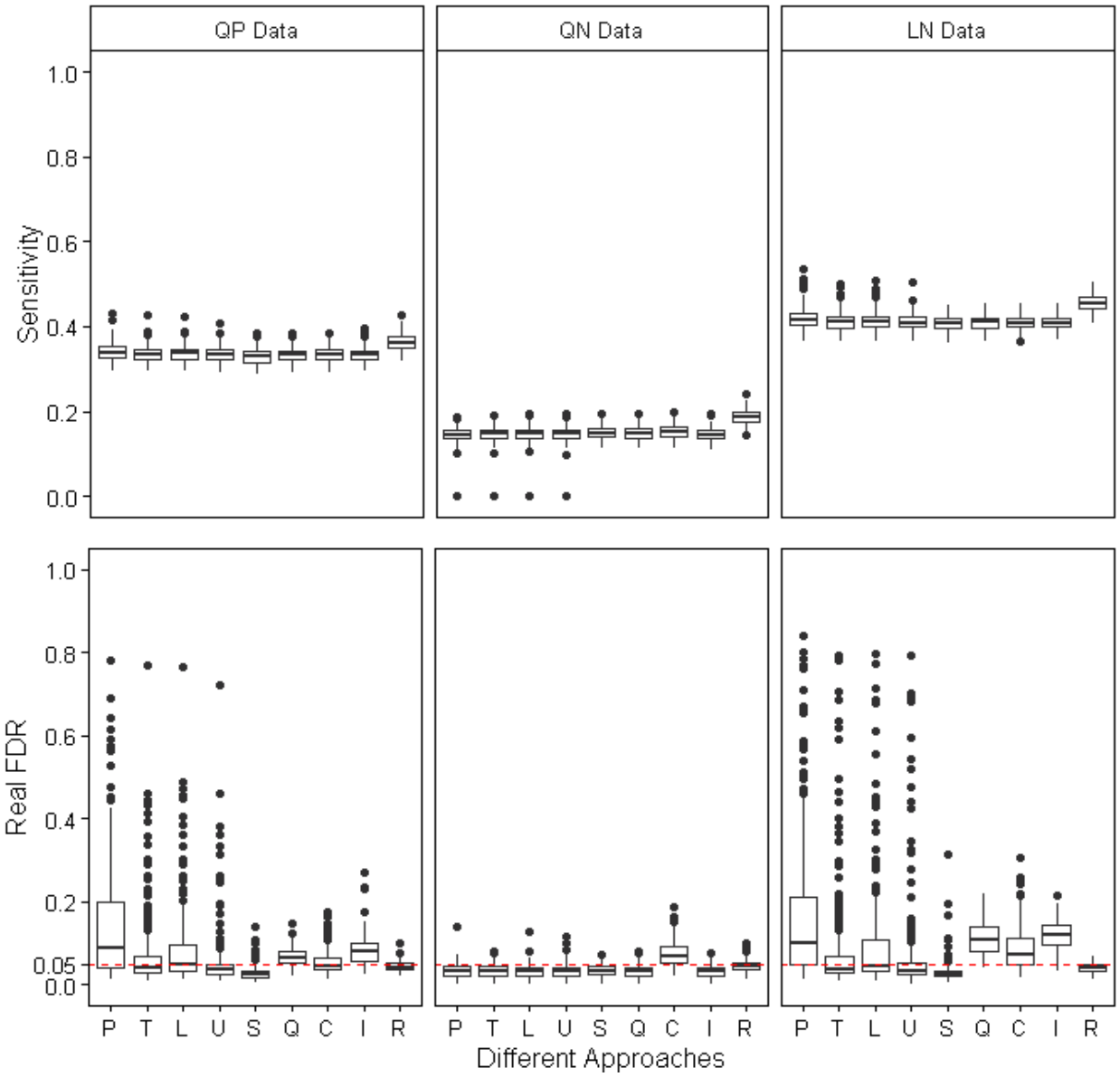

Figure 39. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, Low Fold Change, Fixed Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

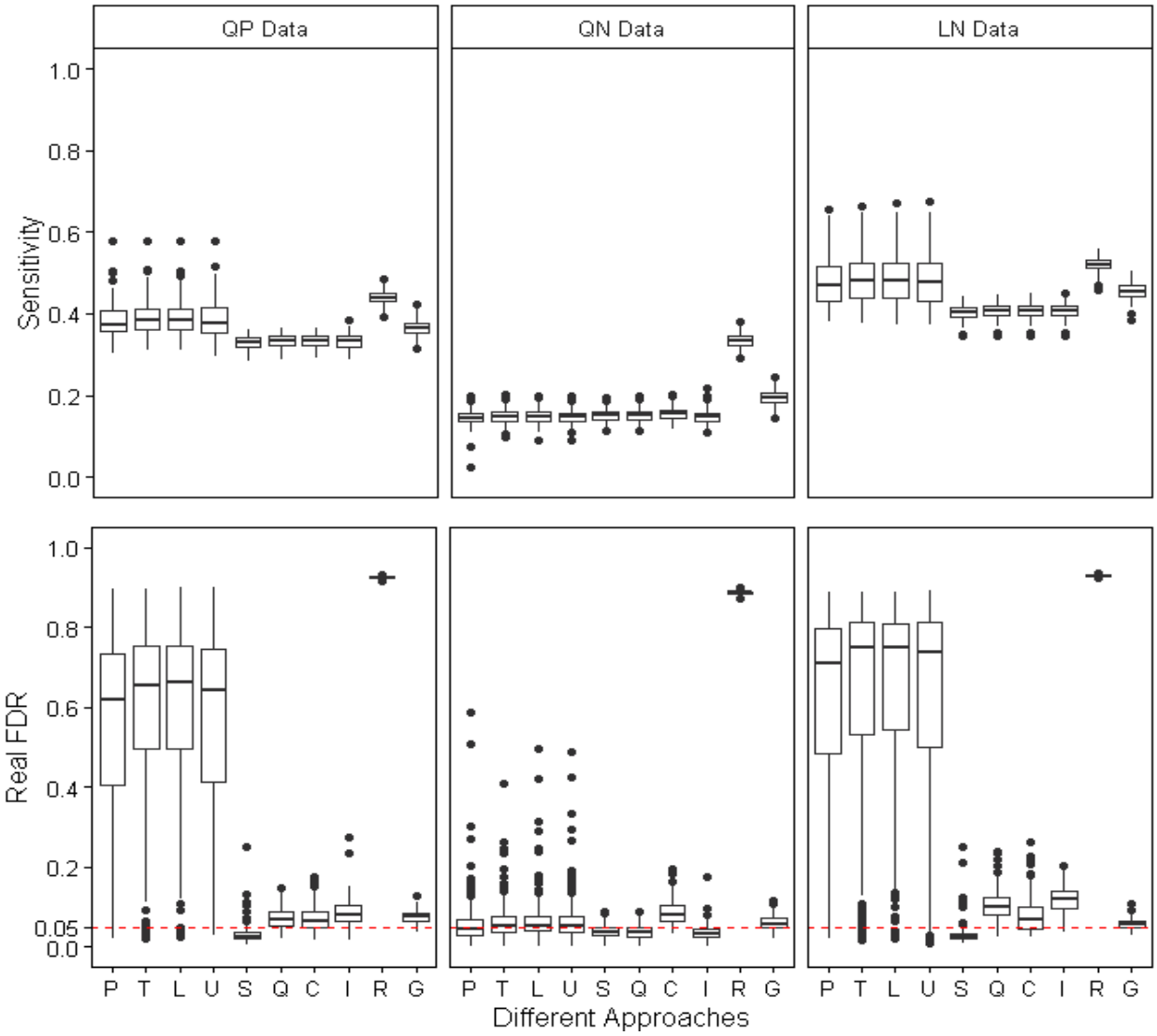

Figure 40. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, Low Fold Change, Varied Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

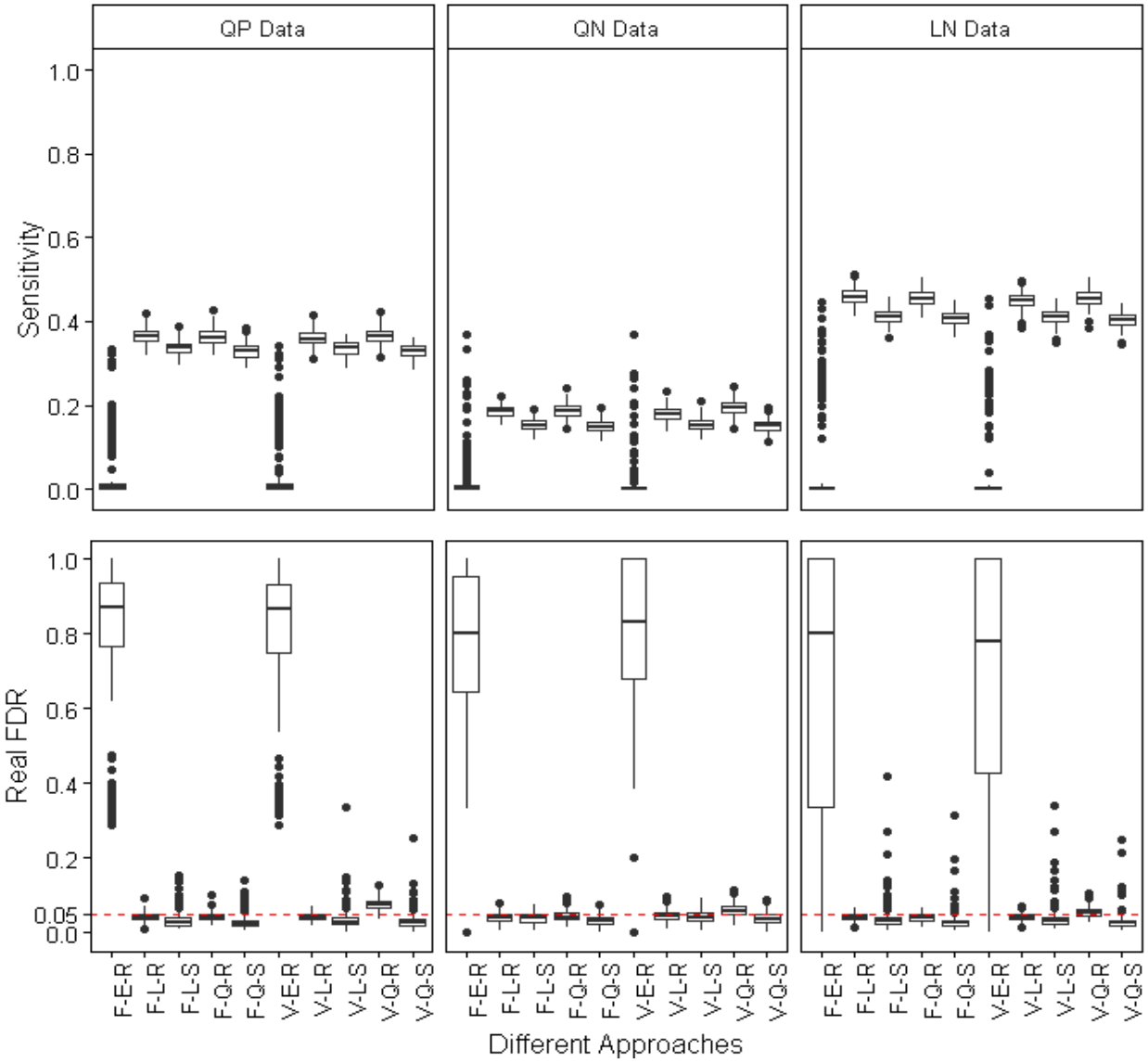

Figure 41. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, Low Fold Change)

The first part of the labels on horizontal axis indicates the gene length setting: F = Fixed Gene Length; V = Varied Gene Length. The second part indicates the statistical method: E = edgeR; L = LIMMA; Q = QuasiDE. The last part indicates the type of data: R = Raw count data normalized by the TMM method when gene length is fixed; Raw count data normalized by the gene length and the TMM method when gene length is varied. S = RPKM data with the scaling normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

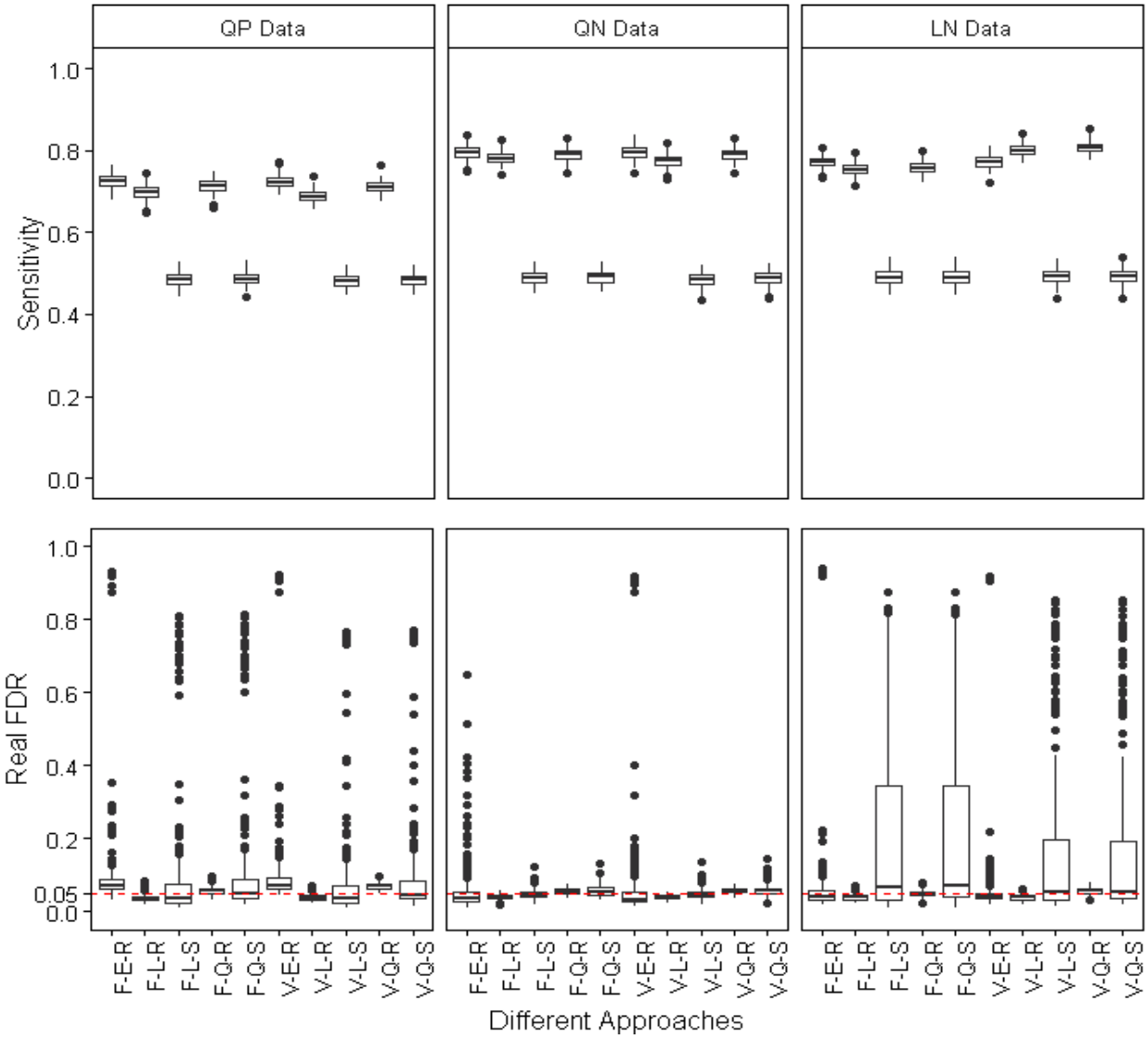

Figure 42. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, High Fold Change)

The first part of the labels on horizontal axis indicates the gene length setting: F = Fixed Gene Length; V = Varied Gene Length. The second part indicates the statistical method: E = edgeR; L = LIMMA; Q = QuasiDE. The last part indicates the type of data: R = Raw count data normalized by the TMM method when gene length is fixed; Raw count data normalized by the gene length and the TMM method when gene length is varied. S = RPKM data with the scaling normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

method. These scenarios are compared in Figure 41 in the simulation setting of sample size 10 and low fold change. Scenarios 1 and 6 have very low sensitivities, in which the edgeR method loses sensitivity. This is due to the low treatment effect and relative small sample size. The performance of the QuasiDE method is similar to the LIMMA method and they are better than the edgeR method. Analyzing the raw count data normalized by the gene length and TMM method has a higher power than analyzing the RPKM data. In addition, the real FDR is better controlled with fewer outliers. The performance of the edgeR method gets better in a relative larger sample size and high treatment effect. This is shown in Figure 42, in which the above scenarios are compared in the simulation setting of sample size 40 and high treatment effect. The edgeR method is comparable with the QuasiDE and LIMMA methods in analyzing the RPKM data but is still inferior to the QuasiDE and LIMMA methods in analyzing the raw count data normalized by the gene length and TMM method. In this figure, analyzing the raw count data normalized by the gene length and TMM method is also better than analyzing the RPKM data normalized by the scaling method for the QuasiDE and LIMMA methods. The comparisons of the above scenarios in other simulation settings are presented in Appendix D (Figure D1-D4). In summary, the QuasiDE and LIMMA methods have similar performance and are better than the edgeR method in most of the simulation settings. Only in the setting of high treatment effect and sample size 40, the edgeR method is competitive to the QuasiDE and LIMMA methods in analyzing the RPKM data. Analyzing the raw count data normalized by the gene length and TMM method is better than analyzing the RPKM data in all the simulation settings for the QuasiDE and LIMMA methods.

4.6 Case Study

In Reid data (Reid et al., 2012), toxoplasma gondii is a zoonotic protozoan parasite and Neospora caninum is its close relative. Both species share many common features. The research interest is to detect those orthologs which have an interaction effect between species and time. The dataset has 6,017 pairs of one-to-one orthologs, which are used in this real data analysis. The ortholog pairs were found by Stacy Hung (Hospital for Sick Children and University of Toronto). The gene lengths of these orthologs are mostly different between these two species to a low or moderate extent. The raw count data are analyzed in three approaches: the first approach is to analyze the raw count data by the edgeR method adjusting for the gene length and the library size effect; the other two approaches are using the QuasiDE and LIMMA methods, in which the raw count data are normalized by the gene length and TMM method. The RPKM data are analyzed by the QuasiDE and LIMMA methods in which the RPKM data is further normalized by the scaling method. To detect those orthologs which have an interaction between time and species, the time covariate is modeled as a continuous variable. In the QuasiDE method, the variance function is estimated by the mean variance pairs within each subgroup: there are four subgroups for Toxoplasma gondii and three subgroups for Neospora caninum. The number of orthologs with interaction detected by the QuasiDE and LIMMA methods between analyzing the raw count data with the gene length and TMM normalization and analyzing the RPKM data with the scaling normalization are presented in Table 16. The results from these analyses are similar. Since the main focus of this chapter is on the analysis of the RPKM data, the analysis results by the QuasiDE and LIMMA methods using the raw count data are not

Table 16: Comparison between the Analysis of the Raw Count Data and the RPKM Data by the QuasiDE and LIMMA Methods

Number of Orthologs			**Analysis of the Raw Count Data**			
			QuasiDE		**LIMMA**	
			EE	**DE**	**EE**	**DE**
Analysis of the RPKM data	**QuasiDE**	**EE**	5418	29		
		DE	32	538		
	LIMMA	**EE**			5355	4
		DE			57	601

Table 17: Top 10 DE Orthologs Detected by the QuasiDE Method and their Corresponding Results in the LIMMA and edgeR Methods

Orthologs	**QuasiDE**		**LIMMA**		**edgeR**	
	Rank	**Adjusted p-value**	**Rank**	**Adjusted p-value**	**Rank**	**Adjusted p-value**
NCLIV_009495.1* TGME49_099000**	1	8.0e-04	459	3.0e-02	5869	1.0e-00
NCLIV_034790 TGME49_072530	2	1.8e-03	71	3.0e-03	56	2.9e-07
NCLIV_017600 TGME49_042570	3	4.5e-03	4058	7.2e-01	1	6.3e-21
NCLIV_018330 TGME49_043610	4	4.5e-03	6	5.6e-04	41	4.8e-08
NCLIV_065203 TGME49_047570	5	5.0e-03	183	7.1e-03	30	1.3e-08
NCLIV_033460 TGME49_033860	6	7.3e-03	68	2.9e-03	491	6.8e-03
NCLIV_059660 TGME49_016610	7	7.3e-03	2724	4.4e-01	80	3.8e-06
NCLIV_027170 TGME49_059620	8	7.3e-03	1082	1.1e-01	157	1.2e-04
NCLIV_004720 TGME49_020940	9	7.3e-03	129	4.7e-03	283	8.5e-04
NCLIV_032360 TGME49_032380	10	7.3e-03	5	4.6e-04	155	1.2e-04

***** *Neospora caninum gene ID*

****** *Toxoplasma gondii gene ID*

Note: The RPKM data with the scaling normalization are used in the QuasiDE and LIMMA methods. The raw count data is used in the edgeR method adjusting for the gene length and using the TMM normalization.

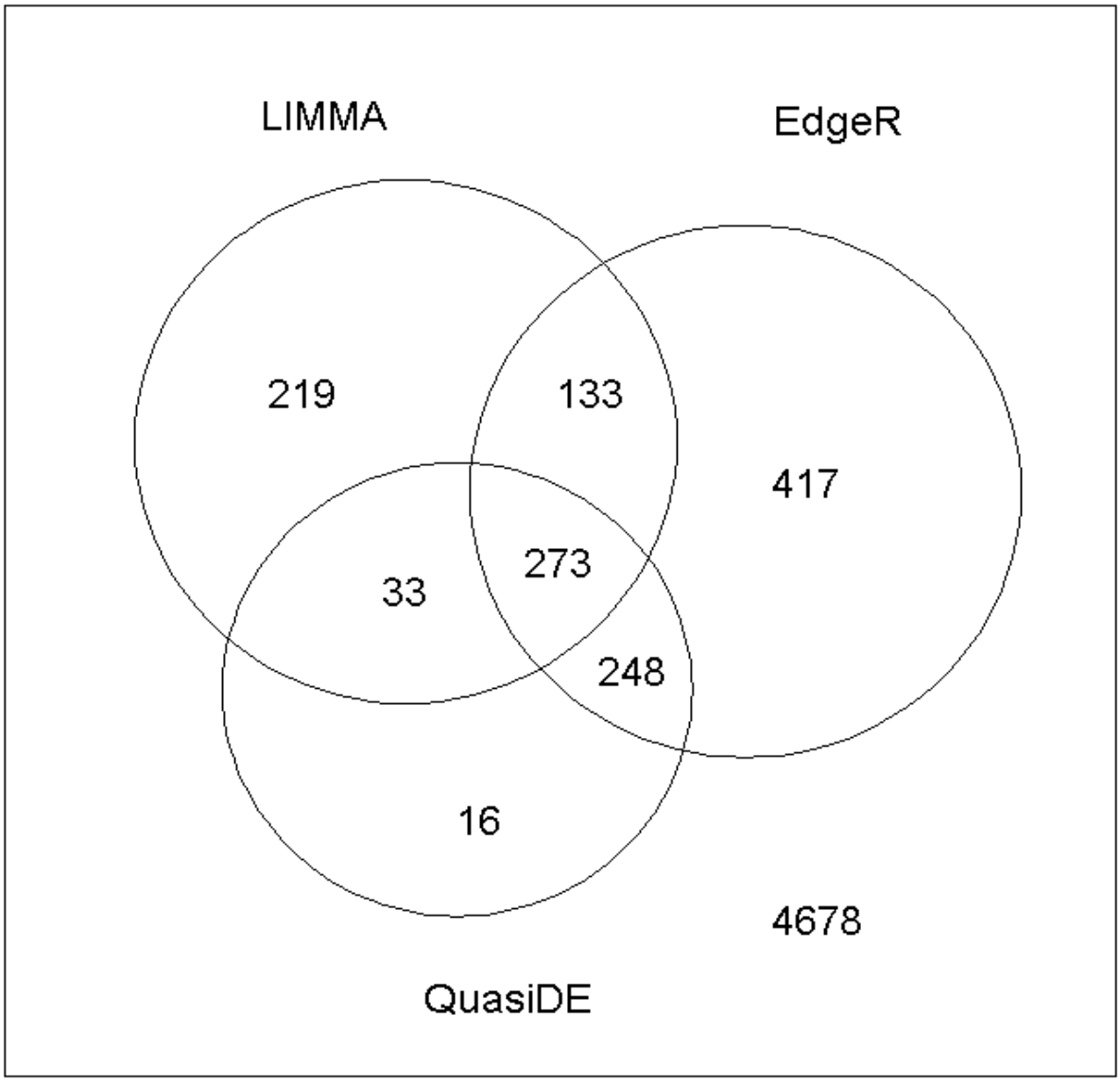

Figure 43. Venn Diagram of Counts for Ortholog having the interaction between species and time (Reid Data)

This shows the number of orthologs having the interaction betwwen time and species detected by the QuasiDE, LIMMA, edgeR methods, and their relationship. The RPKM data is normalized first with the scaling method and then analyzed by the QuasiDE and LIMMA methods. They are compared with the edgeR method adjusting for the gene length and using the TMM normalization.

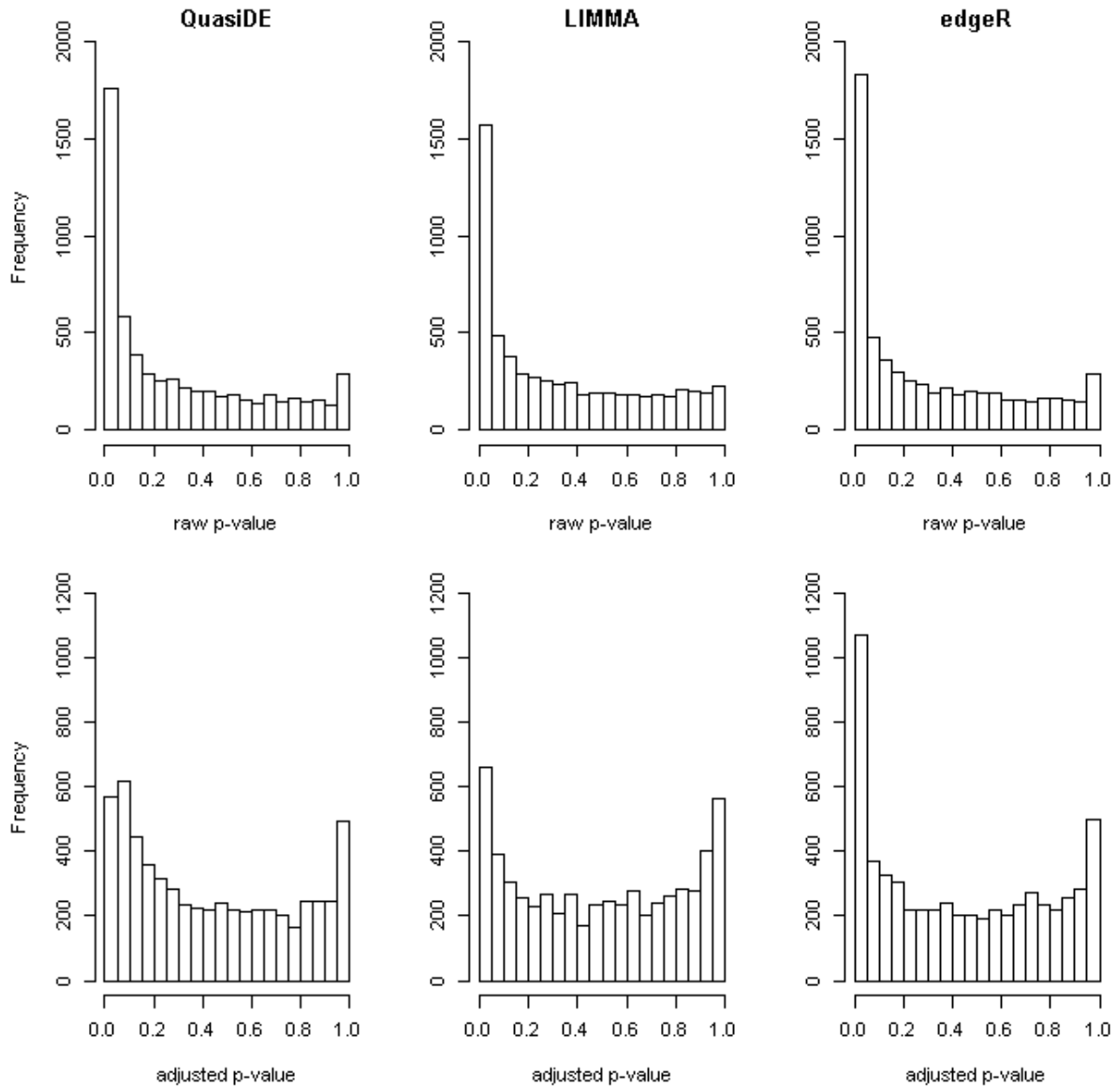

Figure 44. P-values and Adjusted p-values by QuasiDE, LIMMA and edgeR (Reid Data)

The three panels in the first row show the distributions of the raw p-values by the QuasiDE, LIMMA and edgeR methods. The three panels in the second row show the distributions of adjusted p-values by the QuasiDE, LIMMA and edgeR methods. The RPKM data are used in the QuasiDE and LIMMA methods. The raw count data are used in the edgeR method.

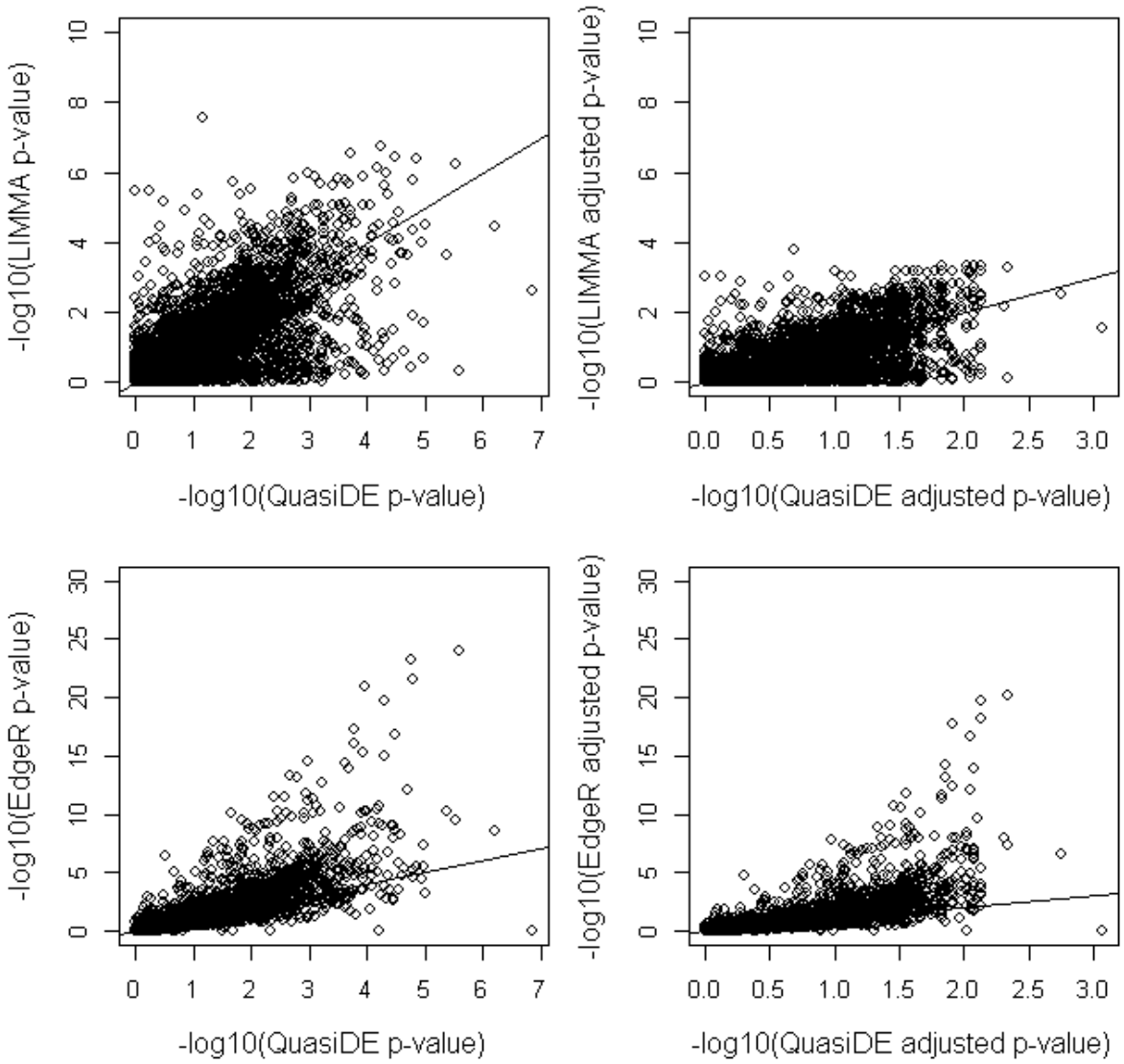

Figure 45. Comparison of the p-values and Adjusted p-values by the QuasiDE, LIMMA and edgeR Methods (Reid Data)

The panels in the left column show the p-values by the QuasiDE method vs. p-values by the LIMMA and edgeR methods in -log10 scale. The panels in the right column show the adjusted p-values by the QuasiDE method vs. adjusted p-values by the LIMMA and edgeR methods in -log10 scale. The RPKM data are used in the QuasiDE and LIMMA methods. The raw count data are used in the edgeR method.

presented subsequently. Instead, the analysis results by the QuasiDE and LIMMA methods using the RPKM data as well as the analysis results by the edgeR method are presented. The edgeR method is included since it is a popular method in the RNA-seq data analysis, although it is only possible to analyze the raw count data. The number of orthologs with interaction detected by the QuasiDE, LIMMA methods using the RPKM data and the edgeR method using the raw count data are compared in Figure 43. The edgeR, LIMMA and QuasiDE methods have detected 1,071, 658 and 570 orthologs with interaction respectively. The top ten orthologs with interaction detected by the QuasiDE method using the RPKM data are listed in Table 17 with the corresponding rank and adjusted p-values by the LIMMA method using the RPKM data and the edgeR method using the raw count data. The gene, which is ranked third by the QuasiDE method, is not statistically significant in the LIMMA method but is statistically significant in the edgeR method. The corresponding raw p-values and adjusted p-values by these three methods are compared in Figure 44. The QuasiDE and edgeR methods have a higher number of small p-values (< 0.05) compared with the LIMMA method. After multiple comparison correction, the edgeR method has the largest number of small adjusted p-values (< 0.05). The small p-values and adjusted p-values are further compared on -log10 scale by these three methods in Figure 45. The small p-values of the QuasiDE method using the RPKM data appear to be similar as those from the LIMMA method using the RPKM data. The edgeR method produced much more extreme small p-values. The adjusted p-values have a similar pattern.

4.7 Discussion

Dillies et al. (2012) compared many normalization methods including the RPKM. In their paper, the TMM and RLE normalization methods are recommended. They have reported that using RPKM directly results in poorly controlled FDR. This is consistent with the findings in this dissertation.

In this chapter, the normalization methods are compared assuming only RPKM data available. This is different from the work by Dillies et al. (2012), which assumed the raw count data is available. In their work, the normalization methods are applied to the raw count data. In the Gene Expression Omnibus database (Edgar et al., 2002; Barrett et al., 2013), many datasets are uploaded as RPKM datasets only. Analyzing RPKM data is necessary if the raw count data can not be obtained.

The LIMMA method has not been considered in Chapter 2 since the method was developed for the microarray continuous intensity data, but not for the RNA-seq count data. Also, extending the LIMMA method in the setting of the RNA-seq count data is not the major goal in Chapter 2. In Chapter 4, the LIMMA method has been considered since the continuous RPKM data is a potential candidate to be analyzed by the LIMMA method. Furthermore, to compare with the analyses using the raw count data, the LIMMA method is also extended to be applied to the continuous normalized count data.

When gene length is varied, a method has been proposed to normalize the raw count data by the gene length and then use the TMM method. This method works well and is valid in the simulation. However, the proper normalization methods in this setting have not been studied extensively in this dissertation. It is still not clear what is the best

normalization method to use in this setting when the gene length is varied across samples. This deserves further research.

In the literature, there are some other gene expression units in RNA-seq experiments such as FPKM (fragments per kilobase per million), TPM (normalized by gene length first to have reads per kilobase (RPK), then scaled by a factor which represents the total number of RPKs in a sample) and CPM (counts per million, which is calculated as dividing the raw count by its library size and then multiplying by 10^6). These metrics are either the raw read counts adjusted by the gene length or the library size or both. Like the RPKM, they can be considered as a certain type of normalization. In a situation where the comparison is between species, adjusting for different gene lengths is a must. In this case, RPKM, FPKM and TPM are useful. Just like the RPKM, the FPKM and TPM may need further normalization to have a valid analysis. In a situation that the gene length is fixed across samples, adjusting for the gene length appears to be irrelevant when the effect can be canceled out in the statistical analysis. In this situation, the raw count data and the CPM can be used. For the CPM, it may need further normalization as well. The QuasiDE method can be useful to analyze all these metrics.

4.8 Conclusion

In this chapter, it has been shown that it is not proper to analyze the RPKM data directly without any further normalization. Comparing different normalization methods initially not devised for the RPKM data, simple scaling is the best method which produces results with relatively well controlled real FDR in most cases. Compared with the valid approach to analyze the raw count data, the further normalization on the RPKM data loses efficiency and also has a worse controlled FDR. When gene length varies across

samples, analyzing the raw count data without adjusting for the gene length is not a valid approach. The QuasiDE and LIMMA methods have similar performance and they are better than the edgeR method in most cases.

CHAPTER 5. Future Research

In this dissertation, a new method, the QuasiDE method, has been developed based on quasi-likelihood theory to detect DE genes in RNA-seq data analysis in an experiment with two conditions. It has been extended to some other popular experimental designs such as an experiment with multiple conditions, an experiment with block design and an experiment with factorial design. The QuasiDE method is also applied to analyze the RPKM data. In simulation, the QuasiDE method has been shown to have more similar sensitivities and FDRs across the data with different variance functions compared with other methods. Also the performance of the QuasiDE method is at least as good as the other popular methods. Throughout the new methodology development, the following topics have been found to be worth pursuing in future research:

1. It has been previously mentioned that the normalization methods in the RNA-seq data with varied gene length across samples deserve further research. Possible normalization methods may include two parts: correct or remove the impact from the gene length and the library size. The normalization can be on the gene length first or on the library size first. Alternatively, these two factors may be able to be adjusted simultaneously such as using the GLM methodology. The existing normalization methods used in the RNA-seq data analysis in the setting of fixed gene length can also be used in designing the potential normalization methods in the varied gene length setting. The validity and performance of these methods need to be evaluated.
2. It also has been mentioned that most normalization methods assume the balance of the over-expressed and under-expressed DE genes. The influence of imbalance on the normalization is unknown. To explore that, a simulation study can be employed.

With different extents of imbalance in the simulated datasets, the performance of different normalization methods can be evaluated. In the evaluation, the QuasiDE method can be used as the statistical method along with other popular methods.

3. Most normalization methods assume most genes are non-DE genes. In this dissertation, 5% of DE genes are used in simulation. Also a small simulation using different DE ratios has been conducted. To fully understand the impact of a high proportion of DE genes on different statistical methods, a comprehensive simulation study can be used. There are possibly some interactions between over-expressed and under-expressed imbalance and the percentage of DE genes. This can be studied in a more extensive simulation in which these two factors are used.
4. There are numerous possible experimental designs in RNA-seq experiments. It is important to understand the proper way to use the QuasiDE method in those designs, such as matched pairs design and time course design. In matched pair design, the differences within pairs are of interest and only form one group. To apply the QuasiDE method in this setting, the variance function can be estimated by the mean-variance pairs in this group. The quasi-likelihood F statistic can be constructed by comparing two models: one model is that the true difference is zero and the other model is that the true difference is the mean of the differences. The performance of the QuasiDE method in matched pairs design can be evaluated through simulation. In time course design, the gene expressions can be measured in the same sample but at different time points. Alternatively, the gene expressions can be measured in different samples at different time points. For example, the mouse in some cases is sacrificed after the tissues are taken out. Also, different ways to model time can be

studied. Time can be modeled as a continuous covariate or as a categorical variable. The proper method to estimate the variance function in these models can be studied.

5. There are also other gene expression measures such as FPKM, TPM and CPM. These types of data are potentially analyzable by the QuasiDE method. As when studying the RPKM data, the proper normalization methods and statistical methods for these types of data can be studied.
6. The computation time for the QuasiDE method is longer than the other three methods. It has been mentioned that this is due to the numeric evaluation of the quasi-deviance in the R program. This can be possibly improved by optimizing the algorithm, using the parallel processing or using the C program implementation.

BIBLIOGRAPHY

Affymetrix. (2001). *Affymetrix Microarray Suite Users Guide*. Version 5.0.

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

Anders S, Huber W. (2016). *Differential expression of RNA-Seq data at the gene-level - the DESeq package*

Ansorge WJ. (2009). Next-generation DNA sequencing techniques. *New Biotechnol.* 25(4): 195-203

Auer PL, Doerge RW. (2010). Statistical design and analysis of RNA sequencing data.*Genetics.* 185(2): 405-416

Barrett T, Wilhite SE, Ledoux P, Evangelista C, Kim IF, Tomashevsky M, Marshall KA, Phillippy KH, Sherman PM, Holko M, Yefanov A, Lee H, Zhang N, Robertson CL, Serova N, Davis S, Soboleva A. (2013). NCBI GEO: archive for functional genomics data sets--update. *Nucleic Acids Res.* 41(Database issue): D991-5

Benjamini Y, Hochberg Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *J. R. Stat. Soc. Series B Stat. Methodol.* 57(1): 289-300

Bloom JS, Khan Z, Kruglyak L, Singh M, Caudy AA. (2009). Measuring differential gene expression by short read sequencing: quantitative comparison to 2-channel gene expression microarrays. *BMC Genomics*. 10: 221

Bolstad BM, Irizarry RA, Astrand M, Speed TP. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics.* 19: 185-93

Bottomly D, Walter NA, Hunter JE, Darakjian P, Kawanw S, Buck KJ, Searles RP, Mooney M, McWeeney SK, Hitzemann R. (2011). Evaluating Gene Expression in

C57BL/6J and DBA/2J Mouse Striatum Using RNA-Seq and Microarrays. *PLoS ONE* 6(3): e17820

Bullard JH, Purdom E, Hansen KD, Dudoit S. (2010). Evaluation of statistical methods for normalization and differential expression in mRNA-seq experiments. *BMC Bioinformatics.* 11: 94

Chen Y, McCarthy DJ, Robinson MD, Smyth GK. (2016). *edgeR: differential expression analysis of digital gene expression data*. User's Guide

Cox DR, Reid N. (1987). Parameter Orthogonality and Approximate Conditional Inference, *J R Statist. Soc. B.* 49(1)

Cunningham F, Amode MR, Barrell D, Beal K, Billis K, Brent S, Carvalho-Silva D, Clapham P, Coates G, Fitzgerald S, Gil L, Girón CG, Gordon L, Hourlier T, Hunt SE, Janacek SH, Johnson N, Juettemann T, Kähäri AK, Keenan S, Martin FJ, Maurel T, McLaren W, Murphy DN, Nag R, Overduin B, Parker A, Patricio M, Perry E, Pignatelli M, Riat HS, Sheppard D, Taylor K, Thormann A, Vullo A, Wilder SP, Zadissa A, Aken BL, Birney E, Harrow J, Kinsella R, Muffato M, Ruffier M, Searle SM, Spudich G, Trevanion SJ, Yates A, Zerbino DR, Flicek P. (2015). Ensembl 2015. *Nucleic Acids Res.* 43:D662-9

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

Di Bucchiano A. (1999). Combinatorics, computer algebra and the Wilcoxon-Mann-Whitney test. *J Stat Plann Inference.* 79: 349-64

Di YM, Schafer DW, Cumbie JS, Chang JH. (2011). The NBP negative binomial model for assessing differential gene expression from RNA-seq. *Stat Appl Genet Mol Biol.* 10(1): 1-28

Dillies MA, Rau A, Aubert J, Hennequet-Antier C, Jeanmougin M, Servant N, Keime C, Marot G, Castel D, Estelle J, Guernec G, Jagla B, Jouneau L, Laloë D, Le Gall C, Schaëffer B, Le Crom S, Guedj M, Jaffrézic F. (2012). A comprehensive evaluation of

normalization methods for Illumina high-throughput RNA Sequencing data analysis. *Brief Bioinform.* 14(6): 671-83

Dolatshad H, Pellagatti A, Fernandez-Mercado M, Yip BH, Malcovati L, Attwood M, Przychodzen B, Sahgal N, Kanapin AA, Lockstone H, Scifo L, Vandenberghe P, Papaemmanuil E, Smith CW, Campbell PJ, Ogawa S, Maciejewski JP, Cazzola M, Savage KI, Boultwood J. (2015). Disruption of SF3B1 results in deregulated expression and splicing of key genes and pathways in myelodysplastic syndrome hematopoietic stem and progenitor cells. *Leukemia*. 29(8): 1798

Edgar R, Domrachev M, Lash AE. (2002). Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res.* 30(1):207-10

Esnaola M, Puig P, Gonzalez D, Castelo R, Gonzalez JR. (2013). A flexible count data model to fit the wide diversity of expression profiles arising from extensively replicated RNA-seq experiments. *BMC Bioinformatics.* 14: 254

Himes BE, Jiang X, Wagner P, Hu R, Wang Q, Klanderman B, Whitaker RM, Duan Q, Lasky-Su J, Nikolos C, Jester W, Johnson M, Panettieri R Jr, Tantisira KG, Weiss ST, Lu Q. (2014). RNA-Seq Transcriptome Profiling Identifies CRISPLD2 as a Glucocorticoid Responsive Gene that Modulates Cytokine Function in Airway Smooth Muscle Cells. *PLoS One*. 9(6): e99625

Härdle W. (1986). Approximations to the Mean Integrated Squared Error with Applications to optimal Bandwidth Selection for Nonparametric Regression Function Estimators. *J Multivar Anal*. 18: 150-168

Li J, Tibshirani, R. (2013). Finding consistent patterns: a nonparametric approach for identifying differential expression in RNA-Seq data. *Stat Methods Med Res.* 22(5): 519-36

Li C, Wong WH. (2001). Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error application. *Genome Biol.* 2(8): RESEARCH0032

Love MI, Anders S, Kim V, Huber W. (2015). RNA-Seq workflow: gene-level exploratory analysis and differential expression. *F1000Res.* 4: 1070

Lund SP, Nettleton D, McCarthy DJ, Smyth GK. (2012). Detecting Differential Expression In RNA-sequence Data Using Quasi-likelihood with Shrunken Dispersion Estimates. *Stat Appl Genet Mol Biol.* 11(5)

Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, Haussler D. (2002). The human genome browser at UCSC. *Genome Res.* 12(6): 996-1006

Marioni JC, Mason CE, Mane SM, Stephens M, Gilad Y. (2008). RNA-Seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Res*. 18**:** 1509–1517

McCarthy, JD, Chen Y, Smyth GK. (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Res.* 40(10): 9

McCullagh P. (1983). Quasi-Likelihood Functions. *Ann Stat.* 11(1): 59-67

McCullagh P, Nelder JA. (1989). *Generalized Linear Models*. Second ed. Chapman and Hall

Mortazavi A, Williams BA, McCue K, Schaeffer L, Wold B. (2008). Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nat Methods.* 5(7): 621-8

Metzker ML. (2010). Sequencing technologies - the next generation. *Nat Rev Genet.* 11(1): 31-46

Morgan MT. (2014). *RNASeq Analysis*. URL https://www.bioconductor.org/help/course-materials/2014/SeattleFeb2014/RNASeq.pdf

Nettleton D, Hwang JTG, Caldo RA, Wise RP. (2006). Estimating the number of true null hypotheses from a histogram of P values. *J Agr Biol Envir St.* 11:337-356

Osborne, MR (1992). Fisher's method of scoring. *Int Stat Rev.* 60 : 99–117

Ozsolak F, Milos PM. (2011). RNA sequencing: advances, challenges and opportunities. *Nat Rev Genet.* 12(2): 87-98

Pichrell JK, Marioni JC, Pai AA, Degner JF, Engelhart BE, Nkadori E, Veyrieras JB, Stephens M, Gilad Y, Pritchard JK. (2010). Understanding mechanisms underlying human gene expression variation with RNA sequencing. *Nature.* 464(7289): 768-72

Pritchard CC, Hsu L, Delrow J, Nelson PS. (2001). Project normal: defining normal variance in mouse gene expression. *Proc Natl Acad Sci U S A.* 98(23): 13266-71

Reid AJ, Vermont SJ, Cotton JA, Harris D, Hill-Cawthorne GA, Könen-Waisman S, Latham SM, Mourier T, Norton R, Quail MA, Sanders M, Shanmugam D, Sohal A, Wasmuth JD, Brunk B, Grigg ME, Howard JC, Parkinson J, Roos DS, Trees AJ, Berriman M, Pain A, Wastling JM. (2012). Comparative genomics of the apicomplexan parasites Toxoplasma gondii and Neospora caninum: Coccidia differing in host range and transmission strategy. *PLoS Pathog.* 8(3): e1002567

Rice JR. (1969). *The approximation of functions, Vol II*. Addison-Wesley, Massachusetts

Rigby RA, Stasinopoulos DM, Akantziliotou C. (2008). A framework for modelling overdispersed count data, including the Poisson-shifted generalized inverse Gaussian distribution. *Comput Stat Data An.* 53(2): 381-393

Risso D, Nqai J, Speed TP, Dudoit S. (2014). Normalization of RNA-seq data using factor analysis of control genes or samples. *Nat Biotechnol.* 32(9):896-902

Risso D, Schwartz K, Sherlock G, Dudoit S. (2011). GC-Content Normalization for RNA-Seq Data. *BMC Bioinformatics*. 12: 480

Robinson MD, McCarthy DJ, Smyth GK. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics.* 26(1): 139-40

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

Robinson MD, Smyth GK. (2007). Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*. 23: 2881-2887.

Robinson MD, Smyth GK. (2008). Small sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*

Rosenbloom KR, Armstrong J, Barber GP, Casper J, Clawson H, Diekhans M, Dreszer TR, Fujita PA, Guruvadoo L, Haeussler M, Harte RA, Heitner S, Hickey G, Hinrichs AS, Hubley R, Karolchik D, Learned K, Lee BT, Li CH, Miga KH, Nguyen N, Paten B, Raney BJ, Smit AF, Speir ML, Zweig AS, Haussler D, Kuhn RM, Kent WJ. (2015). The UCSC Genome Browser database: 2015 update. *Nucleic Acids Res*. 43 (Database issue): D670-81

Seyednasrollah F, Laiho A, Elo LL. (2015). Comparison of software packages for detecting differential expression in RNA-seq studies. *Brief Bioinform*. 16(1): 59-71

Smyth GK. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Stat Appl Genet Mol Biol.* 3: article 3

Squadrito ML, Baer C, Burdet F, Maderna C, Gilfillan GD, Lyle R, Ibberson M, De Palma M. (2014). Endogenous RNAs modulate microRNA sorting to exosomes and transfer to acceptor cells. *Cell Rep*. 8(5): 1432-46

Storey JD, Tibshirani R. (2003). Statistical significance for genome-wide experiments. *Proc Natl Acad Sci.* 100: 9440-9445

Sultan M, Schulz MH, Richard H, Magen A, Klingenhoff A, Scherf M, Seifert M, Borodina T, Soldatov A, Parkhomchuk D, Schmidt D, O'Keeffe S, Haas S, Vingron M, Lehrach H, Yaspo ML. (2008). A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome. *Science*. 321(5891): 956-60

Sørlie T, Perou CM, Tibshirani R, Aas T, Geisler S, Johnsen H, Hastie T, Eisen MB, van de Rijn M, Jeffrey SS, Thorsen T, Quist H, Matese JC, Brown PO, Botstein D, Lønning PE, Børresen-Dale AL. (2001). Gene expression patterns of breast carcinomas

distinguish tumor subclasses with clinical implications. *Proc Natl Acad Sci U S A.* 98(19): 10869-74

Tjur T. (1998). Nonlinear Regression, Quasi Likelihood, and Overdispersion in Generalized Linear Models. *Am Stat*. 52(3) : 222-7

Wagner GP, Kin K, Lynch VJ. (2012). Measurement of mRNA abundance using RNA-seq data : RPKM measure is inconsistent among samples. *Theory Biosci.* 131(4): 281-5

Wang Z, Gerstein M, Snyder M. (2009). RNA-seq: a revolutionary tool for transcriptomics. *Nat Rev Genet.* 10(1): 57-63

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

Yang YH, Dudoit S, Luu P, Lin DM, Peng V, Ngai J, Speed TP. (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res.* 30

Yang K, Li J, Gao H. (2006). The impact of sample imbalance on identifying differentially expressed genes. *BMC Bioinformatics*. 7(Suppl 4):S8

Zerbino DR, Wilder SP, Johnson N, Juettemann T, Flicek PR. (2015). The ensembl regulatory build. *Genome Biol.* 16: 56

Zhang Y, Yu J, Lee C, Xu B, Sartor MA, Koenig RJ. (2015). Genomic binding and regulation of gene expression by the thyroid carcinoma-associated PAX8-PPARG fusion protein. *Oncotarget.* 6(38): 40418-32

APPENDIX A

```
# ============================================================================================
# R.code (Chapter 2)
# ============================================================================================

# Modified quasi function used in R GLM function
# The log link function is used and the variance function is a spline object
# Quasi-deviance is numerically evaluated

quasi1 <- function (link = "log", variance = "spline", splmodel = modnorm1)
{
  # trapz - trapezoid rule to calculate Area
   trapz <- function (x,y) {
            area <- NULL
            if (length(x) == 1) area <- 0
            else area <- sum((y[-1]+y[-length(y)])/2*(x[-1]-x[-length(x)]))
            return(area)
   }
 # evaluate the quasi-deviance numerically using  the spline variance function
   areadev.unit <- function(y,mu,wt){
            res <- NULL
            if ((length(mu) == 1) && length(mu)<length(y)) m = rep(mu,length(y))
            else m = mu
            for (i in 1:length(y)){
                     x <- seq(m[i], y[i],len = 501)
                     ql <- 2*(y[i]-x)/(exp(predict(splmodel,x)$y))
                     if (sum(is.nan(ql))>0) {
                              x <- x[! is.nan(ql)]
                              ql <- ql[! is.nan(ql)]
                     }
                     res <- c(res,trapz(x,ql))
            }
            return(res)
   }
   linktemp <- substitute(link)
   if (!is.character(linktemp))
      linktemp <- deparse(linktemp)
   if (linktemp %in% c("logit", "probit", "cloglog", "identity",
      "inverse", "log", "1/mu^2", "sqrt"))
      stats <- make.link(linktemp)
   else if (is.character(link)) {
      stats <- make.link(link)
      linktemp <- link
   }
   else {
      stats <- link
      linktemp <- if (!is.null(stats$name))
         stats$name
      else deparse(linktemp)
   }
   vtemp <- substitute(variance)
   if (!is.character(vtemp))
      vtemp <- deparse(vtemp)
   variance_nm <- vtemp
  # modified to add new option spline
   switch(vtemp, `spline` = {
      varfun <- function(mu) exp(predict(splmodel,mu)$y)
      validmu <- function(mu) all(mu > 0)
      dev.resids <- function(y, mu,wt) areadev.unit(y,mu,wt)
      initialize <- expression({
         n <- rep.int(1, nobs)
         mustart <- y + 0.1 * (y == 0)
      })
   }, variance_nm <- NA)
   if (is.na(variance_nm)) {
      if (is.character(variance))
         stop(gettextf("'variance' \"%s\" is invalid: possible values are \"spline\"",
            variance_nm), domain = NA)
```

```
        varfun <- variance$varfun
        validmu <- variance$validmu
        dev.resids <- variance$dev.resids
        initialize <- variance$initialize
        variance_nm <- variance$name
    }
    aic <- function(y, n, mu, wt, dev) NA
    structure(list(family = "quasi", link = linktemp, linkfun = stats$linkfun,
        linkinv = stats$linkinv, variance = varfun, dev.resids = dev.resids,
        aic = aic, mu.eta = stats$mu.eta, initialize = initialize,
        validmu = validmu, valideta = stats$valideta, varfun = variance_nm),
        class = "family")
}

# QuasiDE approach
# if there are two models, compare two models
# if only one model supplied,  then compare the model with null model
# return F statistic and raw p-value

quasiDE <- function(data,design1,design2=NULL,splmodel){
# data - the input data frame or matrix with normalized count data
# design 1 - design matrix for the full model
# design 2 - design matrix for the reduced model
# splmodel is the estimated spline variance function
j <- 1
res <- matrix(NA,nrow=nrow(data),ncol=2)
repeat{
# calculate some initial values in case the model fitting is failed
yobs=as.numeric(data[j,])
stf1 <- as.data.frame(cbind(yobs,design1))
sfit1 <- glm(yobs~.,data=stf1)
stf1$u <- predict(sfit1)
if (! is.null(design2)){
stf2 <-  as.data.frame(cbind(yobs,design2))
sfit2 <- glm(yobs~.,data=stf2)
stf2$u <- predict(sfit2)
}
# fit GLM using Quasi-likelihood and estimated spline varinance function
fit1 <- tryCatch({glm(yobs~design1, family = quasi1(variance = "spline", link = "log",splmodel=splmodel))}, error = function(e){} )
# if model fitting is failed, try new initial values
if (is.null(fit1)) fit1 <- tryCatch({glm(yobs~design1, family = quasi1(variance = "spline", link =
"log",splmodel=splmodel),mustart=stf1$u)}, error = function(e){} )
if (! is.null(design2)){
fit2 <- tryCatch({glm(yobs~design2, family = quasi1(variance = "spline", link = "log",splmodel=splmodel))}, error = function(e){} )
if (is.null(fit2)) fit2 <- tryCatch({glm(yobs~design2, family = quasi1(variance = "spline", link =
"log",splmodel=splmodel),mustart=stf2$u)}, error = function(e){} )
}
if ((!is.null(fit1)) & (!is.null(design2))) {
if (!is.null(fit2)) {
res[j,1] <- ((fit2$deviance-fit1$deviance)/(ncol(design1)-ncol(design2)))/(sum(residuals(fit1,"pearson")^2)/fit1$df.residual)
res[j,2] <- 1-pf(res[j,1], df1=ncol(design1)-ncol(design2),df2=fit1$df.residual)
}
}
if (is.null(design2) & (!is.null(fit1))) {
res[j,1] <- ((fit1$null.deviance-fit1$deviance)/(ncol(design1)-1))/(sum(residuals(fit1,"pearson")^2)/fit1$df.residual)
res[j,2] <- 1-pf(res[j,1], df1=ncol(design1)-1,df2=fit1$df.residual)
}
j <- j + 1
if (j==nrow(data)) break
}
colnames(res)<-c("F","p-value")
return(res)
}

# evaluate performance
paraest <- function(nDE, ngene, p, ng, der){
# nDE - number of DE genes after filtering
# ngene - number of genes after filtering
# p - adjusted p-values
# ng - number of DE genes before filtering, it is 20000 in simulation
```

```
# der - percent of DE genes before filtering, it is 5% in simulation

tp <- sum(p[1:nDE]<0.05,na.rm = TRUE)
fp <- sum(p[(nDE+1):(ngene)]<0.05,na.rm = TRUE)
pfilter <- ng*der-nDE
nfilter <- ng*(1-der) - (ngene-nDE)
tn <- sum(p[(nDE+1):(ngene)]>=0.05,na.rm = TRUE)+ nfilter
fn <- sum(p[1:nDE]>=0.05,na.rm = TRUE) + pfilter
sen <- tp/(tp+fn)
spc <- tn/(tn+fp)
accuracy <- (tp+tn)/(tp+tn+fp+fn)
typ1err <- fp/(tn+fp)
typ2err <- fn/(tp+fn)
FDR <- fp/(tp+fp)
return(round(c(tp,fp,tn,fn,sen,spc,accuracy,FDR),digits=4))
}

# adjust raw p-values, allow na in p-values
getadjp <- function(data,rawp){
  select <- is.nan(rawp) | is.na(rawp)
dset1 <- data[select,]
dset2 <- data[! select,]
  if (nrow(dset1)>0) dset1$adjp <- NA
dset2$adjp <- p.adjust(rawp[! select], method="BH")
dset <- rbind(dset1,dset2)
lbl <- rownames(dset)
lblnum <- as.numeric(substr(lbl,2,max(nchar(lbl))))
dset <- dset[order(lblnum),]
return(dset$adjp)
}

# load libraries for the edgeR method
library(edgeR)
# use the sample means in experimental conditions from Himes data to simulate data
library(airway)
data(airway)
air <- assays(airway)$count
# rearrange by experimental conditions
air <- air[,c(1,3,5,7,2,4,6,8)]
n <- 8
libsize = apply(air[,1:n],2,sum)
# normalized by the TMM method
nrmfactor1 <-calcNormFactors(air[,1:n],method = "TMM")*libsize
air1 <- as.data.frame(t(t(air[,1:n])/nrmfactor1))*mean(nrmfactor1)
# calculate the means in each experiemtnal conditions
air1$mean1 <- apply(air1[,1:(n/2)],1,mean)
air1$mean2 <- apply(air1[,(n/2+1):n],1,mean)
meanest1 <- c(air1$mean1,air1$mean2)
# use only non-zero means
# zero means are dropped since it will produce zero variance by variance function in subsequent simulation
mnair <- meanest1[meanest1>0]

# ======== simulate a dataset with 2 conditions ========

# sample size
n <- 10
# treatment effect
foldchg <- 3
# total number of genes needed
n.genes.need <- 20000
# total number of genes simulated may have those all zero genes
n.genes<- n.genes.need * 1.2
# 5% DE genes
ratio <- 0.05
# treatment variable
trt<-rep(1:2,each=n/2)
# sample means from himes data, same in two experimental conditions
sim.mn<-matrix(sample(mnair,n.genes, replace=F),n.genes,2)
# simulate over-expressed and under-expressed genes
nover <- 0.5*n.genes*ratio
```

```
nunder <- 0.5*n.genes*ratio
sim.mn[1:nover,1]<- sim.mn[1:nover,1]/foldchg
sim.mn[(nover+1):(nover+nunder),1]<-foldchg*sim.mn[(nover+1):(nover+nunder),1]

# Simulate library size factors
a <- sim.mn[,trt]
offset<-2^(rnorm(n,0,.5))
a<-t(t(a)*offset)
# dispersion
disp <- exp(rnorm(n.genes, 0, 1.2))
# variance function
vf <- a+log(a+1)
# simulate gamma random variable
simdat<-round(matrix(rgamma(n.genes*n, a^2/rep(disp,n)/vf, a/rep(disp,n)/vf),n.genes,n))
# drop those all zero genes and keep only number of genes needed
nDEover <- sum(rowSums(simdat[1:(n.genes*ratio/2),1:n])>0)
nDEunder <- sum(rowSums(simdat[(n.genes*ratio/2+1):(n.genes*ratio),1:n])>0)
nNDE <- sum(rowSums(simdat[-(1:(n.genes*ratio)),1:n])>0)
if ((nDEunder>(n.genes.need*ratio/2)) && (nDEover>(n.genes.need*ratio/2)) && (nNDE>(n.genes.need*(1-ratio)))){
                        simdat1 <- simdat[1:(n.genes*ratio/2),]
                        simdat2 <- simdat[(n.genes*ratio/2+1):(n.genes*ratio),]
                        simdat3 <- simdat[-(1:(n.genes*ratio)),]
                        simdat1 <- simdat1[rowSums(simdat1[,1:n])>0,]
                        simdat2 <- simdat2[rowSums(simdat2[,1:n])>0,]
                        simdat3 <- simdat3[rowSums(simdat3[,1:n])>0,]
                        simdat1 <- simdat1[1:(n.genes.need*ratio/2),]
                        simdat2 <- simdat2[1:(n.genes.need*ratio/2),]
                        simdat3 <- simdat3[1:(n.genes.need*(1-ratio)),]
                        simdat <- rbind(simdat1,simdat2,simdat3)
                        simdat <- as.data.frame(simdat)
                        rownames(simdat) <- paste("n",1:n.genes.need,sep="")
}

# analyze the simuated data

# load libraries for DESeq and NBQLSpline
library(DESeq)
library(QuasiSeq)

n <- 10
simdat <- as.data.frame(simdat[,1:n])

 # filter genes with zero IQR and low count
 IQR <- apply(simdat,1,IQR)
 simdat <- simdat[IQR != 0 & rowMeans(simdat)>1,]

# normalized the raw count data by the TMM method
 libsize = apply(simdat[,1:n],2,sum)
 nrmfactor1 <-calcNormFactors(simdat[,1:n],method="TMM")*libsize
 simdatnorm1 <- as.data.frame(t(t(simdat[,1:n])/nrmfactor1))*mean(nrmfactor1)

# ======== QuasiDE ========

# estimate the variance function by smooth cubic spline
 simdatnorm1$mean <- apply(simdatnorm1[,1:n],1,mean)
 simdatnorm1$mean1 <- apply(simdatnorm1[,1:(n/2)],1,mean)
 simdatnorm1$mean2 <- apply(simdatnorm1[,(n/2+1):n],1,mean)
 simdatnorm1$var <- apply(simdatnorm1[,1:n],1,var)
 simdatnorm1$var1 <- apply(simdatnorm1[,1:(n/2)],1,var)
 simdatnorm1$var2 <- apply(simdatnorm1[,(n/2+1):n],1,var)
 meanest1 <- c(simdatnorm1$mean1,simdatnorm1$mean2)
 varest1 <- c(simdatnorm1$var1,simdatnorm1$var2)
 modnorm1 <- smooth.spline(meanest1[varest1 != 0],log(varest1[varest1 != 0]))

# specify model
 dsgn <- model.matrix(~as.factor(trt))
 d1 <- quasiDE(simdatnorm1[,1:n],design1=dsgn,splmodel=modnorm1)
# get raw p-value
 simdat$quasiDE.1.praw <- d1[,2]
# adjusted raw p-values by the BH method
```

```
 simdat$quasiDE.1 <- getadjp(simdat, simdat$quasiDE.1.praw)

# ======== NBQLSpline ========

# put two models to compare into a list object
 design.list<-vector("list",2)
 design.list[[1]]<-model.matrix(~as.factor(trt))
 design.list[[2]]<-rep(1,n)
# specify contrast (model 1 vs. 2 in the list) and give a name
 test.mat <- matrix(1:2,nrow=1,ncol=2)
 row.names(test.mat) <- c("trt")

# GLM NB model with the TMM normalization
 fit1.NB<-QL.fit(as.matrix(simdat[,1:n]),test.mat=test.mat,design.list,log.offset=log(nrmfactor1),Model="NegBin")
# get p-values and adjusted p-values
 results1.NB<-QL.results(fit1.NB,Plot=FALSE)
 simdat$NBQLSPq.trt.1.rawp <- as.numeric(results1.NB$P.values$QLSpline[,"trt"])
 simdat$NBQLSPq.trt.1 <- p.adjust(simdat$NBQLSPq.trt.1.rawp, method="BH")

# ======== edgeR ========

# estimate dispersion
 grp = as.factor(rep(1:2,each=n/2))
 DGE1 = DGEList(simdat[,1:n],lib.size=nrmfactor1,group=grp)
 DGE1 = estimateCommonDisp(DGE1)
 DGE1 = estimateTrendedDisp(DGE1)
 DGE1 = estimateTagwiseDisp(DGE1)
# exact test
 et1 <- exactTest(DGE1)
# processing results and get adjusted p-value
 results1 <- topTags(et1,n=nrow(simdat))$table
 gname1 <- rownames(results1)
 num1 <- as.numeric(substr(rownames(results1),2,max(nchar(gname1))))
 edgeRres1 <- results1[order(num1),]
 simdat$edgeq1 <- edgeRres1$FDR

# ======== DESeq ========

 # estimate dispersion
 dcds1 <- newCountDataSet(simdat[,1:n], grp)
 dcds1 <- estimateSizeFactors(dcds1)
 pData(dcds1)$sizeFactor <- nrmfactor1/mean(nrmfactor1)
 dcds1 <- estimateDispersions(dcds1, method="pooled",sharingMode = "fit-only", fitType="local")
 # statisitcal test
 DESeqres1 <- nbinomTest(dcds1, "1", "2")
 # get results
 simdat$DESeqq1 <- DESeqres1$padj

 # evaluate performance
 # total number of genes before filtering
 ngenes <- 20000
 lblnum <- as.numeric(substr(rownames(simdat),2,max(nchar(rownames(simdat)))))
 # number of truly DE genes in filtered dataset
 numDE <- sum(lblnum <= ngenes*ratio)
 # number of genes in filtered dataset
 nGen <- nrow(simdat)

 # labels
 collbl <- c("tp","fp","tn","fn","sen","spc","accuracy","FDR")
 rowlbl <- c("NBSpline","EdgeR","DESeq","QuasiDE")
 para <- matrix(NA,nrow=4,ncol=8)
 para[1,] <- paraest(numDE,nGen, simdat$NBQLSPq.trt.1,ngenes,ratio)
 para[2,] <- paraest(numDE,nGen, simdat$edgeq1,ngenes,ratio)
 para[3,] <- paraest(numDE,nGen, simdat$DESeqq1,ngenes,ratio)
 para[4,] <- paraest(numDE,nGen, simdat$quasiDE.1,ngenes,ratio)

 rownames(para) <- rowlbl
 colnames(para) <- collbl
 para
```

```
# ============================================================================================
# R.code (Chapter 3) Experiments with multiple conditions
# ============================================================================================


# ======== simulate three groups: each group has 5 samples ========
n <- 15
# number of genes needed in the simulation
n.genes.need <- 20000
# simulated more than needed to discard those not needed (genes with all zeros)
n.genes<- n.genes.need * 1.3
# percent of DE genes
ratio <- 0.05
# treatment effect in three groups
models <- c(1,1.5,3.0)
grp<-rep(1:3,each=n/3)
sim.mn<-matrix(sample(mnair,n.genes, replace=F),n.genes,3)
numDE <- n.genes*ratio
j <- 1
# impose the treatment effect in random order
repeat{
        sim.mn[j,] <- sim.mn[j,]*sample(models)
        j <- j + 1
        if (j==numDE) break
}

### Simulate library size factors
a<-sim.mn[,grp]
offset<-2^(rnorm(n,0,.5))
a<-t(t(a)*offset)

### simulate dispersion
disp <- exp(rnorm(n.genes, 0, 1.2))
# variance function
vf <- a+log(a+1)
# simulate gamma random variable to have the above variance function with dispersion
simdat<-round(matrix(rgamma(n.genes*n, a^2/rep(disp,n)/vf, a/rep(disp,n)/vf),n.genes,n))
# drop genes with all zeros and keep number of genes we need
nDE <- sum(rowSums(simdat[1:numDE,1:n])>0)
nEE <- sum(rowSums(simdat[-(1:numDE),1:n])>0)
if ( (nDE >(n.genes.need*ratio)) && (nEE>(n.genes.need*(1-ratio)))){
        simdat1 <- simdat[1:numDE,]
        simdat2 <- simdat[-(1:numDE),]
        simdat1 <- simdat1[rowSums(simdat1[,1:n])>0,]
        simdat2 <- simdat2[rowSums(simdat2[,1:n])>0,]
        simdat1 <- simdat1[1:(n.genes.need*ratio),]
        simdat2 <- simdat2[1:(n.genes.need*(1-ratio)),]
        simdat <- rbind(simdat1,simdat2)
        simdat <- as.data.frame(simdat)
        rownames(simdat) <- paste("n",1:n.genes.need,sep="")
}


 # analysis begins
 # filter genes with zero IQR and low mean count
 subn <- n/3
 grp<-rep(1:3,each=subn)
 IQR <- apply(simdat[1:n],1,IQR)
 simdat <- simdat[IQR != 0 & rowMeans(simdat)>1,]

 # normalize the data by the TMM method
 libsize = apply(simdat[,1:n],2,sum)
 nrmfactor1 <-calcNormFactors(simdat[,1:n],method="TMM")*libsize
 simdatnorm1 <- as.data.frame(t(t(simdat[,1:n])/nrmfactor1))*mean(nrmfactor1)

# ======== QuasiDE ========

 # Estimate variance function
 simdatnorm1$mean <- apply(simdatnorm1[,1:n],1,mean)
 simdatnorm1$mean1 <- apply(simdatnorm1[,1:(n/3)],1,mean)
```

```
 simdatnorm1$mean2 <- apply(simdatnorm1[,(n/3+1):(2*n/3)],1,mean)
 simdatnorm1$mean3 <- apply(simdatnorm1[,(2*n/3+1):n],1,mean)
 simdatnorm1$var <- apply(simdatnorm1[,1:n],1,var)
 simdatnorm1$var1 <- apply(simdatnorm1[,1:(n/3)],1,var)
 simdatnorm1$var2 <- apply(simdatnorm1[,(n/3+1):(2*n/3)],1,var)
 simdatnorm1$var3 <- apply(simdatnorm1[,(2*n/3+1):n],1,var)
 meanest1 <- c(simdatnorm1$mean1,simdatnorm1$mean2,simdatnorm1$mean3)
 varest1 <- c(simdatnorm1$var1,simdatnorm1$var2,simdatnorm1$var3)
 modnorm1 <- smooth.spline(meanest1[varest1 != 0],log(varest1[varest1 != 0]))

 # two models to be compared
 dsgn.full <- model.matrix(~as.factor(grp))
 dsgn.null <- dsgn.full[,1,drop=F]

d1 <- quasiDE(simdatnorm1[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel=modnorm1)
simdatnorm1$quasiDE.1.praw <- d1[,2]
# adjusted raw p-value by the BH method
simdatnorm1$quasiDE.1 <- getadjp(simdatnorm1,simdatnorm1$quasiDE.1.praw)

# ======== NBQLSpline ========

# set up two models to be compared
 design.list<-vector("list",2)
 design.list[[1]]<-dsgn.full
 design.list[[2]]<-dsgn.null
 test.mat <- rbind(1:2)
 row.names(test.mat) <- c("anova")

 # fit model by NB regression using quasi-likelihood
 fit1.NB<-QL.fit(as.matrix(simdat[,1:n]),test.mat=test.mat,design.list,log.offset=log(nrmfactor1),Model="NegBin")
 # get results
 results1.NB<-QL.results(fit1.NB,Plot=FALSE)
 simdatnorm1$NBQLSPq1.rawp <- as.numeric(results1.NB$P.values$QLSpline[,"anova"])
 simdatnorm1$NBQLSPq1 <- p.adjust(simdatnorm1$NBQLSPq1.rawp,method="BH")

# ======== edgeR ========

 # estimate the gene-specific dispersion
 grp = as.factor(rep(1:3,each=n/3))
 design <- model.matrix(~grp)
 DGE1 = DGEList(simdat[,1:n],group=grp)
 DGE1 = calcNormFactors(DGE1, method="TMM")
 DGE1 <- estimateGLMCommonDisp(DGE1,design)
 DGE1 <- estimateGLMTrendedDisp(DGE1,design)
 DGE1 <- estimateGLMTagwiseDisp(DGE1,design)

 # statistical test using GLM
 fit1 <- glmFit(DGE1,design=design)
 lrt1 <- glmLRT(fit1,coef=2:3)

 # get results
 results1 <- topTags(lrt1,n=nrow(simdat))$table
 gname1 <- rownames(results1)
 num1 <- as.numeric(substr(rownames(results1),2,max(nchar(gname1))))
 edgeRres1 <- results1[order(num1),]
 simdat$edgeq1 <- edgeRres1$FDR

# ======== DESeq ========

 # prepare the dataset
 dcds1 <- newCountDataSet(simdat[,1:n], grp)
 # set up size factor to be TMM
 dcds1 <- estimateSizeFactors(dcds1)
 pData(dcds1)$sizeFactor <- nrmfactor1/mean(nrmfactor1)
 # estimate dispersion
 dcds1 <- estimateDispersions(dcds1, method="pooled",sharingMode = "fit-only", fitType="local")
 # commpare two models
 fit1 <- fitNbinomGLMs(dcds1,count~grp)
 fit0 <- fitNbinomGLMs(dcds1,count~1)
 # statistical test
```

```
pvGLM1 <- nbinomGLMTest(fit1,fit0)
# adjusted p-value by the BH method
padj1.DESeq <- p.adjust(pvGLM1, method="BH")
simdat$DESeqq1 <- padj1.DESeq

lblnum1 <- as.numeric(substr(rownames(simdat),2,max(nchar(rownames(simdat)))))
# number of DE genes in the simulated dataset after filtering
numDE1 <- sum(lblnum1 <= ngenes*ratio)
# number of genes in the simulated dataset after filtering
nGen1 <- nrow(simdat)

# evaluate performance
paralbl <- c("tp","fp","tn","fn","sen","spc","accuracy","FDR")
rowlbl <- c("NBQLSpline","edgeR","DESeq","QuasiDE")
para <- matrix(NA,nrow=length(rowlbl),ncol=length(paralbl))
rownames(para) <- rowlbl
colnames(para) <- paralbl
para[1,] <- paraest(numDE1,nGen1,simdatnorm1$NBQLSPq1,ng=ngenes,der=ratio)
para[2,] <- paraest(numDE1,nGen1,simdat$edgeq1,ng=ngenes,der=ratio)
para[3,] <- paraest(numDE1,nGen1,simdat$DESeqq1,ng=ngenes,der=ratio)
para[4,] <- paraest(numDE1,nGen1,simdatnorm1$quasiDE.1,ng=ngenes,der=ratio)
para

# ============================================================================================
# R.code (Chapter 3) Block Design
# ============================================================================================

# ======== simulate 2 blocks x 2 treatment groups ========
# number of samples
n <- 20
# number of genes needed
n.genes.need <- 20000
# the number of genes simulated more than needed will be dropped
n.genes<- n.genes.need * 1.4
# percent of DE genes
ratio <- 0.05
# percent of genes having block effect
blkratio <- 0.3
repnum <- 200
i <- 1
# effect patterns in simulated data, first 6 for genes with treatment effect, last two for genes without treatment effect
eff <- matrix(NA,nrow=8,ncol=4)
eff[1,] <- c(1,2,2,4)
eff[2,] <- c(2,1,4,2)
eff[3,] <- c(2,4,1,2)
eff[4,] <- c(4,2,2,1)
eff[5,] <- c(1,1,2,2)
eff[6,] <- c(2,2,1,1)
eff[7,] <- c(1,2,1,2) # block effect only
eff[8,] <- c(2,1,2,1) # block effect only

while (i <= repnum){
        # number of simulated genes with treatment effect
        numDE <- n.genes*ratio
        # number of simulated genes with both treatment and block effect
        numblktrt <- n.genes*ratio*blkratio
        # number of simulated genes with only treatment effect
        numtrt <- numDE-numblktrt
        # number of simulated genes with only block effect
        numblk <- n.genes*(1-ratio)*blkratio
        # assign a random mean count to all 4 groups
        grp <- rep(1:4,each=n/4)
        sim.mn <- matrix(sample(mnair,n.genes, replace=F),n.genes,4)
        # apply the designed effects
        sim.mn[1:(numblktrt/4),] <- t(t(sim.mn[1:(numblktrt/4),])*eff[1,])
        sim.mn[(numblktrt/4+1):(numblktrt/2),] <- t(t(sim.mn[(numblktrt/4+1):(numblktrt/2),])*eff[2,])
        sim.mn[(numblktrt/2+1):(numblktrt*3/4),] <- t(t(sim.mn[(numblktrt/2+1):(numblktrt*3/4),])*eff[3,])
        sim.mn[(numblktrt*3/4+1):numblktrt,] <- t(t(sim.mn[(numblktrt*3/4+1):numblktrt,])*eff[4,])
        sim.mn[(numblktrt+1):(numblktrt+numtrt/2),] <- t(t(sim.mn[(numblktrt+1):(numblktrt+numtrt/2),])*eff[5,])
```

```
        sim.mn[(numblktrt+numtrt/2+1):numDE,] <- t(t(sim.mn[(numblktrt+numtrt/2+1):numDE,] )*eff[6,])
        sim.mn[(numDE+1):(numDE+numblk/2),] <- t(t(sim.mn[(numDE+1):(numDE+numblk/2),])*eff[7,])
        sim.mn[(numDE+numblk/2+1):(numDE+numblk),] <- t(t(sim.mn[(numDE+numblk/2+1):(numDE+numblk),])*eff[8,])

        # Simulate library size factors
        a<-sim.mn[,grp]
        offset<-2^(rnorm(n,0,.5))
        a<-t(t(a)*offset)

        # Simulate dispersion
        disp <- exp(rnorm(n.genes, 0, 1.2))
        # variance function
        vf <- a+log(a+1)
        # simulate gamma random variable to have the variance determined by variance function and dispersion
        simdat<-round(matrix(rgamma(n.genes*n, a^2/rep(disp,n)/vf, a/rep(disp,n)/vf),n.genes,n))
        simdat <- cbind(simdat, sim.mn,disp)
        # number of non-all-zero genes in each type of simulated genes
        n1 <- sum(rowSums(simdat[1:(numblktrt/4),1:n])>0)
        n2 <- sum(rowSums(simdat[(numblktrt/4+1):(numblktrt/2),1:n])>0)
        n3 <- sum(rowSums(simdat[(numblktrt/2+1):(numblktrt*3/4),1:n])>0)
        n4 <- sum(rowSums(simdat[(numblktrt*3/4+1):numblktrt,1:n])>0)
        n5 <- sum(rowSums(simdat[(numblktrt+1):(numblktrt+numtrt/2),1:n])>0)
        n6 <- sum(rowSums(simdat[(numblktrt+numtrt/2+1):numDE,1:n])>0)
        n7 <- sum(rowSums(simdat[(numDE+1):(numDE+numblk/2),1:n])>0)
        n8 <- sum(rowSums(simdat[(numDE+numblk/2+1):(numDE+numblk),1:n])>0)
        n9 <- sum(rowSums(simdat[(numDE+numblk+1):n.genes,1:n])>0)
        # number of simulated genes with treatment effect needed
        numDE.need <- n.genes.need*ratio
        # number of simulated genes with treatment and block effect needed
        numblktrt.need <- n.genes.need*ratio*blkratio
        # number of simulated genes with only treatment effect needed
        numtrt.need <- numDE.need-numblktrt.need
        # number of simulated genes with only block effect needed
        numblk.need <- n.genes.need*(1-ratio)*blkratio
        # number of simulated EE genes needed
        numEE.need <- n.genes.need*(1-ratio)
        # remove those genes with all zeros counts and keep the number of genes needed
        if ( (n1>numblktrt.need/4) && (n2>numblktrt.need/4) && (n3>numblktrt.need/4) && (n4>numblktrt.need/4) &&
(n5>numtrt.need/2)
                && (n6>numtrt.need/2) && (n7>numblk.need/2) && (n8>numblk.need/2) && (n9 > numEE.need-numblk.need)){
                simdat1 <- simdat[1:(numblktrt/4),]
                simdat2 <- simdat[(numblktrt/4+1):(numblktrt/2),]
                simdat3 <- simdat[(numblktrt/2+1):(numblktrt*3/4),]
                simdat4 <- simdat[(numblktrt*3/4+1):numblktrt,]
                simdat5 <- simdat[(numblktrt+1):(numblktrt+numtrt/2),]
                simdat6 <- simdat[(numblktrt+numtrt/2+1):numDE,]
                simdat7 <- simdat[(numDE+1):(numDE+numblk/2),]
                simdat8 <- simdat[(numDE+numblk/2+1):(numDE+numblk),]
                simdat9 <- simdat[(numDE+numblk+1):n.genes,]
                simdat1 <- simdat1[rowSums(simdat1[,1:n])>0,]
                simdat2 <- simdat2[rowSums(simdat2[,1:n])>0,]
                simdat3 <- simdat3[rowSums(simdat3[,1:n])>0,]
                simdat4 <- simdat4[rowSums(simdat4[,1:n])>0,]
                simdat5 <- simdat5[rowSums(simdat5[,1:n])>0,]
                simdat6 <- simdat6[rowSums(simdat6[,1:n])>0,]
                simdat7 <- simdat7[rowSums(simdat7[,1:n])>0,]
                simdat8 <- simdat8[rowSums(simdat8[,1:n])>0,]
                simdat9 <- simdat9[rowSums(simdat9[,1:n])>0,]
                simdat1 <- simdat1[1:(numblktrt.need/4),]
                simdat2 <- simdat2[1:(numblktrt.need/4),]
                simdat3 <- simdat3[1:(numblktrt.need/4),]
                simdat4 <- simdat4[1:(numblktrt.need/4),]
                simdat5 <- simdat5[1:(numtrt.need/2),]
                simdat6 <- simdat6[1:(numtrt.need/2),]
                simdat7 <- simdat7[1:(numblk.need/2),]
                simdat8 <- simdat8[1:(numblk.need/2),]
                simdat9 <- simdat9[1:(numEE.need-numblk.need),]
                simdat <- rbind(simdat1,simdat2,simdat3,simdat4,simdat5,simdat6,simdat7,simdat8,simdat9)
                simdat <- as.data.frame(simdat)
                rownames(simdat) <- paste("n",1:n.genes.need,sep="")
```

```
                        datafile<-paste(datadir,dsgn,"-n",n,"-",i,"-",varf,".dat",sep="")
                        write.table(simdat,file=datafile)
                        i <- i + 1
                }
}


# analysis
# treatment group
trt <- rep(1:2,each=2*subn)
# block group
blk <- c(rep(1:2,each=subn),rep(1:2,each=subn)) # block

# genes are filtered by zero IQR and low mean count
IQR <- apply(simdat[1:n],1,IQR)
simdat <- simdat[IQR != 0 & rowMeans(simdat)>1,]

# normalize the data by the TMM method
libsize = apply(simdat[,1:n],2,sum)
nrmfactor1 <-calcNormFactors(simdat[,1:n],method="TMM")*libsize
simdatnorm1 <- as.data.frame(t(t(simdat[,1:n])/nrmfactor1))*mean(nrmfactor1)

# ======== QuasiDE ========

# estimate the variance function
simdatnorm1$mean <- apply(simdatnorm1[,1:n],1,mean)
simdatnorm1$mean1 <- apply(simdatnorm1[,1:(n/4)],1,mean)
simdatnorm1$mean2 <- apply(simdatnorm1[,(n/4+1):(2*n/4)],1,mean)
simdatnorm1$mean3 <- apply(simdatnorm1[,(2*n/4+1):(3*n/4)],1,mean)
simdatnorm1$mean4 <- apply(simdatnorm1[,(3*n/4+1):n],1,mean)
simdatnorm1$var <- apply(simdatnorm1[,1:n],1,var)
simdatnorm1$var1 <- apply(simdatnorm1[,1:(n/4)],1,var)
simdatnorm1$var2 <- apply(simdatnorm1[,(n/4+1):(2*n/4)],1,var)
simdatnorm1$var3 <- apply(simdatnorm1[,(2*n/4+1):(3*n/4)],1,var)
simdatnorm1$var4 <- apply(simdatnorm1[,(3*n/4+1):n],1,var)
meanest1 <- c(simdatnorm1$mean1,simdatnorm1$mean2,simdatnorm1$mean3,simdatnorm1$mean4)
varest1 <- c(simdatnorm1$var1,simdatnorm1$var2,simdatnorm1$var3,simdatnorm1$var4)
modnorm1 <- smooth.spline(meanest1[varest1 != 0],log(varest1[varest1 != 0]))

dsgn.blk <- model.matrix(~as.factor(blk))
dsgn.full <- model.matrix(~as.factor(trt)+as.factor(blk))

d1 <- quasiDE(simdatnorm1[,1:n],design1=dsgn.full, design2=dsgn.blk, splmodel=modnorm1)
simdat$quasiDE.1.praw <- d1[,2]
# adjusted raw-p by the BH method
simdat$quasiDE.1 <- getadjp(simdat,simdat$quasiDE.1.praw)

# ======== NBQLSpline ========

# specify the models to be compared
design.list<-vector("list",2)
design.list[[1]]<-model.matrix(~as.factor(trt)+as.factor(blk))
design.list[[2]]<-model.matrix(~as.factor(blk))
test.mat <- rbind(1:2)
row.names(test.mat) <- "trt"
# fit GLM using quasi-likelihood assuming NB
fit1.NB<-QL.fit(as.matrix(simdat[,1:n]),test.mat=test.mat,design.list,log.offset=log(nrmfactor1),Model="NegBin")
# get results
results1.NB<-QL.results(fit1.NB,Plot=FALSE)
simdat$NBQLSPq.trt.1.rawp <- as.numeric(results1.NB$P.values$QLSpline[,"trt"])
simdat$NBQLSPq.trt.1 <- p.adjust(simdat$NBQLSPq.trt.1.rawp,method="BH")

# ======== edgeR ========

# Specify the full model and prepare the data
design <- model.matrix(~as.factor(trt)+as.factor(blk))
grp = as.factor(rep(1:4,each=n/4))
DGE1 = DGEList(simdat[,1:n],group=grp)
# normalize by the TMM method
DGE1 = calcNormFactors(DGE1, method="TMM")
```

```
# estimate dispersion
DGE1 <- estimateGLMCommonDisp(DGE1,design)
DGE1 <- estimateGLMTrendedDisp(DGE1,design)
DGE1 <- estimateGLMTagwiseDisp(DGE1,design)
# statistical test of the treatment effect
fit1 <- glmFit(DGE1,design=design)
lrt.trt.1 <- glmLRT(fit1,coef=2)
# get the results
results1.trt <- topTags(lrt.trt.1,n=nrow(simdat))$table
gname1.trt <- rownames(results1.trt)
num1.trt <- as.numeric(substr(rownames(results1.trt),2,max(nchar(gname1.trt))))
edgeRres1.trt <- results1.trt[order(num1.trt),]
simdat$edgeq1.trt <- edgeRres1.trt$FDR

# ======== DESeq ========

# specify the factors in the models
desn <- data.frame(trt=as.factor(trt),blk=as.factor(blk))
dcds1 <- newCountDataSet(simdat[,1:n], desn)
# assign the TMM normalized factor
dcds1 <- estimateSizeFactors(dcds1)
pData(dcds1)$sizeFactor <- nrmfactor1/mean(nrmfactor1)
# estimate the dispersion
dcds1 <- estimateDispersions(dcds1, method="pooled",sharingMode = "fit-only", fitType="local")
# compare two models
fit1 <- fitNbinomGLMs(dcds1,count~trt+blk)
fit0 <- fitNbinomGLMs(dcds1,count~blk)
pvGLM1.trt <- nbinomGLMTest(fit1,fit0)
# adjust raw p-values
padj1.DESeq.trt <- p.adjust(pvGLM1.trt, method="BH")
simdat$DESeqq1.trt <- padj1.DESeq.trt

# number of genes before filtering
ngenes <- 20000
# calculate the number of DE genes after filtering
lblnum1 <- as.numeric(substr(rownames(simdat),2,max(nchar(rownames(simdat)))))
numDE1 <- sum(lblnum1 <= ngenes*ratio)
# the number of genes in the data after filtering
nGen1 <- nrow(simdatnorm1)

# evaluate performance
paralbl <- c("tp","fp","tn","fn","sen","spc","accuracy","FDR")
rowlbl <- c("NBQLSpline","edgeR","DESeq","QuasiDE")
para <- matrix(NA,nrow=length(rowlbl),ncol=length(paralbl))
rownames(para) <- rowlbl
colnames(para) <- paralbl
para[1,] <- paraest(numDE1,nGen1,simdat$NBQLSPq.trt.1 ,ng=ngenes,der=ratio)
para[2,] <- paraest(numDE1,nGen1,simdat$edgeq1.trt,ng=ngenes,der=ratio)
para[3,] <- paraest(numDE1,nGen1,simdat$DESeqq1.trt,ng=ngenes,der=ratio)
para[4,] <- paraest(numDE1,nGen1,simdat$quasiDE.1,ng=ngenes,der=ratio)
para

# ============================================================================================
# R.code (Chapter 3) Factorial Design
# ============================================================================================
# ======== simulate 2x2 factorial design data ========
# number of samples
n <- 20
# number of genes needed
n.genes.need <- 20000
# number of genes simulated
n.genes<- n.genes.need * 1.3
# 5% DE genes
ratio <- 0.05

# the effect models for the DE genes
models1 <- models2 <- matrix(NA, nrow=4,ncol=4)
# main effect and interaction
models1[1,] <- c(1,1.5,2,4)
models1[2,] <- c(1.5,1,4,2)
```

```
models1[3,] <- c(4,2,1.5,1)
models1[4,] <- c(2,4,1,1.5)
# main effects from two factors
models2[1,] <- c(1,1.5,2,3)
models2[2,] <- c(1.5,1,3,2)
models2[3,] <- c(3,2,1.5,1)
models2[4,] <- c(2,3,1,1.5)
models3 <- models4 <- matrix(NA, nrow=2,ncol=4)
# main effect from one factor
models3[1,] <- c(1,1.5,1,1.5)
models3[2,] <- c(1.5,1,1.5,1)
models4[1,] <- c(1,1,2,2)
models4[2,] <- c(2,2,1,1)

# number of genes simulated
numDE <- n.genes*ratio
# two factors each with two levels = 4 groups
grp<-rep(1:4,each=n/4)
# sample the mean from the himes data and assign to 4 groups
sim.mn <- matrix(sample(mnair, n.genes, replace=F),n.genes,4)
# impose the treatment effect models to the DE genes
j <- 1
repeat{
        if (j<=numDE) sim.mn[j,] <- sim.mn[j,]* models1[sample(1:4,1),]
        else if (j<=2*numDE) sim.mn[j,] <- sim.mn[j,]* models2[sample(1:4,1),]
        else if (j<=3*numDE) sim.mn[j,] <- sim.mn[j,]* models3[sample(1:2,1),]
        else if (j<=4*numDE) sim.mn[j,] <- sim.mn[j,]* models4[sample(1:2,1),]
        if (j==4*numDE) break
        j <- j + 1
}

# Simulate library size factors
a<-sim.mn[,grp]
offset<-2^(rnorm(n,0,.5))
a<-t(t(a)*offset)

# simulate dispersion
disp <- exp(rnorm(n.genes, 0, 1.2))
# variance function
vf <- a+log(a+1)
# simulate rounded gamma random variable with the variance determined by variance function and dispersion
simdat<-round(matrix(rgamma(n.genes*n, a^2/rep(disp,n)/vf, a/rep(disp,n)/vf),n.genes,n))
# remove those genes with all zero counts and keep only the number of genes needed
nDE1 <- sum(rowSums(simdat[1:numDE,1:n])>0)
nDE2 <- sum(rowSums(simdat[(numDE+1):(2*numDE),1:n])>0)
nDE3 <- sum(rowSums(simdat[(2*numDE+1):(3*numDE),1:n])>0)
nDE4 <- sum(rowSums(simdat[(3*numDE+1):(4*numDE),1:n])>0)
nEE <- sum(rowSums(simdat[-(1:(4*numDE)),1:n])>0)
crt <- n.genes.need*ratio
eecrt <- n.genes.need*(1-4*ratio)
if ( (nDE1 > crt) && (nDE2 > crt) && (nDE3 > crt) && (nDE4 > crt) && (nEE>eecrt)){
        simdat1 <- simdat[1:numDE,]
        simdat2 <- simdat[(numDE+1):(2*numDE),]
        simdat3 <- simdat[(2*numDE+1):(3*numDE),]
        simdat4 <- simdat[(3*numDE+1):(4*numDE),]
        simdat5 <- simdat[-(1:(4*numDE)),]
        simdat1 <- simdat1[rowSums(simdat1[,1:n])>0,]
        simdat2 <- simdat2[rowSums(simdat2[,1:n])>0,]
        simdat3 <- simdat3[rowSums(simdat3[,1:n])>0,]
        simdat4 <- simdat4[rowSums(simdat4[,1:n])>0,]
        simdat5 <- simdat5[rowSums(simdat5[,1:n])>0,]
        simdat1 <- simdat1[1:crt,]
        simdat2 <- simdat2[1:crt,]
        simdat3 <- simdat3[1:crt,]
        simdat4 <- simdat4[1:crt,]
        simdat5 <- simdat5[1:eecrt,]
        simdat <- rbind(simdat1,simdat2,simdat3,simdat4,simdat5)
        simdat <- as.data.frame(simdat)
        rownames(simdat) <- paste("n",1:n.genes.need,sep="")
}
```

```
# analysis

n <- 20
subn <- n/4
grp<-rep(1:4,each=subn)
# treatment factor 1 and 2
trt1 <- rep(1:2,each=2*subn)
trt2 <- c(rep(1:2,each=subn),rep(1:2,each=subn))

# genes with zero IQR and low mean count are filtered
IQR <- apply(simdat[1:n],1,IQR)
simdat <- simdat[IQR != 0 & rowMeans(simdat)>1,]

# normalize data by the TMM method
libsize = apply(simdat[,1:n],2,sum)
nrmfactor1 <-calcNormFactors(simdat[,1:n],method="TMM")*libsize
simdatnorm1 <- as.data.frame(t(t(simdat[,1:n])/nrmfactor1))*mean(nrmfactor1)

# ======== QuasiDE ========

# estimate the variance function
simdatnorm1$mean <- apply(simdatnorm1[,1:n],1,mean)
simdatnorm1$mean1 <- apply(simdatnorm1[,1:(n/4)],1,mean)
simdatnorm1$mean2 <- apply(simdatnorm1[,(n/4+1):(2*n/4)],1,mean)
simdatnorm1$mean3 <- apply(simdatnorm1[,(2*n/4+1):(3*n/4)],1,mean)
simdatnorm1$mean4 <- apply(simdatnorm1[,(3*n/4+1):n],1,mean)
simdatnorm1$var <- apply(simdatnorm1[,1:n],1,var)
simdatnorm1$var1 <- apply(simdatnorm1[,1:(n/4)],1,var)
simdatnorm1$var2 <- apply(simdatnorm1[,(n/4+1):(2*n/4)],1,var)
simdatnorm1$var3 <- apply(simdatnorm1[,(2*n/4+1):(3*n/4)],1,var)
simdatnorm1$var4 <- apply(simdatnorm1[,(3*n/4+1):n],1,var)
meanest1 <- c(simdatnorm1$mean1,simdatnorm1$mean2,simdatnorm1$mean3,simdatnorm1$mean4)
varest1 <- c(simdatnorm1$var1,simdatnorm1$var2,simdatnorm1$var3,simdatnorm1$var4)
modnorm1 <- smooth.spline(meanest1[varest1 != 0],log(varest1[varest1 != 0]))

# Compare models to test interaction
dsgn.full <- model.matrix(~as.factor(trt1)+as.factor(trt2)+as.factor(trt1):as.factor(trt2))
dsgn.main <- model.matrix(~as.factor(trt1)+as.factor(trt2))

d1 <- quasiDE(simdatnorm1[,1:n],design1=dsgn.full, design2=dsgn.main, splmodel=modnorm1)
simdat$quasiDE.int.1.praw <- d1[,2]
# adjusted raw-p by the BH method
simdat$quasiDE.int.1 <- getadjp(simdat,simdat$quasiDE.int.1.praw)

# ======== NBQLSpline ========

# Specify the models to be compared
design.list<-vector("list",2)
design.list[[1]]<-model.matrix(~as.factor(trt1)+as.factor(trt2)+as.factor(trt1):as.factor(trt2))
design.list[[2]]<-model.matrix(~as.factor(trt1)+as.factor(trt2))
test.mat <- rbind(1:2)
row.names(test.mat) <- c("int")
# fit GLM by quasi-likelihood under NB assumption
fit1.NB<-QL.fit(as.matrix(simdat[,1:n]),test.mat=test.mat,design.list,log.offset=log(nrmfactor1),Model="NegBin")
# get results
results1.NB<-QL.results(fit1.NB,Plot=FALSE)
simdat$NBQLSPq.int.1.rawp <- as.numeric(results1.NB$P.values$QLSpline[,"int"])
simdat$NBQLSPq.int.1 <- p.adjust(simdat$NBQLSPq.int.1.rawp,method="BH")

# ======== edgeR ========

# specify the model
design <- model.matrix(~as.factor(trt1)+as.factor(trt2)+as.factor(trt1):as.factor(trt2))
grp = as.factor(rep(1:4,each=n/4))
DGE1 = DGEList(simdat[,1:n],group=grp)

# normalize the data by the TMM method
DGE1 = calcNormFactors(DGE1, method="TMM")
# estimate dispersion
```

```
DGE1 <- estimateGLMCommonDisp(DGE1,design)
DGE1 <- estimateGLMTrendedDisp(DGE1,design)
DGE1 <- estimateGLMTagwiseDisp(DGE1,design)
# test the interaction
fit1 <- glmFit(DGE1,design=design)
lrt.int.1 <- glmLRT(fit1,coef=4)
# get results
results1.int <- topTags(lrt.int.1,n=nrow(simdat))$table
gname1.int <- rownames(results1.int)
num1.int <- as.numeric(substr(rownames(results1.int),2,max(nchar(gname1.int))))
edgeRres1.int <- results1.int[order(num1.int),]
simdat$edgeq1.int <- edgeRres1.int$FDR

# ======== DESeq ========

# model factors
desn <- data.frame(trt1=as.factor(trt1),trt2=as.factor(trt2))
dcds1 <- newCountDataSet(simdat[,1:n], desn)
dcds1 <- estimateSizeFactors(dcds1)
# assign TMM normalized factor as size factor
pData(dcds1)$sizeFactor <- nrmfactor1/mean(nrmfactor1)
# estimate dispersion
dcds1 <- estimateDispersions(dcds1, method="pooled",sharingMode = "fit-only", fitType="local")
# compare two models by the likelihood ratio test
fit0 <- fitNbinomGLMs(dcds1,count~trt1+trt2+trt1:trt2)
fit1 <- fitNbinomGLMs(dcds1,count~trt1+trt2)
pvGLM1.int <- nbinomGLMTest(fit0,fit1)
# adjust p-values
padj1.DESeq.int <- p.adjust(pvGLM1.int, method="BH")
simdat$DESeqq1.int <- padj1.DESeq.int

# number of genes in the data before filtering
ngenes <- 20000
lblnum1 <- as.numeric(substr(rownames(simdat),2,max(nchar(rownames(simdat)))))
# number of DE genes after filtering
numDE1 <- sum(lblnum1 <= nd)
# number of genes after filtering
nGen1 <- nrow(simdat)

# evaluate performance
paralbl <- c("tp","fp","tn","fn","sen","spc","accuracy","FDR")
rowlbl <- c("NBQLSpline","edgeR","DESeq","QuasiDE")
para <- matrix(NA,nrow=length(rowlbl),ncol=length(paralbl))
rownames(para) <- rowlbl
colnames(para) <- paralbl
para[1,] <- paraest(numDE1,nGen1, simdat$NBQLSPq.int.1 ,ng=ngenes,der=ratio)
para[2,] <- paraest(numDE1,nGen1, simdat$edgeq1.int ,ng=ngenes,der=ratio)
para[3,] <- paraest(numDE1,nGen1, simdat$DESeqq1.int,ng=ngenes,der=ratio)
para[4,] <- paraest(numDE1,nGen1, simdat$quasiDE.int.1 ,ng=ngenes,der=ratio)
para

# ============================================================================================
# R.code (Chapter 4)
# ============================================================================================

# ======== simulate data with two experiment conditions, gene length varied with experimental coditions ========

# load gene length dataset (not provided)
load("c:/genlen.RData")
# sample size
n <- 10
# treatment effect
foldchg <- 3
# number of genes needed
n.genes.need <- 20000
# nnumber of genes simulated
n.genes<- n.genes.need * 1.2
# 5% DE genes
ratio <- 0.05
# treatment group
```

```
trt<-rep(1:2,each=n/2)
# sample means from himes data
sim.mn<-matrix(sample(mnair,n.genes, replace=F),n.genes,2)
# half DE genes over-expressed, half under-expressed
nover <- 0.5*n.genes*ratio
nunder <- 0.5*n.genes*ratio
# setup treatment effect
sim.mn[1:nover,1]<- sim.mn[1:nover,1]/foldchg
sim.mn[(nover+1):(nover+nunder),1]<-foldchg*sim.mn[(nover+1):(nover+nunder),1]

# Simulate library size factors
a <- sim.mn[,trt]
offset<-2^(rnorm(n,0,.5))
a<-t(t(a)*offset)

# varied gene length effect
# sample gene length
glen <- sample(genlen)
glenraw <- glen[1:n.genes]
# set up the gene length in the other treatment group
# fixed gene length: gvlen <- glenraw
gvlen <- ceiling(2^(rnorm(n.genes,0,1/3))*glenraw)
glen <- cbind(glenraw,gvlen)
# apply gene length effect
a <- a*glen[,trt]/1000
# simulate dispersion
disp <- exp(rnorm(n.genes, 0, 1.2))
# variance function
vf <- a+log(a+1)
# simuate gamma random variable having the designed variance and then rounded
simdat<-round(matrix(rgamma(n.genes*n, a^2/rep(disp,n)/vf, a/rep(disp,n)/vf),n.genes,n))
# remove those genes with all zero and keep the number of genes needed
# the corresponding gene length is also kept for future use
nDEover <- sum(rowSums(simdat[1:(n.genes*ratio/2),1:n])>0)
nDEunder <- sum(rowSums(simdat[(n.genes*ratio/2+1):(n.genes*ratio),1:n])>0)
nNDE <- sum(rowSums(simdat[-(1:(n.genes*ratio)),1:n])>0)
if ((nDEunder>(n.genes.need*ratio/2)) && (nDEover>(n.genes.need*ratio/2)) && (nNDE>(n.genes.need*(1-ratio)))){
        simdat1 <- simdat[1:(n.genes*ratio/2),]
        glen1 <- glen[1:(n.genes*ratio/2),]
        simdat2 <- simdat[(n.genes*ratio/2+1):(n.genes*ratio),]
        glen2 <- glen[(n.genes*ratio/2+1):(n.genes*ratio),]
        simdat3 <- simdat[-(1:(n.genes*ratio)),]
        glen3 <- glen[-(1:(n.genes*ratio)),]
        ind1 <- rowSums(simdat1[,1:n])>0
        ind2 <- rowSums(simdat2[,1:n])>0
        ind3 <- rowSums(simdat3[,1:n])>0
        simdat1 <- simdat1[ind1,]
        glen1<-glen1[ind1,]
        simdat2 <- simdat2[ind2,]
        glen2<-glen2[ind2,]
        simdat3 <- simdat3[ind3,]
        glen3<-glen3[ind3,]
        simdat1 <- simdat1[1:(n.genes.need*ratio/2),]
        glen1<-glen1[1:(n.genes.need*ratio/2),]
        simdat2 <- simdat2[1:(n.genes.need*ratio/2),]
        glen2<-glen2[1:(n.genes.need*ratio/2),]
        simdat3 <- simdat3[1:(n.genes.need*(1-ratio)),]
        glen3<-glen3[1:(n.genes.need*(1-ratio)),]
        simdat <- rbind(simdat1,simdat2,simdat3)
        gene.length <- rbind(glen1,glen2,glen3)
        simdat <- as.data.frame(simdat)
        simdat$gl <- gene.length
        rownames(simdat) <- paste("n",1:n.genes.need,sep="")
}

# analysis

# R package for LIMMA
 library(limma)
 library(edgeR)
```

```
 # load R package which has the normalization functions for microarray data
 library(affy)

# produce the RPKM data
 glen <- simdat[,-(1:n)]
 glen <- glen[,trt]
 libsize <- apply(simdat[,1:n],2,sum)
 simdatRPKM <- as.data.frame(t(t(simdat[,1:n])/libsize))/glen*(10^9)

# filter data by zero IQR and low mean count for raw count data, keep the corresponding  gene lengths
 IQR1 <- apply(simdat[,1:n],1,IQR)
 simdatglen <- glen[IQR1 != 0 & rowMeans(simdat)>1,]
 simdat <- simdat[IQR1 != 0 & rowMeans(simdat)>1,]

 # filter data by zero IQR for the RPKM data, keep the corresponding  gene lengths
 IQR2 <- apply(simdatRPKM[,1:n],1,IQR)
 simdatRPKMglen <- glen[IQR2 != 0,]
 simdatRPKM <- simdatRPKM[IQR2 != 0,]

# normalize raw count data by the gene length and the TMM method
simdatgl  <- simdat/simdatglen*10^3
libsizegl <- apply(simdatgl [,1:n],2,sum)
nrmfactorgl <- calcNormFactors(simdatgl[,1:n],method = "TMM")* libsizegl
simdatglTMM <- as.data.frame(t(t(simdatgl[,1:n])/ nrmfactorgl))*mean(nrmfactorgl)

# normalize raw count data by the TMM method
 nrmfactorTMM1 <-calcNormFactors(simdat[,1:n],method="TMM")*libsize
 simdatTMM <- as.data.frame(t(t(simdat[,1:n])/nrmfactorTMM1))*mean(nrmfactorTMM1)

 # normalize the RPKM data by the TMM method
 nrmfactorTMM2 <-calcNormFactors(simdatRPKM[,1:n],method="TMM")
 simdatRPKMTMM <- as.data.frame(t(t(simdatRPKM[,1:n])/nrmfactorTMM2))

# normalize raw count data by the RLE method
 nrmfactorRLE1 <-calcNormFactors(simdat[,1:n],method="RLE")*libsize
 simdatRLE <-as.data.frame(t(t(simdat[,1:n])/nrmfactorRLE1))*mean(nrmfactorRLE1)

# normalize the RPKM data by the RLE method
 nrmfactorRLE2 <-calcNormFactors(simdatRPKM[,1:n],method="RLE")
 simdatRPKMRLE <-as.data.frame(t(t(simdatRPKM[,1:n])/nrmfactorRLE2))

# normalize raw count data by the UQ method
 nrmfactorUQ1 <-calcNormFactors(simdat[,1:n],method="upperquartile")*libsize
 simdatUQ <- as.data.frame(t(t(simdat[,1:n])/nrmfactorUQ1))*mean(nrmfactorUQ1)

 # normalize the RPKM data by the UQ method
 nrmfactorUQ2 <-calcNormFactors(simdatRPKM[,1:n],method="upperquartile")
 simdatRPKMUQ <- as.data.frame(t(t(simdatRPKM[,1:n])/nrmfactorUQ2))

# scale normalization for the RPKM data
 simdatRPKMscale <- as.data.frame(normalizeBetweenArrays(as.matrix(simdatRPKM[,1:n]),method = "scale"))
 rownames(simdatRPKMscale) <- rownames(simdatRPKM)

# quantile normalization for the RPKM data
 simdatRPKMqn <- as.data.frame(normalizeBetweenArrays(as.matrix(simdatRPKM[,1:n]),method = "quantile"))
 rownames(simdatRPKMqn) <- rownames(simdatRPKM)

# cyclic loess normalization for the RPKM data
 simdatRPKMcl <- as.data.frame(normalizeBetweenArrays(as.matrix(simdatRPKM[,1:n]),method = "cyclicloess"))
 rownames(simdatRPKMcl) <- rownames(simdatRPKM)

# invariantset normalization for the RPKM data

 simdatRPKMIV <- simdatRPKM[,1:n]
 ref <- simdatRPKM[,1]
 for (m in 2:n){
            x <- simdatRPKM[,m]
            tmp <- normalize.invariantset(x, ref)
            simdatRPKMIV[,m]<- as.numeric(approx(tmp$n.curve$y, tmp$n.curve$x, xout = x, rule = 2)$y)
 }
```

```
# analysis

# ======== LIMMA ========

# model specification
design<-model.matrix(~as.factor(trt))
colnames(design) <- c("ref","beta")

# analyze the raw data with the gene length and TMM normalization
fit <- lmFit(simdatglTMM[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatglTMM))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatglTMM$lq.gl.TMM <- lres$adj.P.Val
LIMMA.gl.TMM <- lres

# analyze the RPKM data directly
fit <- lmFit(simdatRPKM[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRPKM))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatRPKM$lq.raw <- lres$adj.P.Val
LIMMA.RPKM <- lres

# analyze the TMM normalized count data
fit <- lmFit(simdatTMM[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatTMM))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdat$lq.TMM <- lres$adj.P.Val
LIMMA.TMM <- lres

# analyze the TMM normalized RPKM data
fit <- lmFit(simdatRPKMTMM[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRPKMTMM))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatRPKM$lq.TMM <- lres$adj.P.Val
LIMMA.RPKM.TMM <- lres

# analyze the RLE normalized count data
fit <- lmFit(simdatRLE[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRLE))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdat$lq.RLE <- lres$adj.P.Val
LIMMA.RLE <- lres

# analyze the RLE normalized RPKM data
fit <- lmFit(simdatRPKMRLE[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRPKMRLE))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatRPKM$lq.RLE <- lres$adj.P.Val
LIMMA.RPKM.RLE <- lres

# analyze the UQ normalized count data
fit <- lmFit(simdatUQ[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatUQ))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdat$lq.UQ <- lres$adj.P.Val
```

```
LIMMA.UQ <- lres

# analyze the UQ normalized RPKM data
fit <- lmFit(simdatRPKMUQ[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRPKMUQ))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatRPKM$lq.UQ <- lres$adj.P.Val
LIMMA.RPKM.UQ <- lres

# analyze the scaling normalized RPKM data
fit <- lmFit(simdatRPKMscale[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRPKMscale))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatRPKMscale$lq.scale <- lres$adj.P.Val
LIMMA.RPKM.scale <- lres

# analyze the quantile normalized RPKM data
fit <- lmFit(simdatRPKMqn[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRPKMqn))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatRPKMqn$lq.qn <- lres$adj.P.Val
LIMMA.RPKM.qn <- lres

# analyze the cyclic loess normalized RPKM data
fit <- lmFit(simdatRPKMcl[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRPKMcl))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatRPKMcl$lq.cl <- lres$adj.P.Val
LIMMA.RPKM.cl <- lres

# analyze the invariant set normalized RPKM data
fit <- lmFit(simdatRPKMIV[,1:n], design)
fit <- eBayes(fit)
lres <- topTable(fit, coef ="beta",adjust = "fdr",number=nrow(simdatRPKMIV))
lnum <- as.numeric(gsub("n","",rownames(lres)))
lres <- lres[order(lnum),]
simdatRPKMIV$lq.IV <- lres$adj.P.Val
LIMMA.RPKM.IV <- lres

# ======== edgeR ========

# analyze the raw count using the TMM normalization by adjusting for gene length
DGE1 <- DGEList(counts=simdat[,1:n],lib.size=libsize,group=trt,norm.factors=as.numeric(nrmfactorTMM1))

# Column correct log gene lengths
# Columns of gl should add to zero
gl <- log(glen[IQR1 !=0,])
gl <- t(t(gl)-colMeans(gl))

# Combine library sizes, norm factors and gene lengths:
offset <- expandAsMatrix(getOffset(DGE1))
offset = sweep(gl,2,offset,"+")
DGE1$offset <- offset
DGE1 <- estimateGLMCommonDisp(DGE1, design)
DGE1 <- estimateGLMTrendedDisp(DGE1, design)
DGE1 <- estimateGLMTagwiseDisp(DGE1, design,trend=T)
fit1 <- glmFit(DGE1,design)
results1 <- glmLRT(fit1,coef=2)
edgeres <- topTags(results1,n=nrow(simdat))$table
gname1 <- rownames(edgeres)
num1 <- as.numeric(substr(rownames(edgeres),2,max(nchar(gname1))))
res1 <- edgeres[order(num1),]
```

```
#analyze the raw count using the RLE normalization by adjusting for gene length
DGE2 <- DGEList(counts=simdat[,1:n],lib.size=libsize,group=trt,norm.factors=as.numeric(nrmfactorRLE1))

# Column correct log gene lengths
# Columns of gl should add to zero
gl <- log(glen[IQR1 !=0,])
gl <- t(t(gl)-colMeans(gl))

# Combine library sizes, normalization factors and gene lengths:
offset <- expandAsMatrix(getOffset(DGE2))
offset = sweep(gl,2,offset,"+")
DGE2$offset <- offset
DGE2 <- estimateGLMCommonDisp(DGE2, design)
DGE2 <- estimateGLMTrendedDisp(DGE2, design)
DGE2 <- estimateGLMTagwiseDisp(DGE2,design,trend=T)
fit2 <- glmFit(DGE2,design)
results2 <- glmLRT(fit2,coef=2)
edgeres <- topTags(results2,n=nrow(simdat))$table
gname2 <- rownames(edgeres)
num2 <- as.numeric(substr(rownames(edgeres),2,max(nchar(gname2))))
res2 <- edgeres[order(num2),]

# analyze the raw count using the UQ normalization by adjusting for gene length
DGE3 <- DGEList(counts=simdat[,1:n],lib.size=libsize,group=trt,norm.factors=as.numeric(nrmfactorUQ1))

# Column correct log gene lengths
# Columns of gl should add to zero
gl <- log(glen[IQR1 != 0, ])
gl <- t(t(gl)-colMeans(gl))

# Combine library sizes, normalization factors and gene lengths:
offset <- expandAsMatrix(getOffset(DGE3))
offset = sweep(gl,2,offset,"+")
DGE3$offset <- offset
DGE3 <- estimateGLMCommonDisp(DGE3, design)
DGE3 <- estimateGLMTrendedDisp(DGE3, design)
DGE3 <- estimateGLMTagwiseDisp(DGE3,design,trend=T)
fit3 <- glmFit(DGE3,design)
results3 <- glmLRT(fit3,coef=2)
edgeres <- topTags(results3,n=nrow(simdat))$table
gname3 <- rownames(edgeres)
num3 <- as.numeric(substr(rownames(edgeres),2,max(nchar(gname3))))
res3 <- edgeres[order(num3),]

simdat$edgeq.TMM <- res1$FDR
simdat$edgeq.RLE <- res2$FDR
simdat$edgeq.UQ <- res3$FDR
edgeRres.TMM <- res1
edgeRres.RLE <- res2
edgeRres.UQ <- res3

# ======== QuasiDE ========

# analyze the raw count data normalized by the gene length and the TMM method
simdatglTMM$mean <- apply(simdatglTMM[,1:n],1,mean)
simdatglTMM$mean1 <- apply(simdatglTMM[,1:(n/2)],1,mean)
simdatglTMM$mean2 <- apply(simdatglTMM[,(n/2+1):n],1,mean)
simdatglTMM$var <- apply(simdatglTMM[,1:n],1,var)
simdatglTMM$var1 <- apply(simdatglTMM[,1:(n/2)],1,var)
simdatglTMM$var2 <- apply(simdatglTMM[,(n/2+1):n],1,var)
meanest <- c(simdatglTMM$mean1,simdatglTMM$mean2)
varest <- c(simdatglTMM$var1,simdatglTMM$var2)
modnormglTMM <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
dsgn.full <- model.matrix(~as.factor(trt))
dsgn.null <- dsgn.full[,1,drop=F]
d1 <- quasiDE(simdatglTMM[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel=modnormglTMM)
simdatglTMM$QDE.gl.TMM.praw <- d1[,2]
# adjusted raw-p by the BH method
simdatglTMM$quasiDE.gl.TMM <- getadjp(simdatglTMM, simdatglTMM$QDE.gl.TMM.praw)
```

```
# analyze the RPKM data
simdatRPKM$mean <- apply(simdatRPKM[,1:n],1,mean)
simdatRPKM$mean1 <- apply(simdatRPKM[,1:(n/2)],1,mean)
simdatRPKM$mean2 <- apply(simdatRPKM[,(n/2+1):n],1,mean)
simdatRPKM$var <- apply(simdatRPKM[,1:n],1,var)
simdatRPKM$var1 <- apply(simdatRPKM[,1:(n/2)],1,var)
simdatRPKM$var2 <- apply(simdatRPKM[,(n/2+1):n],1,var)
meanest <- c(simdatRPKM$mean1,simdatRPKM$mean2)
varest <- c(simdatRPKM$var1,simdatRPKM$var2)
modnormRPKM <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
dsgn.full <- model.matrix(~as.factor(trt))
dsgn.null <- dsgn.full[,1,drop=F]
d1 <- quasiDE(simdatRPKM [,1:n],design1=dsgn.full, design2=dsgn.null, splmodel=modnormRPKM)
simdatRPKM$QDE.praw <- d1[,2]
# adjusted raw p-values by the BH method
simdatRPKM$quasiDE <- getadjp(simdatRPKM, simdatRPKM$QDE.praw)

# analyze the TMM normalized raw count data
simdatTMM$mean <- apply(simdatTMM[,1:n],1,mean)
simdatTMM$mean1 <- apply(simdatTMM[,1:(n/2)],1,mean)
simdatTMM$mean2 <- apply(simdatTMM[,(n/2+1):n],1,mean)
simdatTMM$var <- apply(simdatTMM[,1:n],1,var)
simdatTMM$var1 <- apply(simdatTMM[,1:(n/2)],1,var)
simdatTMM$var2 <- apply(simdatTMM[,(n/2+1):n],1,var)
meanest <- c(simdatTMM$mean1,simdatTMM$mean2)
varest <- c(simdatTMM$var1,simdatTMM$var2)
modnormTMM <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
d1 <- quasiDE(simdatTMM[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel=modnormTMM)
simdatTMM$QDE.praw <- d1[,2]
simdatTMM$quasiDE <- getadjp(simdatTMM, simdatTMM$QDE.praw)

# analyze the TMM normalized RPKM data
simdatRPKMTMM$mean <- apply(simdatRPKMTMM[,1:n],1,mean)
simdatRPKMTMM$mean1 <- apply(simdatRPKMTMM[,1:(n/2)],1,mean)
simdatRPKMTMM$mean2 <- apply(simdatRPKMTMM[,(n/2+1):n],1,mean)
simdatRPKMTMM$var <- apply(simdatRPKMTMM[,1:n],1,var)
simdatRPKMTMM$var1 <- apply(simdatRPKMTMM[,1:(n/2)],1,var)
simdatRPKMTMM$var2 <- apply(simdatRPKMTMM[,(n/2+1):n],1,var)
meanest <- c(simdatRPKMTMM$mean1,simdatRPKMTMM$mean2)
varest <- c(simdatRPKMTMM$var1,simdatRPKMTMM$var2)
modnormRPKMTMM <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
d1 <- quasiDE(simdatRPKMTMM[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel= modnormRPKMTMM)
simdatRPKMTMM$QDE.praw <- d1[,2]
simdatRPKMTMM$quasiDE <- getadjp(simdatRPKMTMM, simdatRPKMTMM$QDE.praw)

# analyze the RLE normalized raw count data
simdatRLE$mean <- apply(simdatRLE[,1:n],1,mean)
simdatRLE$mean1 <- apply(simdatRLE[,1:(n/2)],1,mean)
simdatRLE$mean2 <- apply(simdatRLE[,(n/2+1):n],1,mean)
simdatRLE$var <- apply(simdatRLE[,1:n],1,var)
simdatRLE$var1 <- apply(simdatRLE[,1:(n/2)],1,var)
simdatRLE$var2 <- apply(simdatRLE[,(n/2+1):n],1,var)
meanest <- c(simdatRLE$mean1,simdatRLE$mean2)
varest <- c(simdatRLE$var1,simdatRLE$var2)
modnormRLE <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
d1 <- quasiDE(simdatRLE[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel=modnormRLE)
simdatRLE$QDE.praw <- d1[,2]
simdatRLE$quasiDE <- getadjp(simdatRLE, simdatRLE$QDE.praw)

# analyze the RLE normalized RPKM data
simdatRPKMRLE$mean <- apply(simdatRPKMRLE[,1:n],1,mean)
simdatRPKMRLE$mean1 <- apply(simdatRPKMRLE[,1:(n/2)],1,mean)
simdatRPKMRLE$mean2 <- apply(simdatRPKMRLE[,(n/2+1):n],1,mean)
simdatRPKMRLE$var <- apply(simdatRPKMRLE[,1:n],1,var)
simdatRPKMRLE$var1 <- apply(simdatRPKMRLE[,1:(n/2)],1,var)
simdatRPKMRLE$var2 <- apply(simdatRPKMRLE[,(n/2+1):n],1,var)
meanest <- c(simdatRPKMRLE$mean1,simdatRPKMRLE$mean2)
varest <- c(simdatRPKMRLE$var1,simdatRPKMRLE$var2)
modnormRPKMRLE <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
```

```
d1 <- quasiDE(simdatRPKMRLE[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel= modnormRPKMRLE)
simdatRPKMRLE$QDE.praw <- d1[,2]
simdatRPKMRLE$quasiDE <- getadjp(simdatRPKMRLE, simdatRPKMRLE$QDE.praw)

# analyze the UQ normalized raw count data
simdatUQ$mean <- apply(simdatUQ[,1:n],1,mean)
simdatUQ$mean1 <- apply(simdatUQ[,1:(n/2)],1,mean)
simdatUQ$mean2 <- apply(simdatUQ[,(n/2+1):n],1,mean)
simdatUQ$var <- apply(simdatUQ[,1:n],1,var)
simdatUQ$var1 <- apply(simdatUQ[,1:(n/2)],1,var)
simdatUQ$var2 <- apply(simdatUQ[,(n/2+1):n],1,var)
meanest <- c(simdatUQ$mean1,simdatUQ$mean2)
varest <- c(simdatUQ$var1,simdatUQ$var2)
modnormUQ <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
d1 <- quasiDE(simdatUQ[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel= modnormUQ)
simdatUQ$QDE.praw <- d1[,2]
simdatUQ$quasiDE <- getadjp(simdatUQ, simdatUQ$QDE.praw)

# analyze the UQ normalized RPKM data
simdatRPKMUQ$mean <- apply(simdatRPKMUQ[,1:n],1,mean)
simdatRPKMUQ$mean1 <- apply(simdatRPKMUQ[,1:(n/2)],1,mean)
simdatRPKMUQ$mean2 <- apply(simdatRPKMUQ[,(n/2+1):n],1,mean)
simdatRPKMUQ$var <- apply(simdatRPKMUQ[,1:n],1,var)
simdatRPKMUQ$var1 <- apply(simdatRPKMUQ[,1:(n/2)],1,var)
simdatRPKMUQ$var2 <- apply(simdatRPKMUQ[,(n/2+1):n],1,var)
meanest <- c(simdatRPKMUQ$mean1, simdatRPKMUQ$mean2)
varest <- c(simdatRPKMUQ$var1, simdatRPKMUQ$var2)
modnormRPKMUQ <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
d1 <- quasiDE(simdatRPKMUQ[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel= modnormRPKMUQ)
simdatRPKMUQ$QDE.praw <- d1[,2]
simdatRPKMUQ$quasiDE <- getadjp(simdatRPKMUQ, simdatRPKMUQ$QDE.praw)

# analyze the scaling normalized RPKM data
simdatRPKMscale$mean <- apply(simdatRPKMscale[,1:n],1,mean)
simdatRPKMscale$mean1 <- apply(simdatRPKMscale[,1:(n/2)],1,mean)
simdatRPKMscale$mean2 <- apply(simdatRPKMscale[,(n/2+1):n],1,mean)
simdatRPKMscale$var <- apply(simdatRPKMscale[,1:n],1,var)
simdatRPKMscale$var1 <- apply(simdatRPKMscale[,1:(n/2)],1,var)
simdatRPKMscale$var2 <- apply(simdatRPKMscale[,(n/2+1):n],1,var)
meanest <- c(simdatRPKMscale$mean1,simdatRPKMscale$mean2)
varest <- c(simdatRPKMscale$var1,simdatRPKMscale$var2)
modnormRPKMscale <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
d1 <- quasiDE(simdatRPKMscale[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel= modnormRPKMscale)
simdatRPKMscale$QDE.praw <- d1[,2]
simdatRPKMscale$quasiDE <- getadjp(simdatRPKMscale, simdatRPKMscale$QDE.praw)

# analyze the quantile normalized RPKM data
simdatRPKMqn$mean <- apply(simdatRPKMqn[,1:n],1,mean)
simdatRPKMqn$mean1 <- apply(simdatRPKMqn[,1:(n/2)],1,mean)
simdatRPKMqn$mean2 <- apply(simdatRPKMqn[,(n/2+1):n],1,mean)
simdatRPKMqn$var <- apply(simdatRPKMqn[,1:n],1,var)
simdatRPKMqn$var1 <- apply(simdatRPKMqn[,1:(n/2)],1,var)
simdatRPKMqn$var2 <- apply(simdatRPKMqn[,(n/2+1):n],1,var)
meanest <- c(simdatRPKMqn$mean1,simdatRPKMqn$mean2)
varest <- c(simdatRPKMqn$var1,simdatRPKMqn$var2)
modnormRPKMqn <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
d1 <- quasiDE(simdatRPKMqn[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel= modnormRPKMqn)
simdatRPKMqn$QDE.praw <- d1[,2]
simdatRPKMqn$quasiDE <- getadjp(simdatRPKMqn, simdatRPKMqn$QDE.praw)

# analyze the Cyclic Loess normalized RPKM data
simdatRPKMcl$mean <- apply(simdatRPKMcl[,1:n],1,mean)
simdatRPKMcl$mean1 <- apply(simdatRPKMcl[,1:(n/2)],1,mean)
simdatRPKMcl$mean2 <- apply(simdatRPKMcl[,(n/2+1):n],1,mean)
simdatRPKMcl$var <- apply(simdatRPKMcl[,1:n],1,var)
simdatRPKMcl$var1 <- apply(simdatRPKMcl[,1:(n/2)],1,var)
simdatRPKMcl$var2 <- apply(simdatRPKMcl[,(n/2+1):n],1,var)
meanest <- c(simdatRPKMcl$mean1,simdatRPKMcl$mean2)
varest <- c(simdatRPKMcl$var1,simdatRPKMcl$var2)
modnormRPKMcl <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
```

```
d1 <- quasiDE(simdatRPKMcl[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel= modnormRPKMcl)
simdatRPKMcl$QDE.praw <- d1[,2]
simdatRPKMcl$quasiDE <- getadjp(simdatRPKMcl, simdatRPKMcl$QDE.praw)

# analyze the invariant set normalized RPKM data
simdatRPKMIV$mean <- apply(simdatRPKMIV[,1:n],1,mean)
simdatRPKMIV$mean1 <- apply(simdatRPKMIV[,1:(n/2)],1,mean)
simdatRPKMIV$mean2 <- apply(simdatRPKMIV[,(n/2+1):n],1,mean)
simdatRPKMIV$var <- apply(simdatRPKMIV[,1:n],1,var)
simdatRPKMIV$var1 <- apply(simdatRPKMIV[,1:(n/2)],1,var)
simdatRPKMIV$var2 <- apply(simdatRPKMIV[,(n/2+1):n],1,var)
meanest <- c(simdatRPKMIV$mean1,simdatRPKMIV$mean2)
varest <- c(simdatRPKMIV$var1,simdatRPKMIV$var2)
modnormRPKMIV <- smooth.spline(meanest[varest != 0],log(varest[varest != 0]))
d1 <- quasiDE(simdatRPKMIV[,1:n],design1=dsgn.full, design2=dsgn.null, splmodel= modnormRPKMIV)
simdatRPKMIV$QDE.praw <- d1[,2]
simdatRPKMIV$quasiDE <- getadjp(simdatRPKMIV, simdatRPKMIV$QDE.praw)

# calculate number of DE genes and total genes after filtering in different datasets
lblnum <- as.numeric(substr(rownames(simdat),2,max(nchar(rownames(simdat)))))
numDE <- sum(lblnum <= ngenes*ratio)
nGen <- nrow(simdat)

lblnum.gl.TMM <- as.numeric(substr(rownames(simdatglTMM),2,max(nchar(rownames(simdatglTMM)))))
numDE.gl.TMM <- sum(lblnum.gl.TMM <= ngenes*ratio)
nGen.gl.TMM <- nrow(simdatglTMM)

lblnumRPKM <- as.numeric(substr(rownames(simdatRPKM),2,max(nchar(rownames(simdatRPKM)))))
numDERPKM <- sum(lblnumRPKM <= ngenes*ratio)
nGenRPKM <- nrow(simdatRPKM)

lblnumscale <- as.numeric(substr(rownames(simdatRPKMscale),2,max(nchar(rownames(simdatRPKMscale)))))
numDEscale <- sum(lblnumscale <= ngenes*ratio)
nGenscale <- nrow(simdatRPKMscale)

lblnumqn <- as.numeric(substr(rownames(simdatRPKMqn),2,max(nchar(rownames(simdatRPKMqn)))))
numDEqn <- sum(lblnumqn <= ngenes*ratio)
nGenqn <- nrow(simdatRPKMqn)

lblnumcl <- as.numeric(substr(rownames(simdatRPKMcl),2,max(nchar(rownames(simdatRPKMcl)))))
numDEcl <- sum(lblnumcl <= ngenes*ratio)
nGencl <- nrow(simdatRPKMcl)

lblnumIV <- as.numeric(substr(rownames(simdatRPKMIV),2,max(nchar(rownames(simdatRPKMIV)))))
numDEIV <- sum(lblnumIV <= ngenes*ratio)
nGenIV <- nrow(simdatRPKMIV)

# calculate sensitivities and real FDR in different settings
collbl <- c("tp","fp","tn","fn","sen","spc","accuracy","FDR")
rowlbl <-
c("GL.TMM","TMM","RLE","UQ","RPKM","RPKM.TMM","RPKM.RLE","RPKM.UQ","RPKM.scale","RPKM.qn","RPKM.cl","RPKM.IV")
paraQuasiDE <- matrix(NA,nrow=12,ncol=8)
paraLIMMA <- matrix(NA,nrow=12,ncol=8)
paraedgeR <- matrix(NA,nrow=3,ncol=8)
rownames(paraQuasiDE) <- rownames(paraLIMMA) <- rowlbl
colnames(paraQuasiDE) <- colnames(paraLIMMA) <- collbl
rownames(paraedgeR) <- rowlbl[1:3]
colnames(paraedgeR) <- collbl

paraQuasiDE[1,] <- paraest(numDE.gl.TMM,nGen.gl.TMM,simdatglTMM$quasiDE.gl.TMM,ngenes,ratio)
paraQuasiDE[2,] <- paraest(numDE,nGen,simdatTMM$quasiDE,ngenes,ratio)
paraQuasiDE[3,] <- paraest(numDE,nGen,simdatRLE$quasiDE,ngenes,ratio)
paraQuasiDE[4,] <- paraest(numDE,nGen,simdatUQ$quasiDE,ngenes,ratio)
paraQuasiDE[5,] <- paraest(numDE,nGen,simdatRPKM$quasiDE,ngenes,ratio)
paraQuasiDE[6,] <- paraest(numDE,nGen,simdatRPKMTMM$quasiDE,ngenes,ratio)
paraQuasiDE[7,] <- paraest(numDE,nGen,simdatRPKMRLE$quasiDE,ngenes,ratio)
paraQuasiDE[8,] <- paraest(numDE,nGen,simdatRPKMUQ$quasiDE,ngenes,ratio)
paraQuasiDE[9,] <- paraest(numDE,nGen,simdatRPKMscale$quasiDE,ngenes,ratio)
paraQuasiDE[10,] <- paraest(numDE,nGen,simdatRPKMqn$quasiDE,ngenes,ratio)
paraQuasiDE[11,] <- paraest(numDE,nGen,simdatRPKMcl$quasiDE,ngenes,ratio)
```

```
paraQuasiDE[12,] <- paraest(numDE,nGen,simdatRPKMIV$quasiDE,ngenes,ratio)

paraLIMMA[1,] <- paraest(numDE.gl.TMM,nGen.gl.TMM,simdatglTMM$lq.gl.TMM,ngenes,ratio)
paraLIMMA[2,] <- paraest(numDE,nGen,simdat$lq.TMM,ngenes,ratio)
paraLIMMA[3,] <- paraest(numDE,nGen,simdat$lq.RLE,ngenes,ratio)
paraLIMMA[4,] <- paraest(numDE,nGen,simdat$lq.UQ,ngenes,ratio)
paraLIMMA[5,] <- paraest(numDE,nGen,simdatRPKM$lq.raw,ngenes,ratio)
paraLIMMA[6,] <- paraest(numDE,nGen,simdatRPKM$lq.TMM,ngenes,ratio)
paraLIMMA[7,] <- paraest(numDE,nGen,simdatRPKM$lq.RLE,ngenes,ratio)
paraLIMMA[8,] <- paraest(numDE,nGen,simdatRPKM$lq.UQ,ngenes,ratio)
paraLIMMA[9,] <- paraest(numDE,nGen,simdatRPKMscale$lq.scale,ngenes,ratio)
paraLIMMA[10,] <- paraest(numDE,nGen,simdatRPKMqn$lq.qn,ngenes,ratio)
paraLIMMA[11,] <- paraest(numDE,nGen,simdatRPKMcl$lq.cl,ngenes,ratio)
paraLIMMA[12,] <- paraest(numDE,nGen,simdatRPKMIV$lq.IV,ngenes,ratio)

paraedgeR[1,] <- paraest(numDE,nGen,simdat$edgeq.TMM,ngenes,ratio)
paraedgeR[2,] <- paraest(numDE,nGen,simdat$edgeq.RLE,ngenes,ratio)
paraedgeR[3,] <- paraest(numDE,nGen,simdat$edgeq.UQ,ngenes,ratio)

print(paraQuasiDE)
print(paraLIMMA)
print(paraedgeR)
```

APPENDIX B

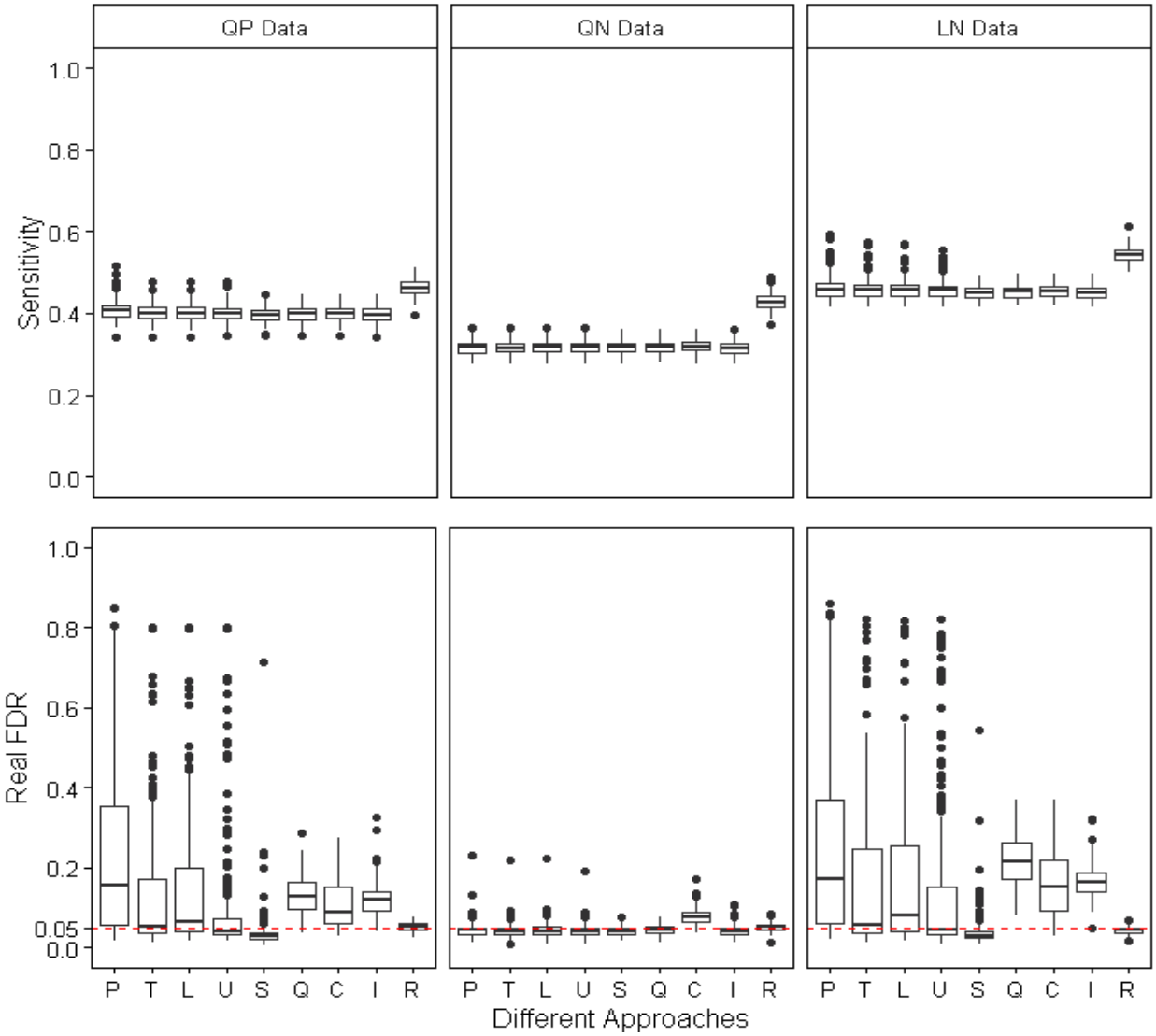

Figure B 1. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, Low Fold Change, Fixed Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

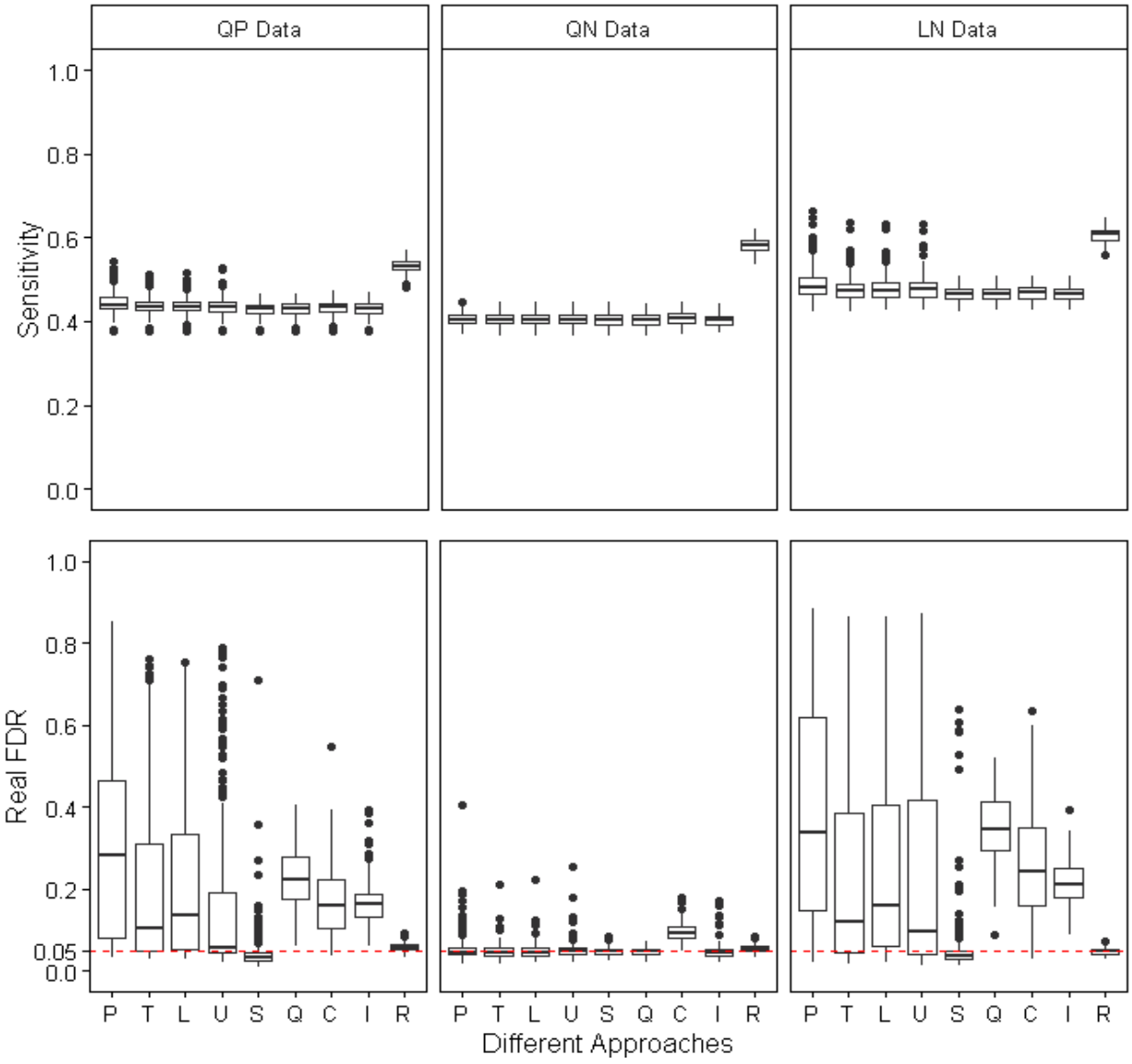

Figure B 2. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, Low Fold Change, Fixed Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

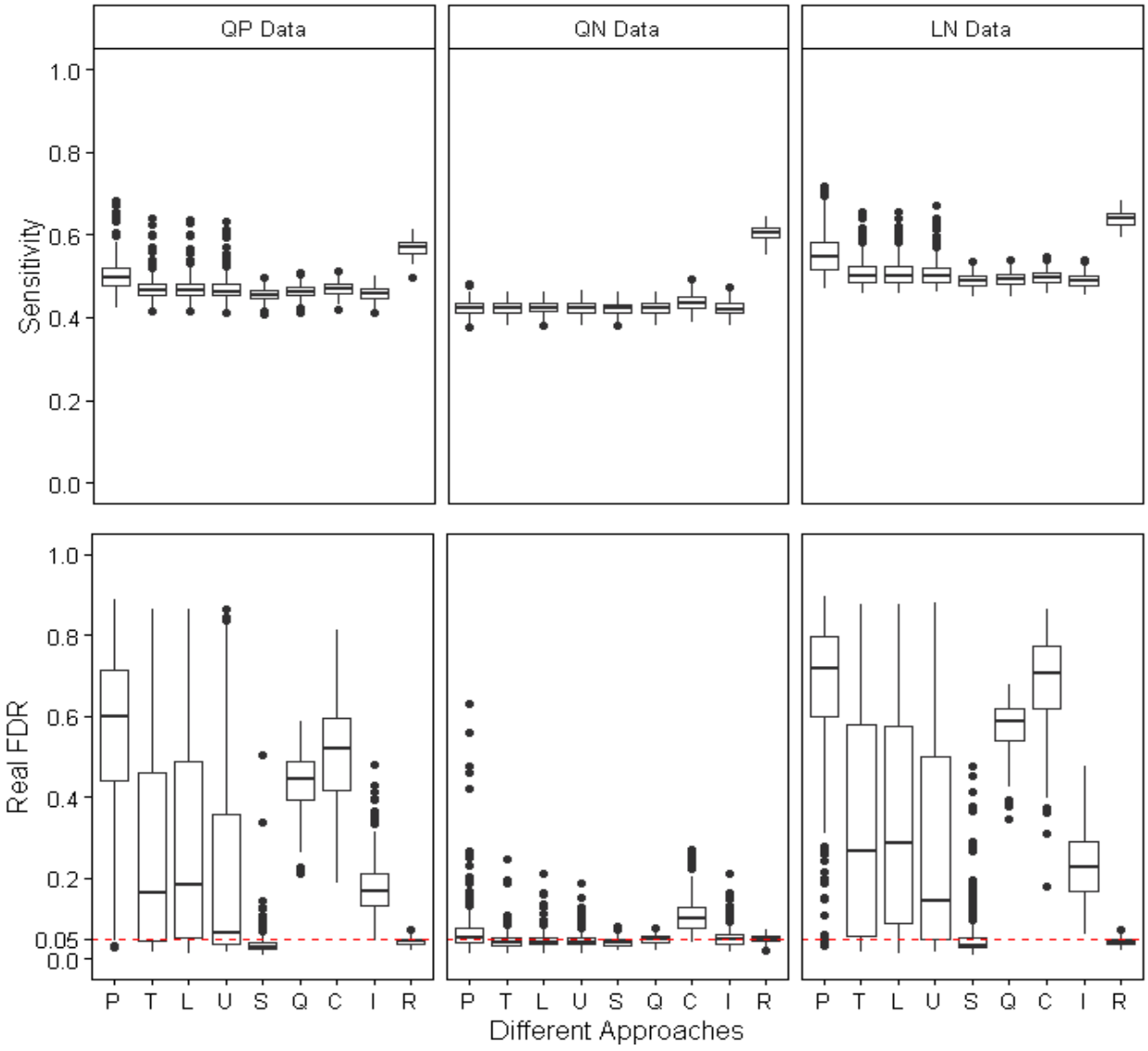

Figure B 3. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, High Fold Change, Fixed Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

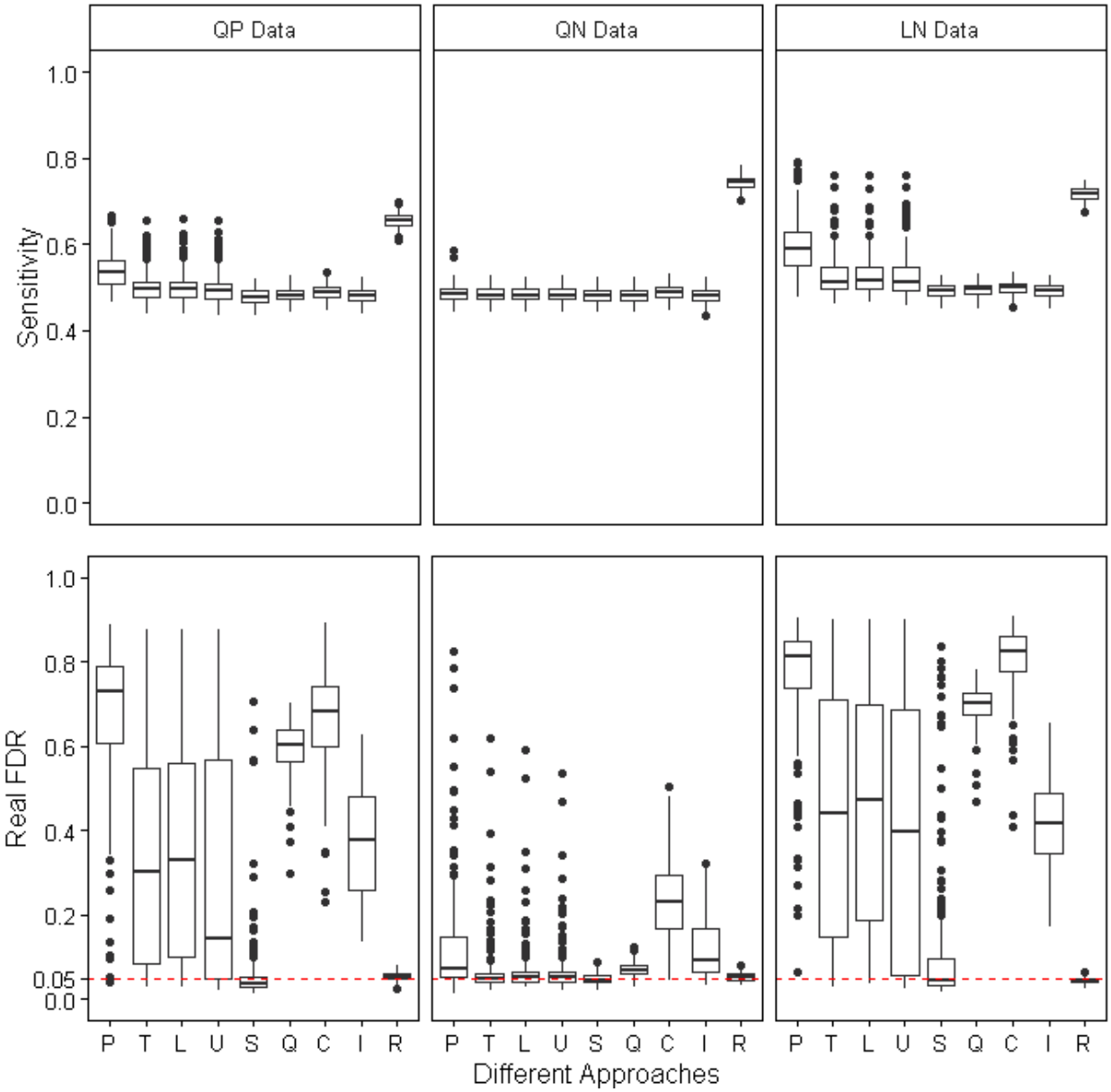

Figure B 4. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, High Fold Change, Fixed Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

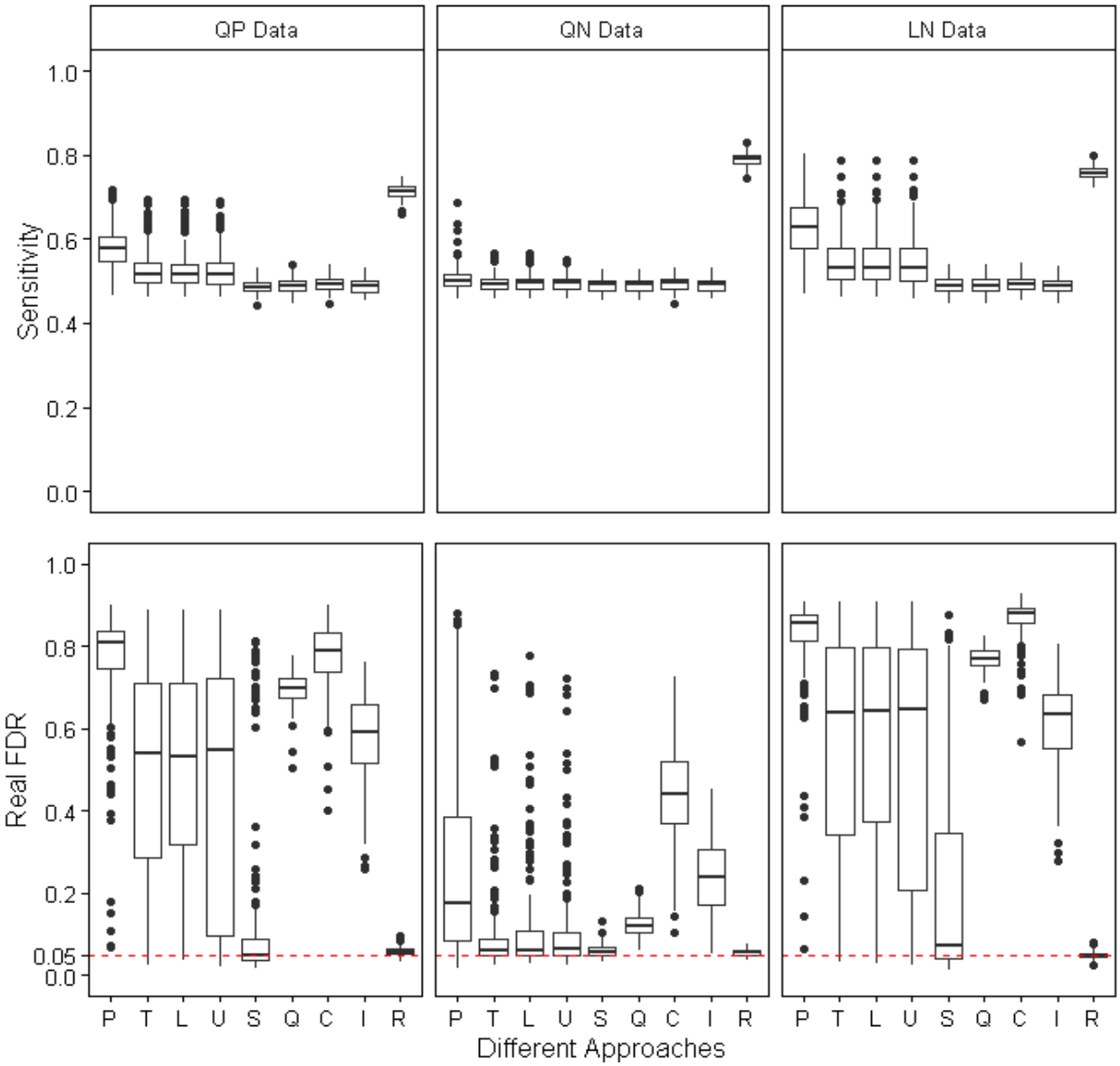

Figure B 5. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, High Fold Change, Fixed Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

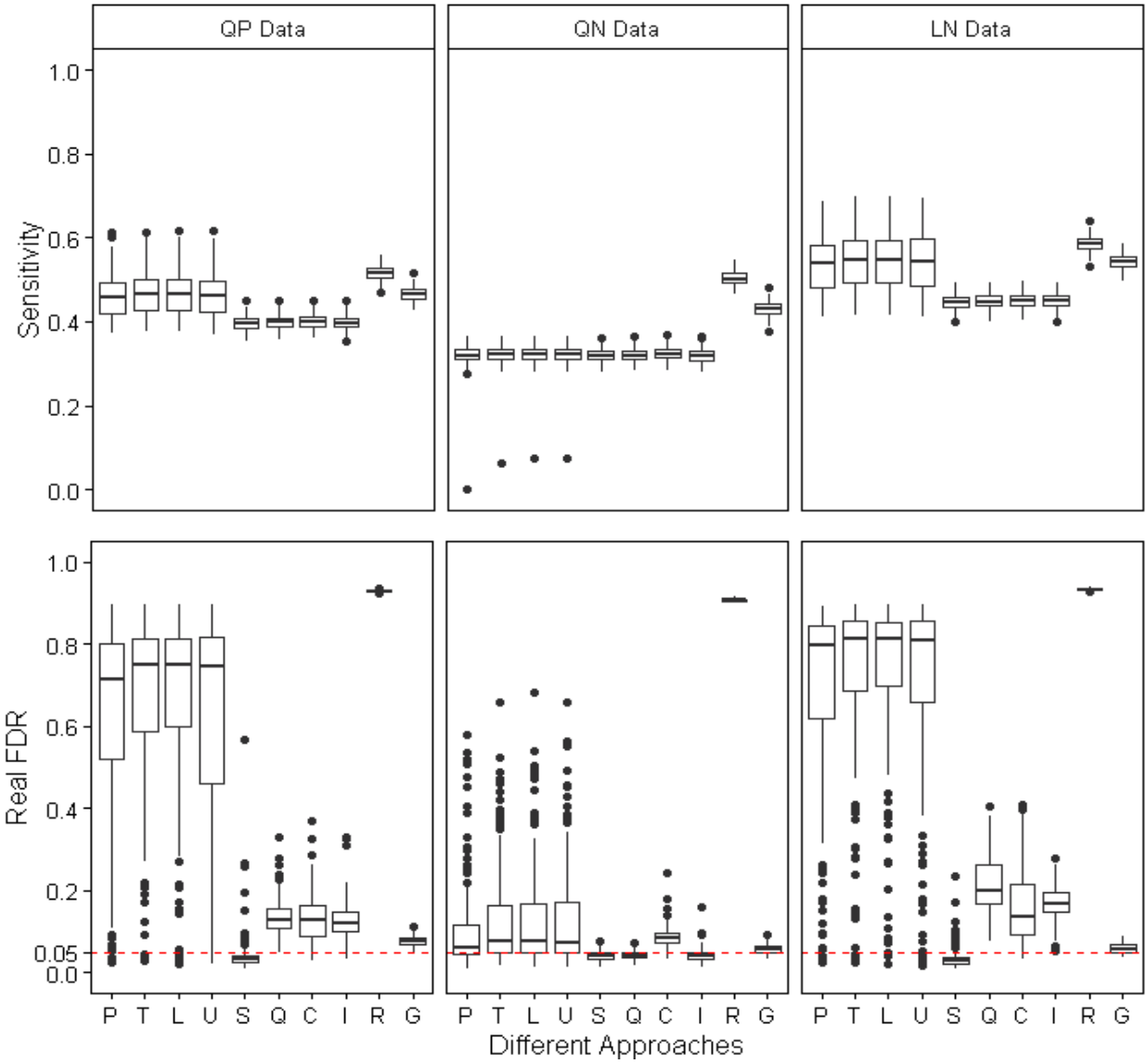

Figure B 6. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, Low Fold Change, Varied Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

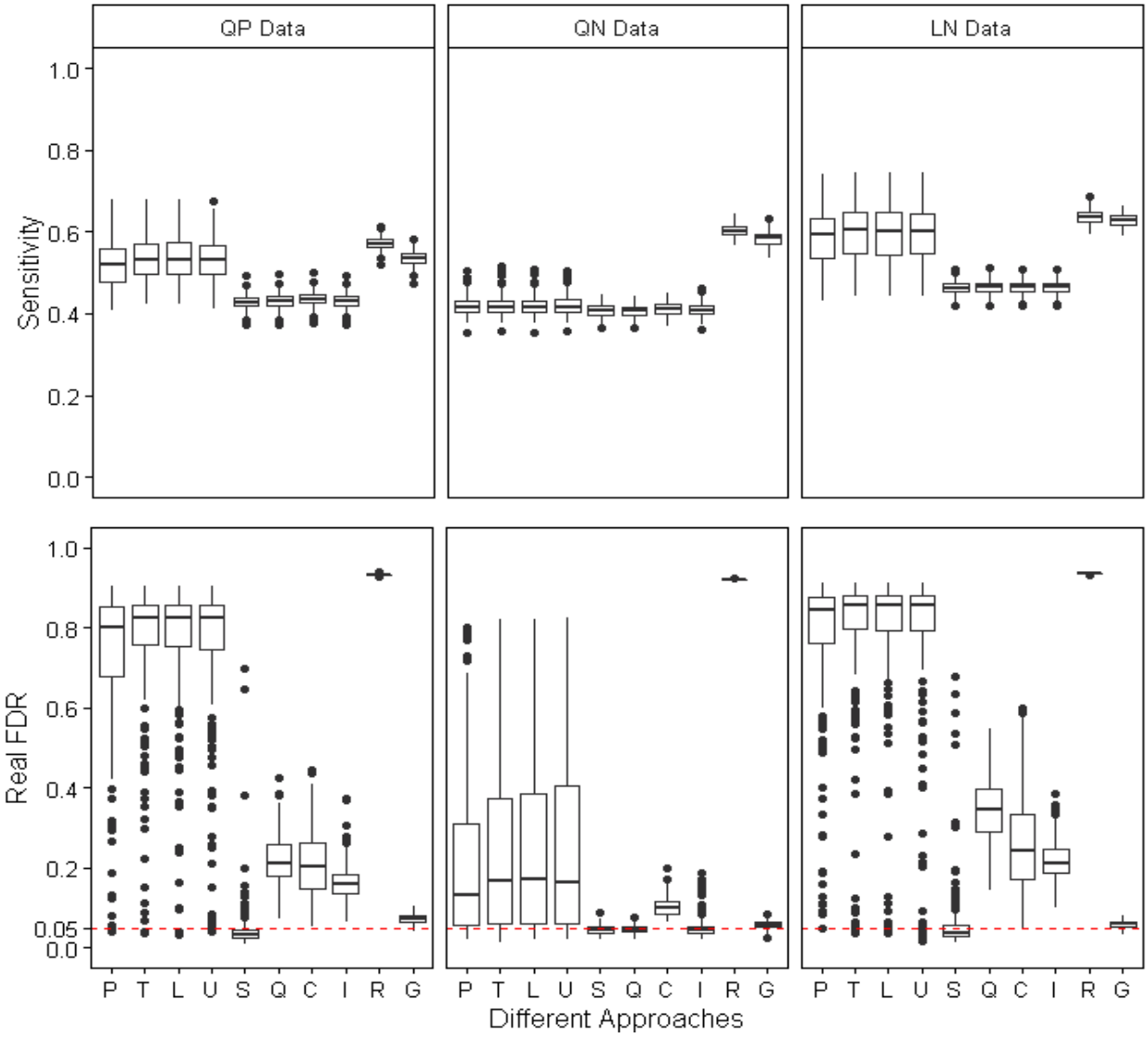

Figure B 7. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, Low Fold Change, Varied Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

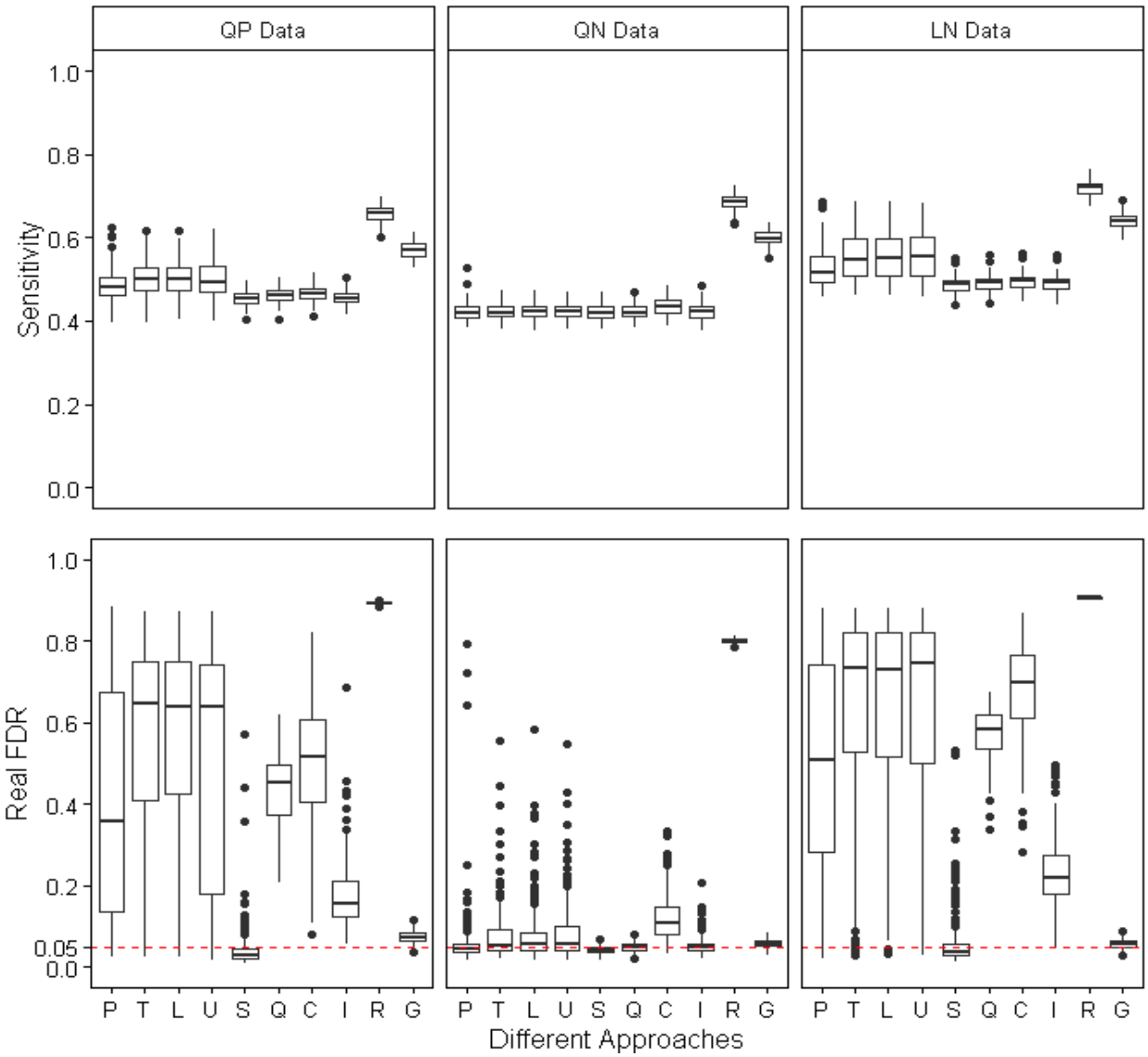

Figure B 8. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, High Fold Change, Varied Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

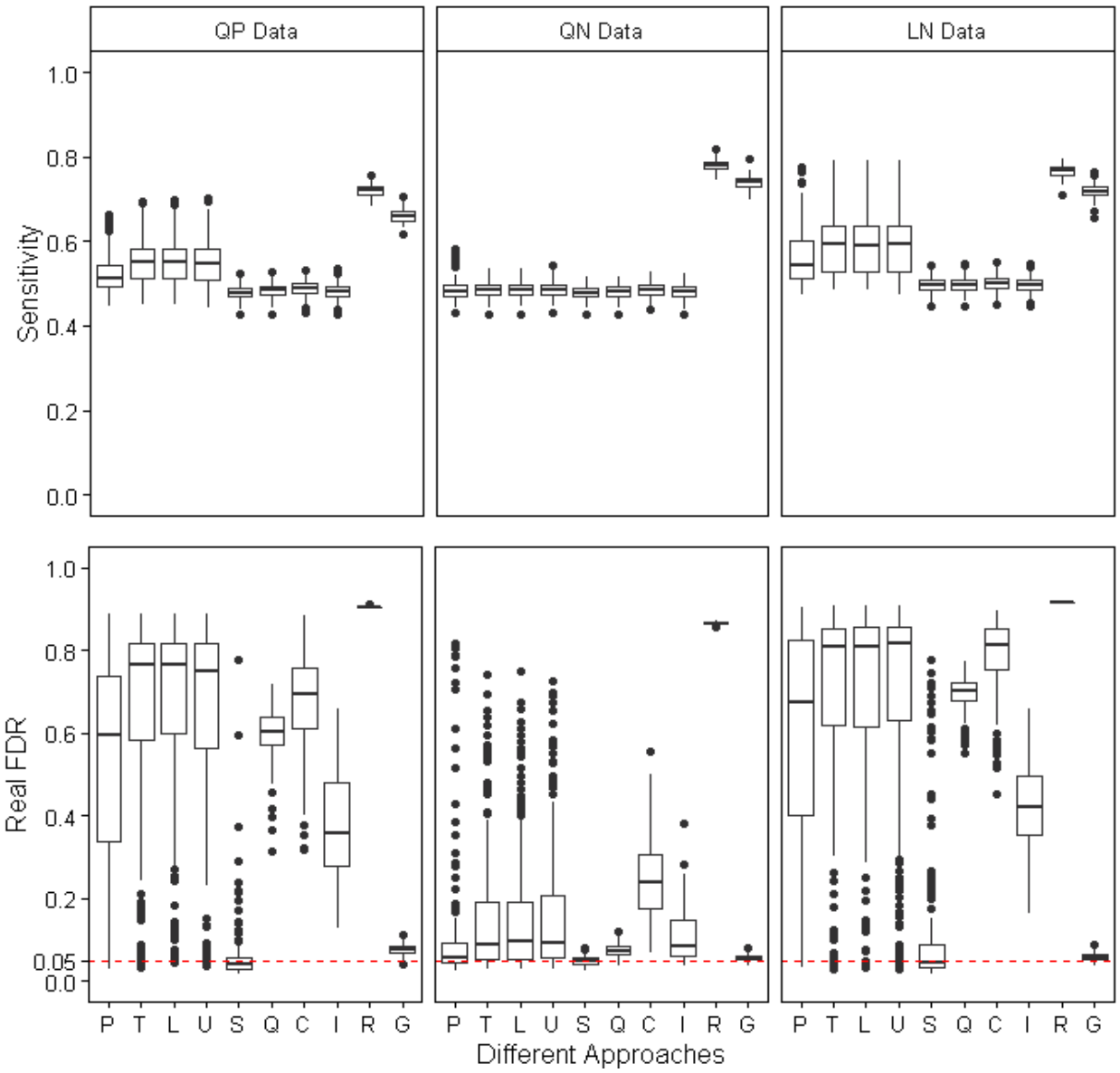

Figure B 9. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, High Fold Change, Varied Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

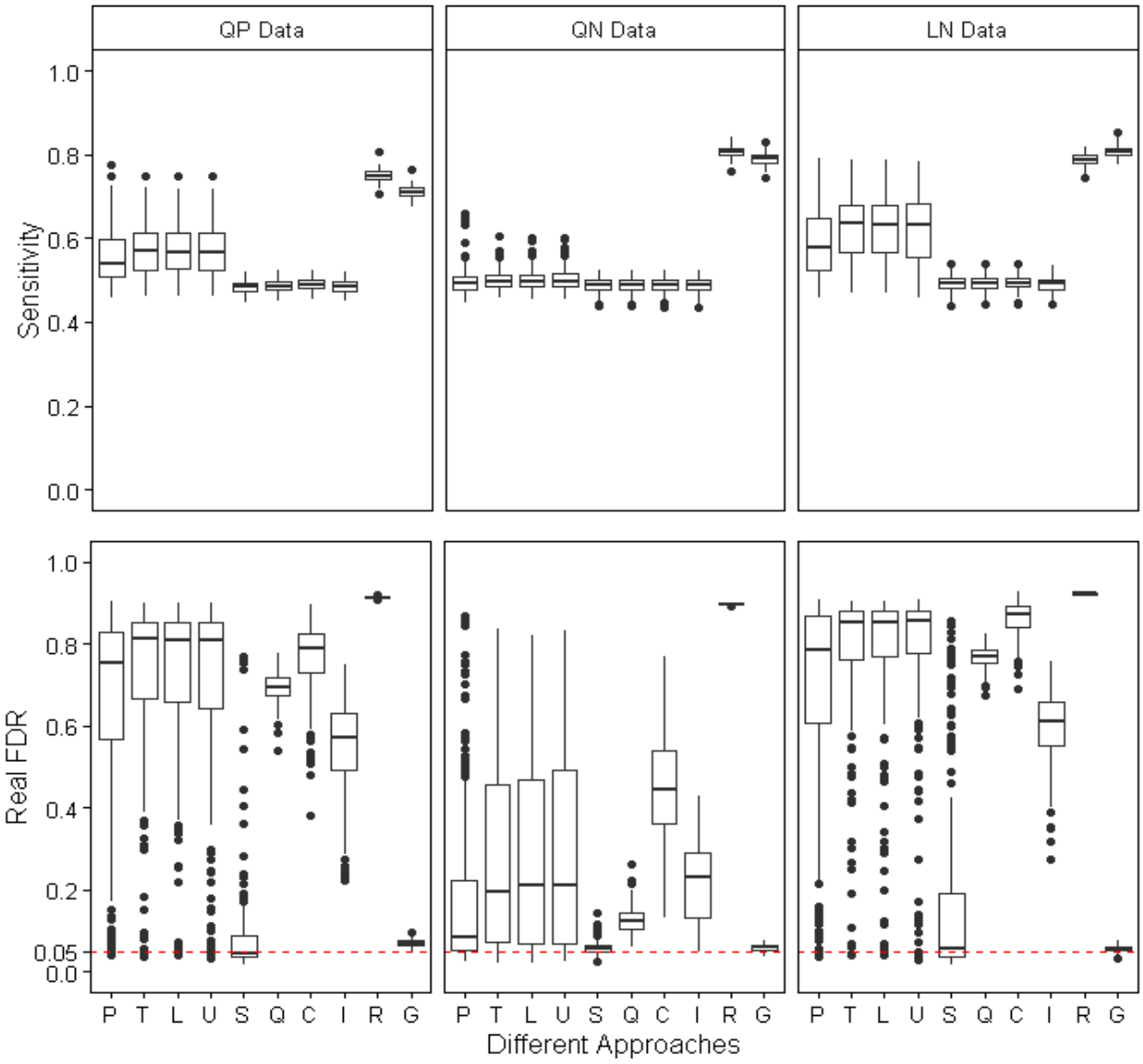

Figure B 10. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, High Fold Change, Varied Gene Length, QuasiDE)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

APPENDIX C

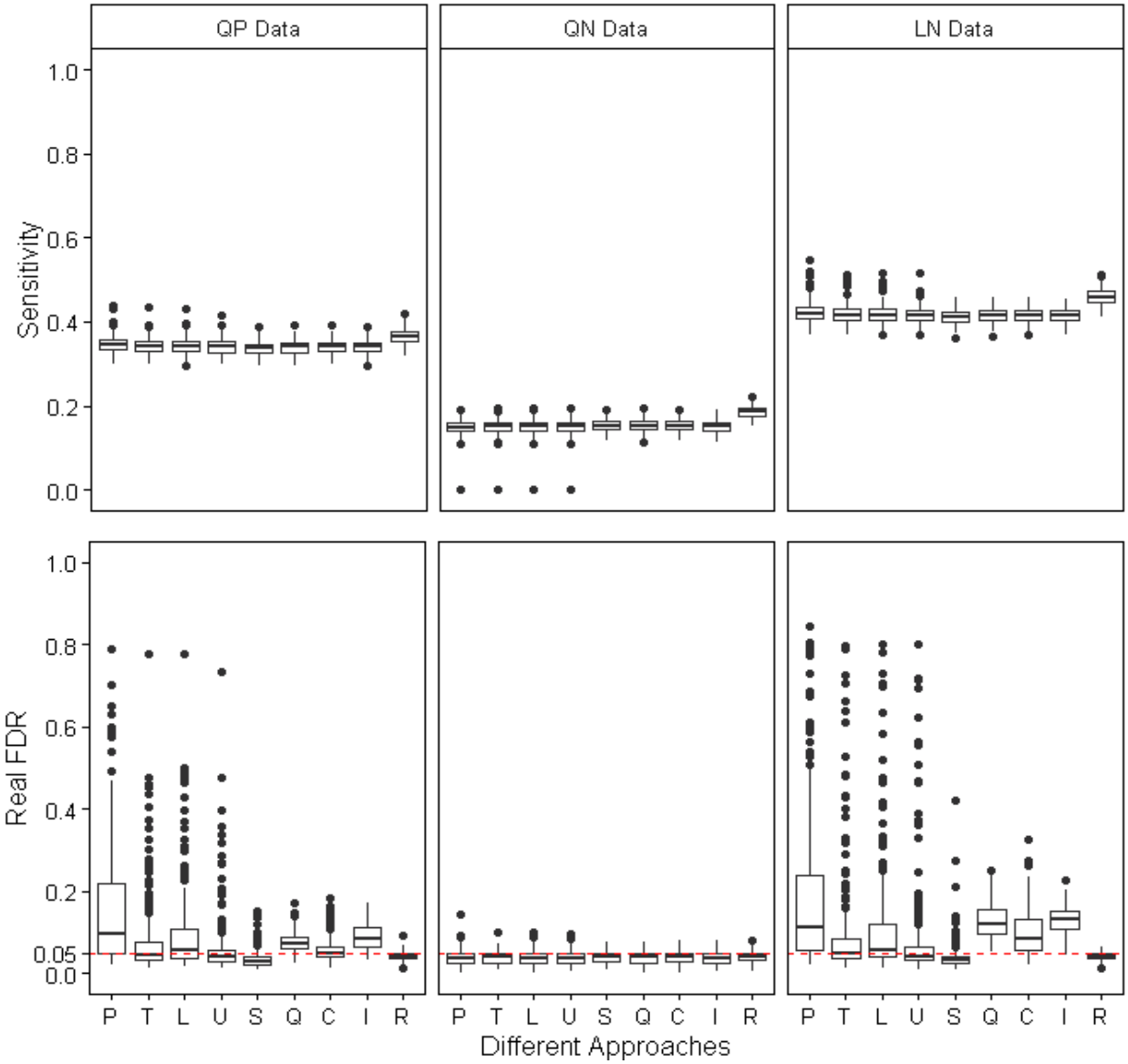

Figure C 1. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, Low Fold Change, Fixed Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

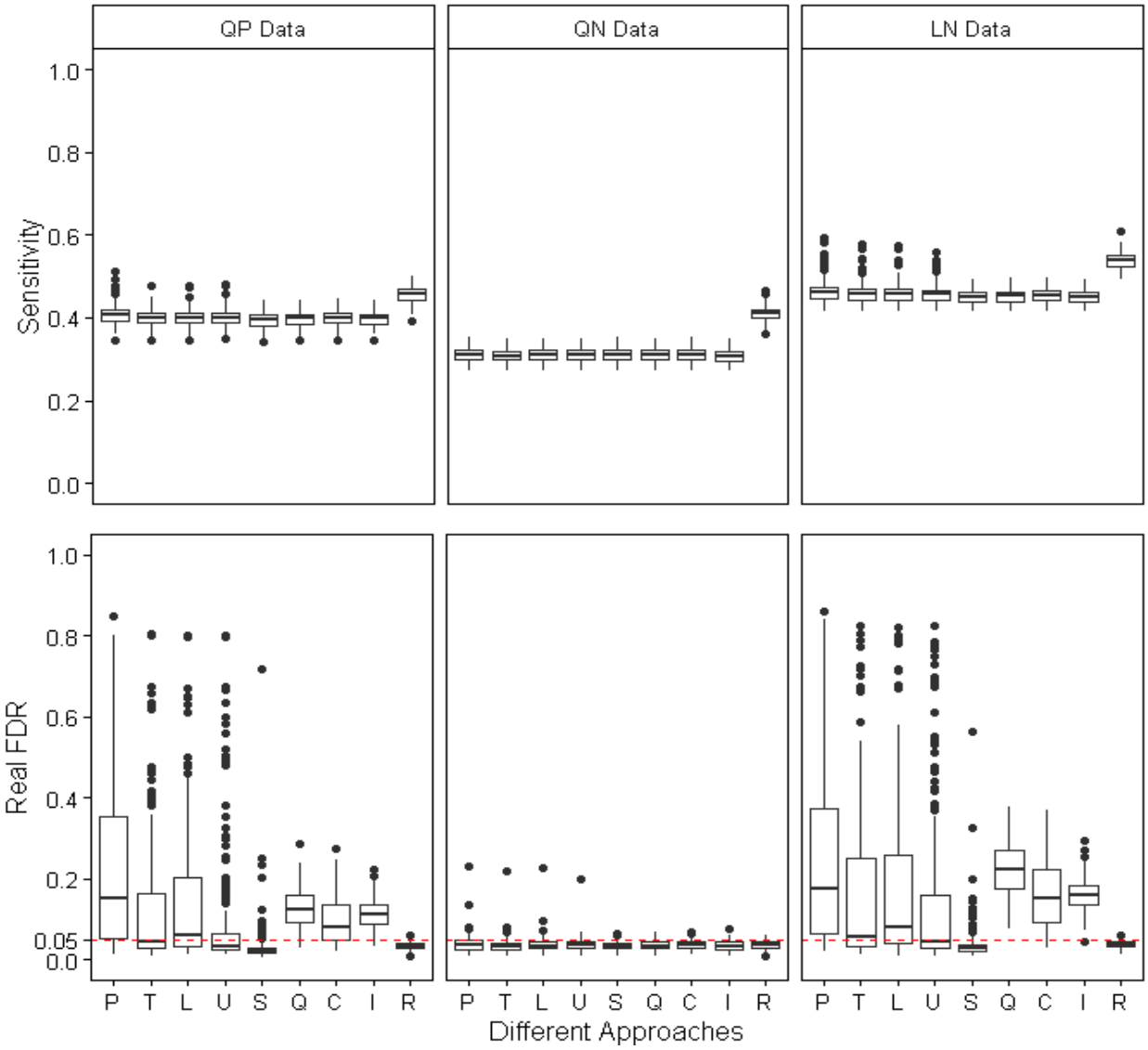

Figure C 2. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, Low Fold Change, Fixed Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

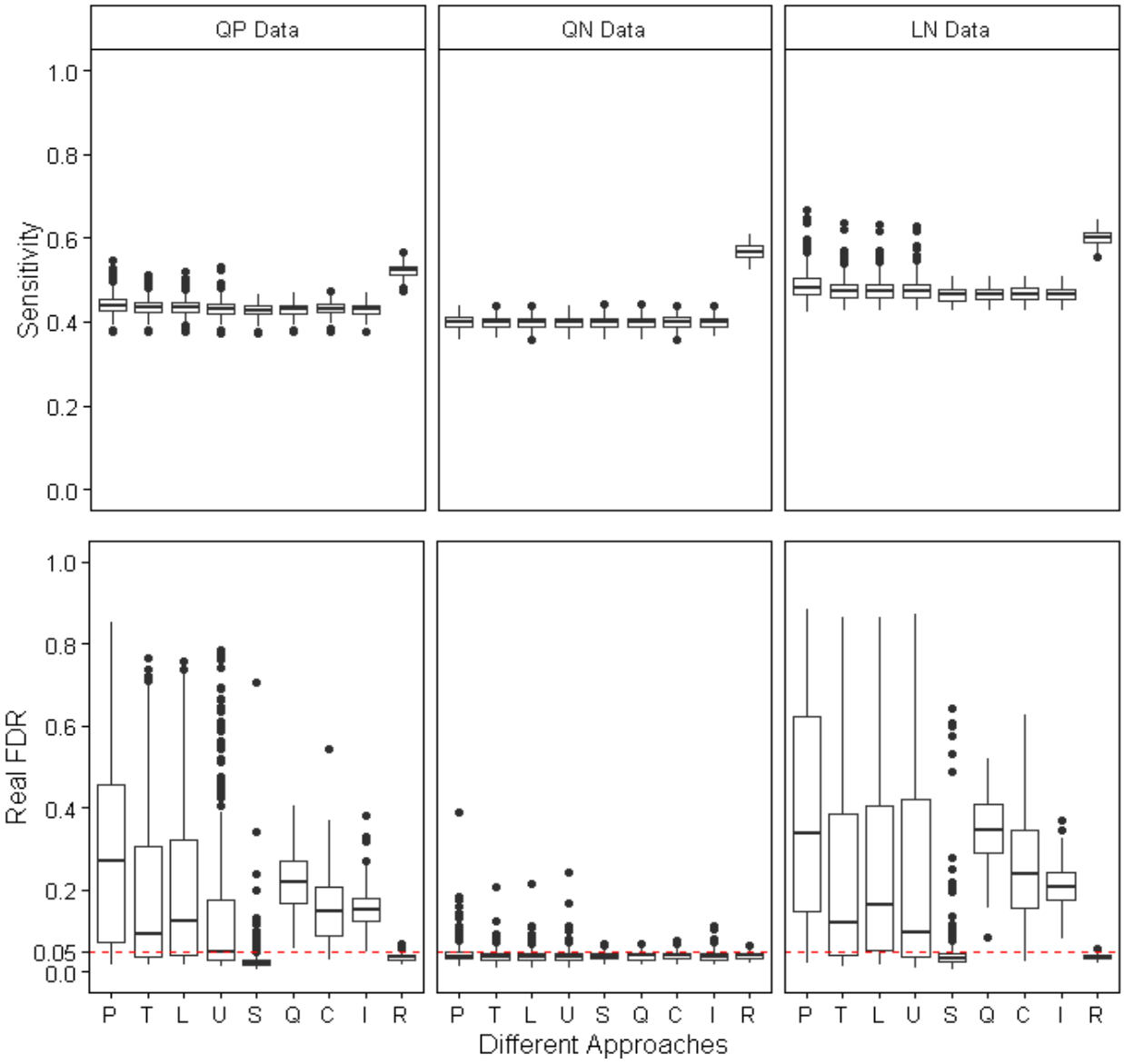

Figure C 3. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, Low Fold Change, Fixed Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

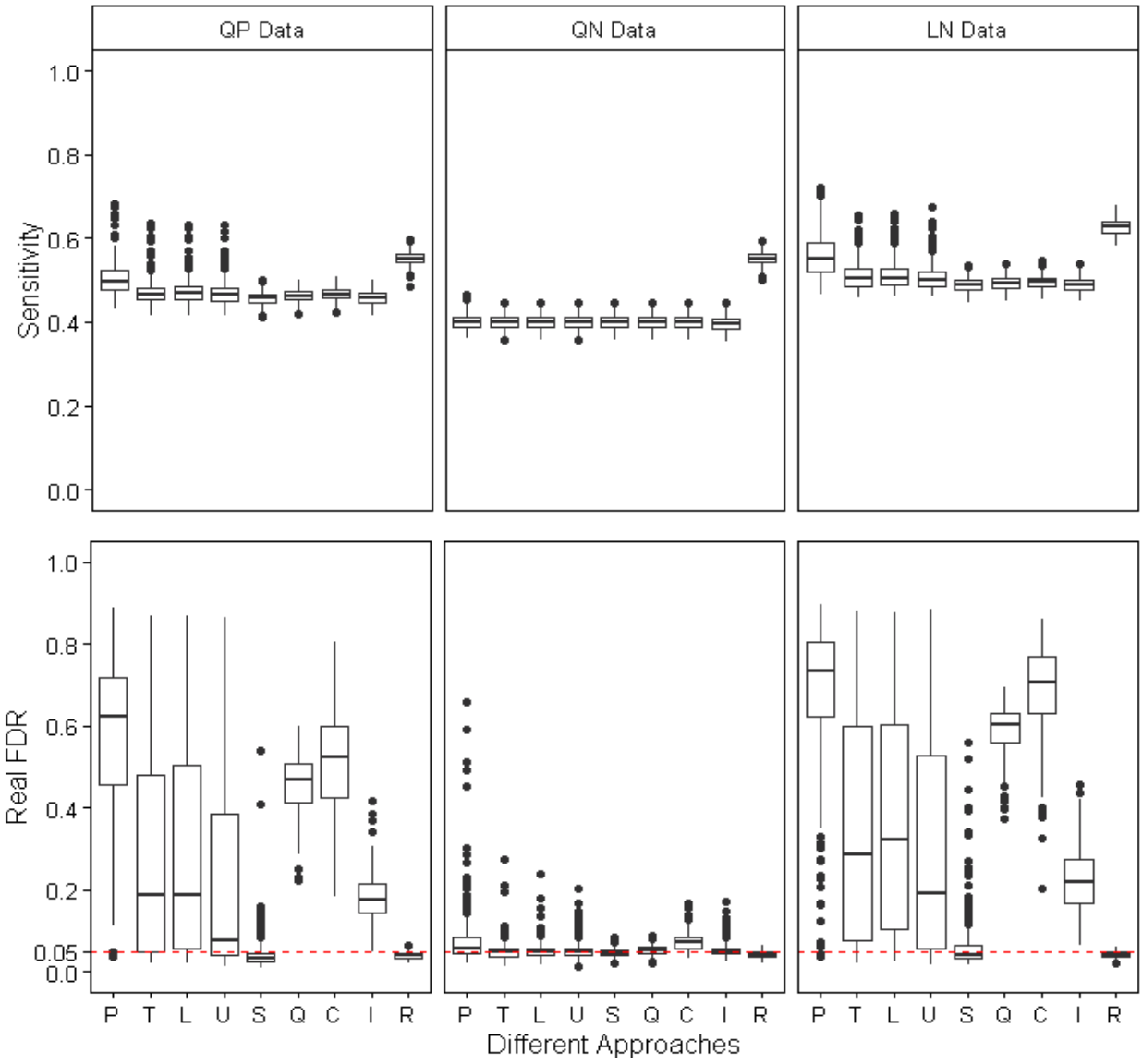

Figure C 4. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, High Fold Change, Fixed Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

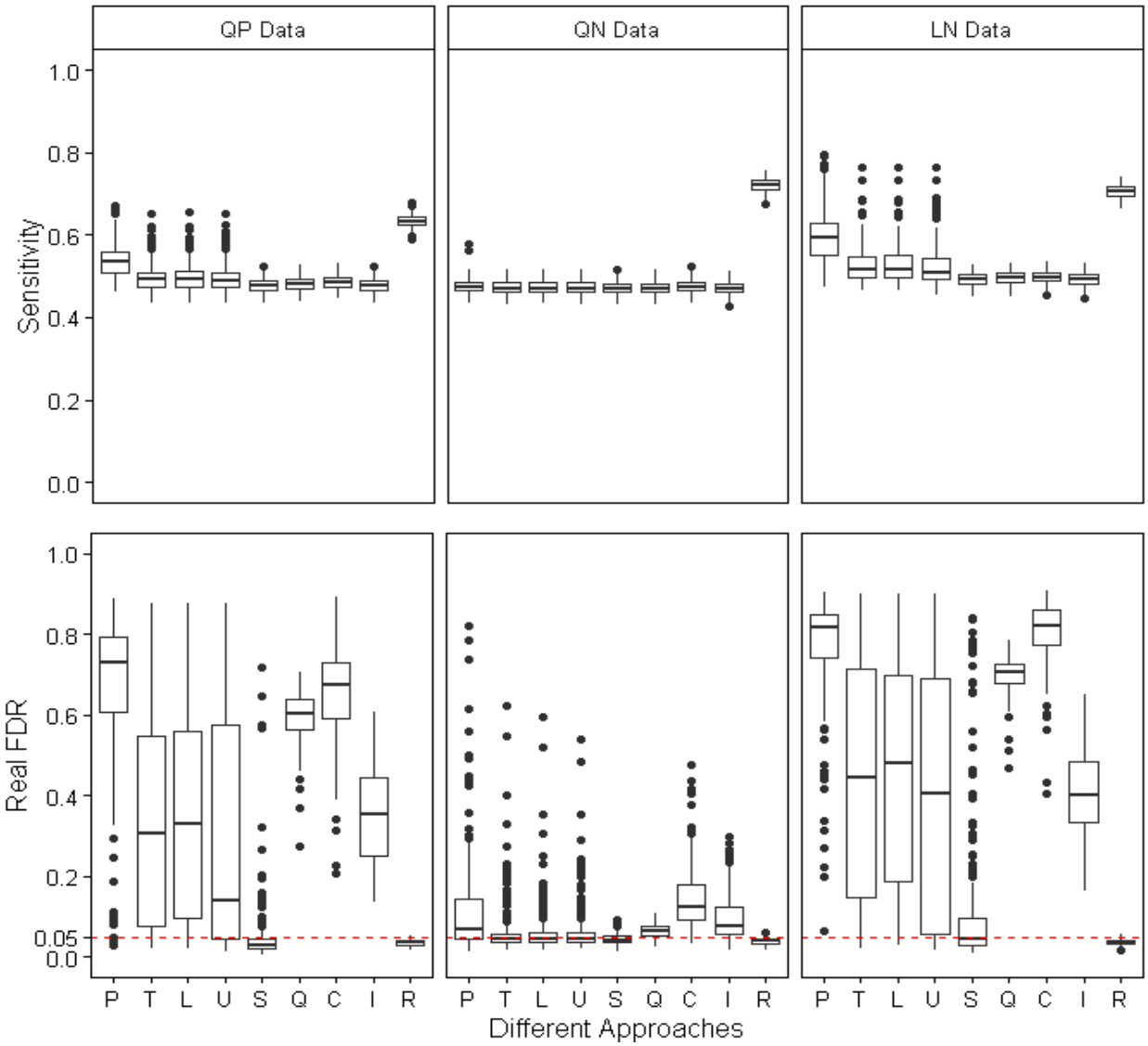

Figure C 5. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, High Fold Change, Fixed Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

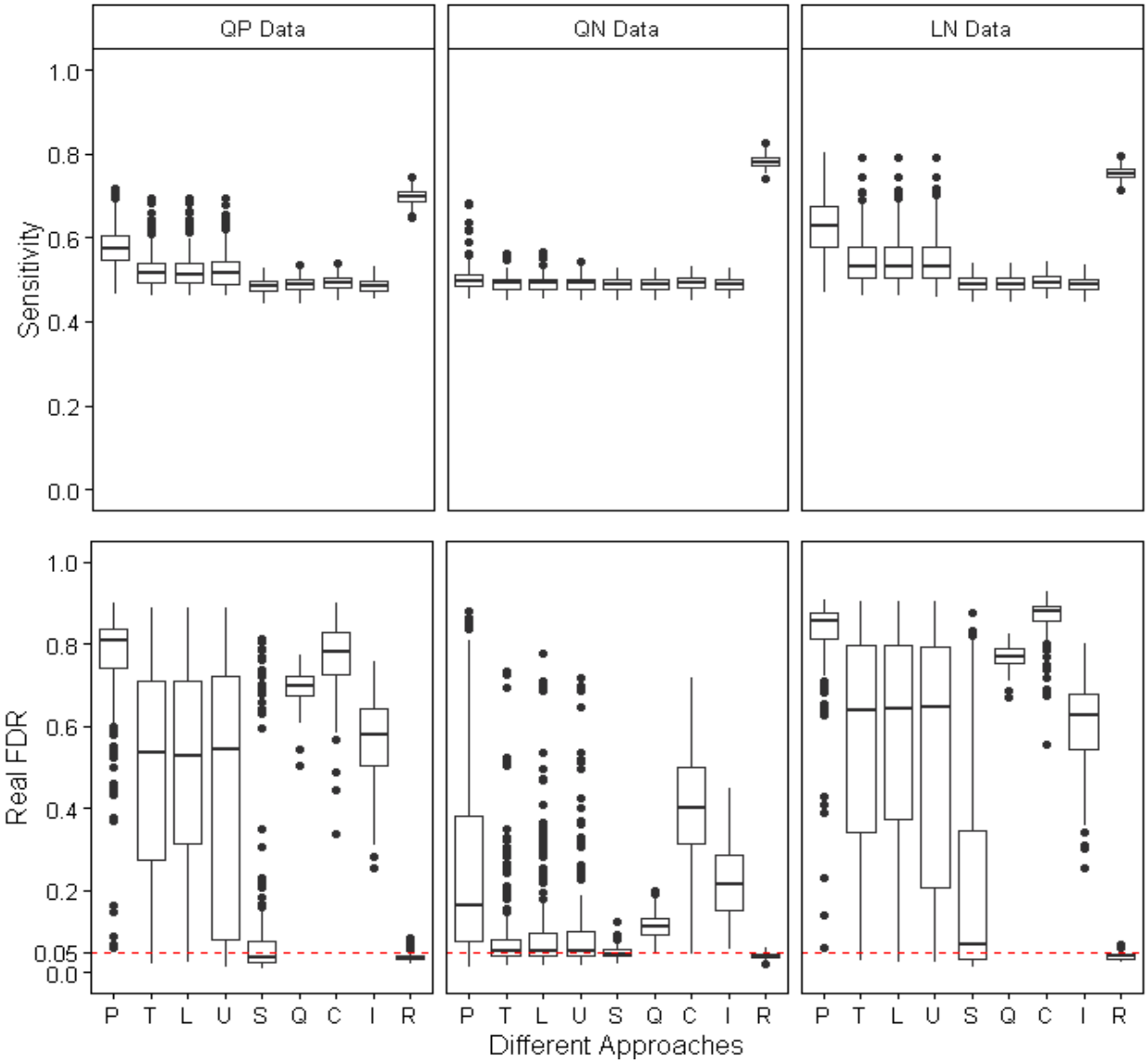

Figure C 6. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, High Fold Change, Fixed Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

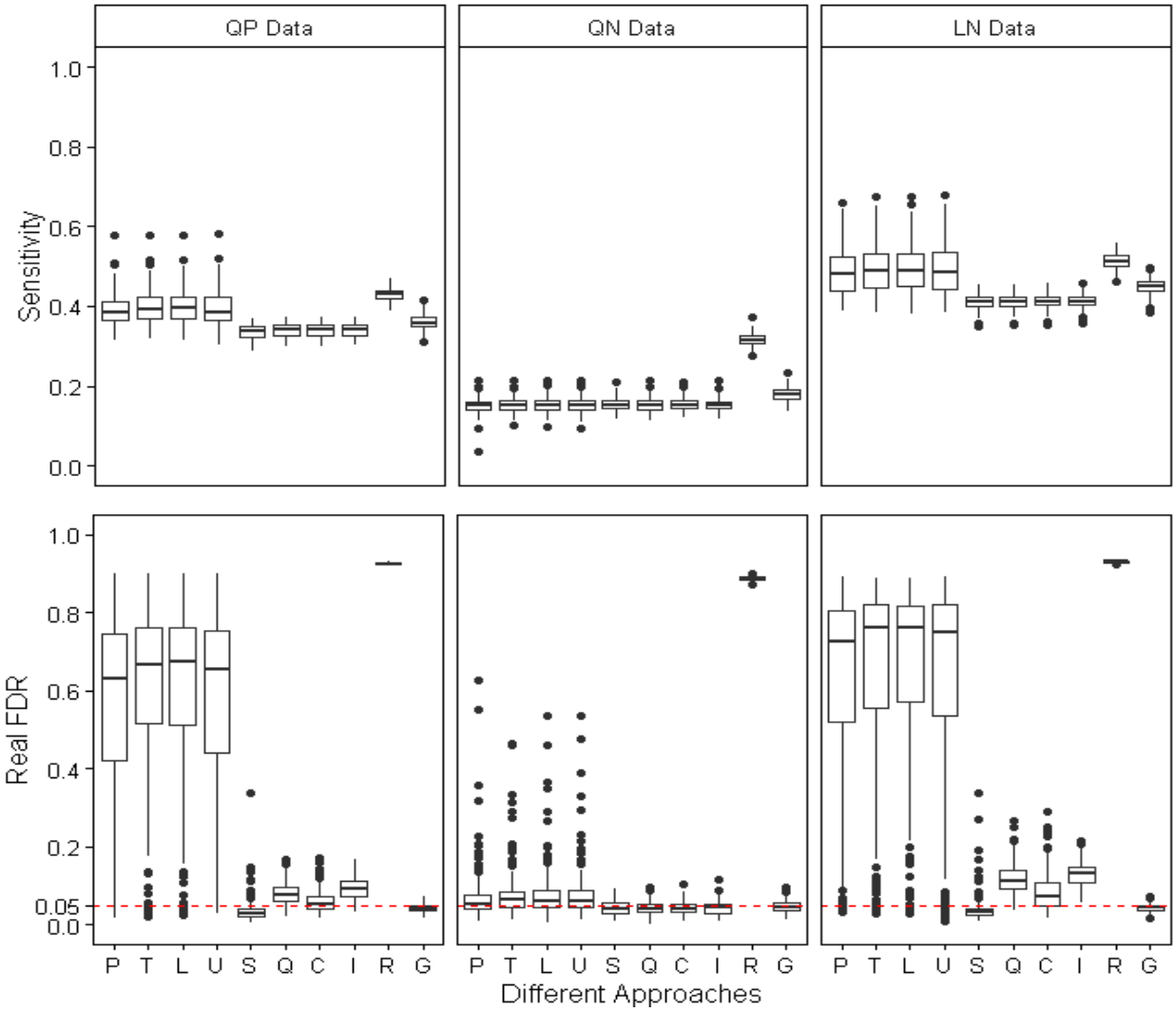

Figure C 7. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, Low Fold Change, Varied Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

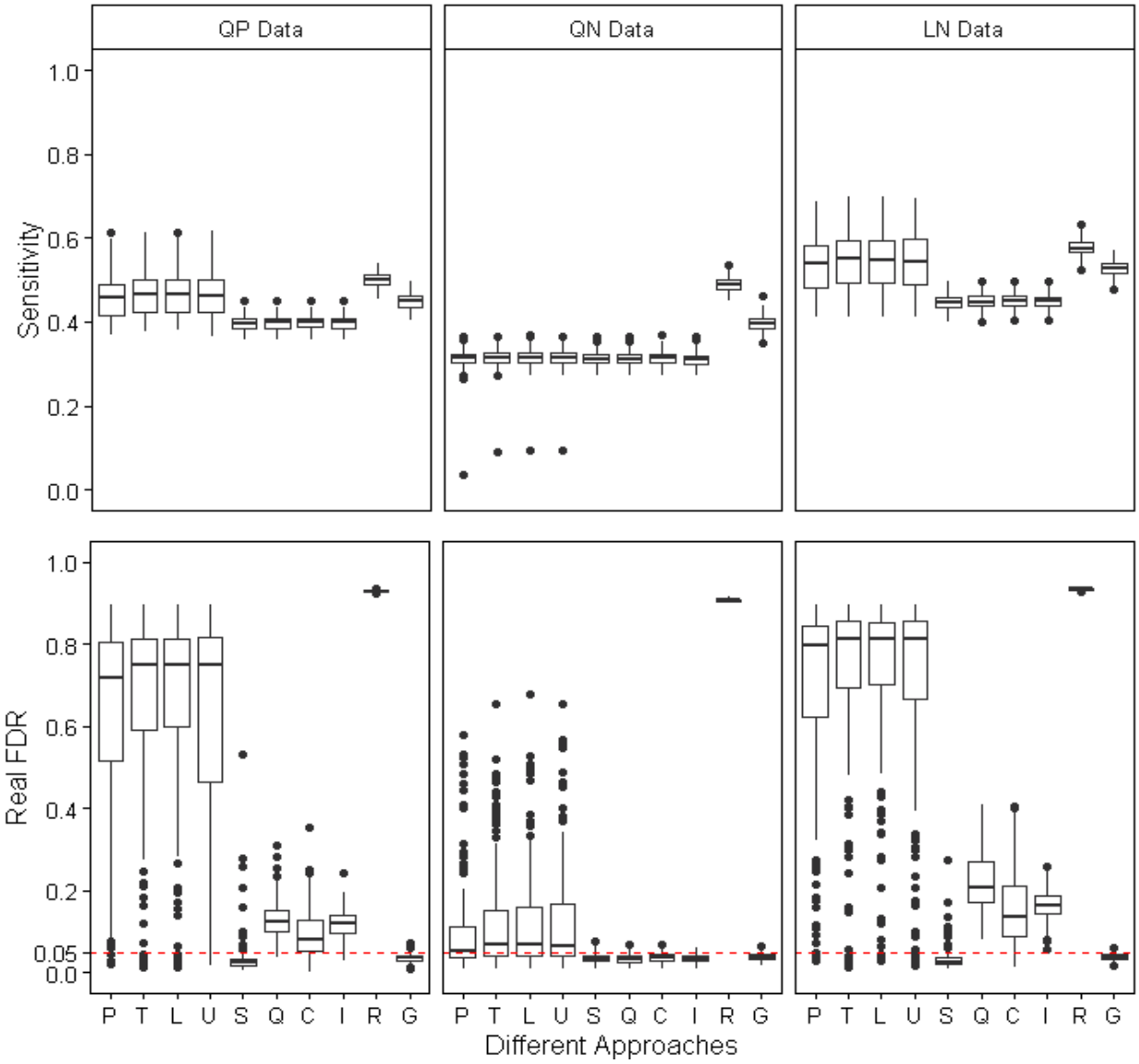

Figure C 8. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, Low Fold Change, Varied Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

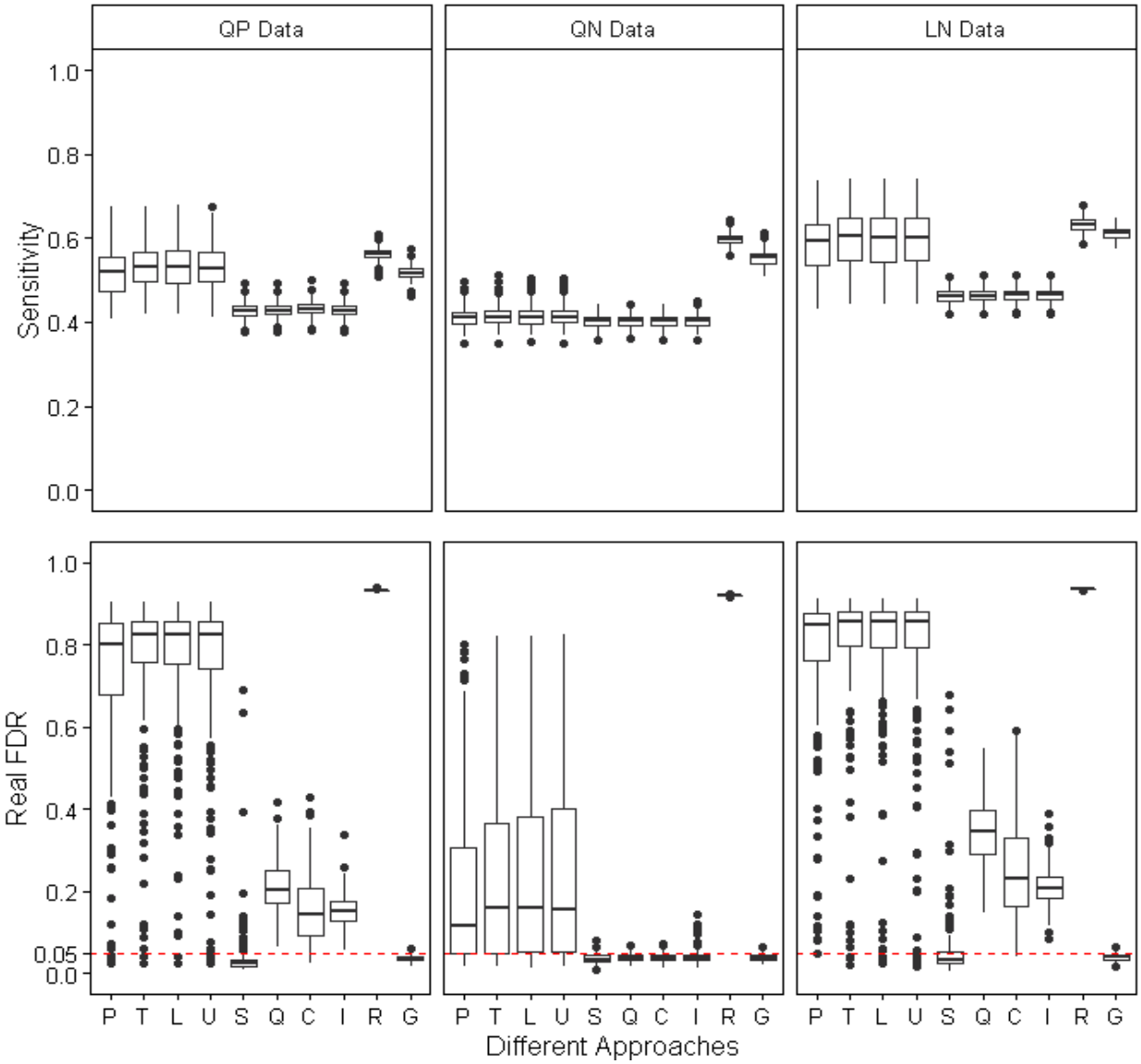

Figure C 9. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, Low Fold Change, Varied Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

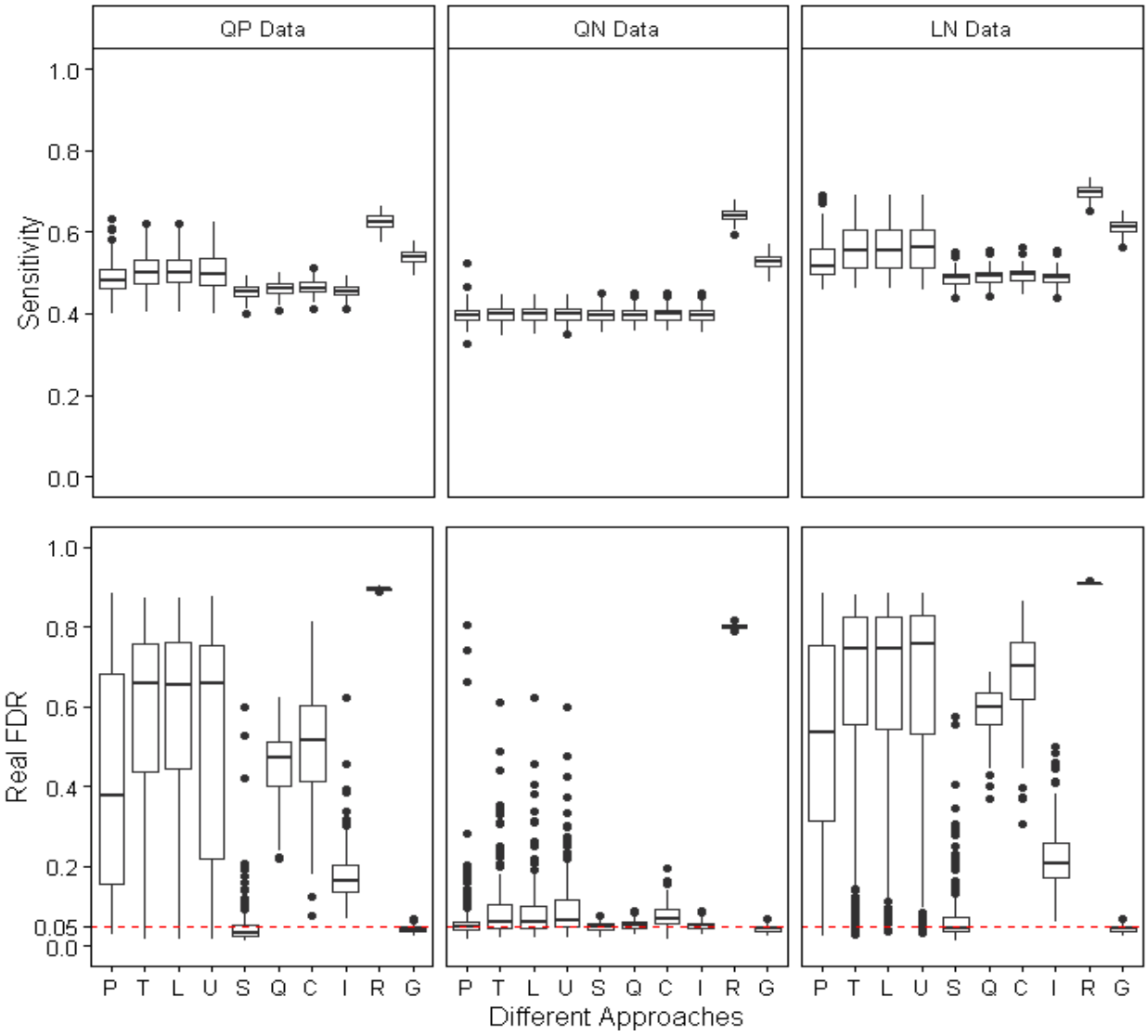

Figure C 10. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, High Fold Change, Varied Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

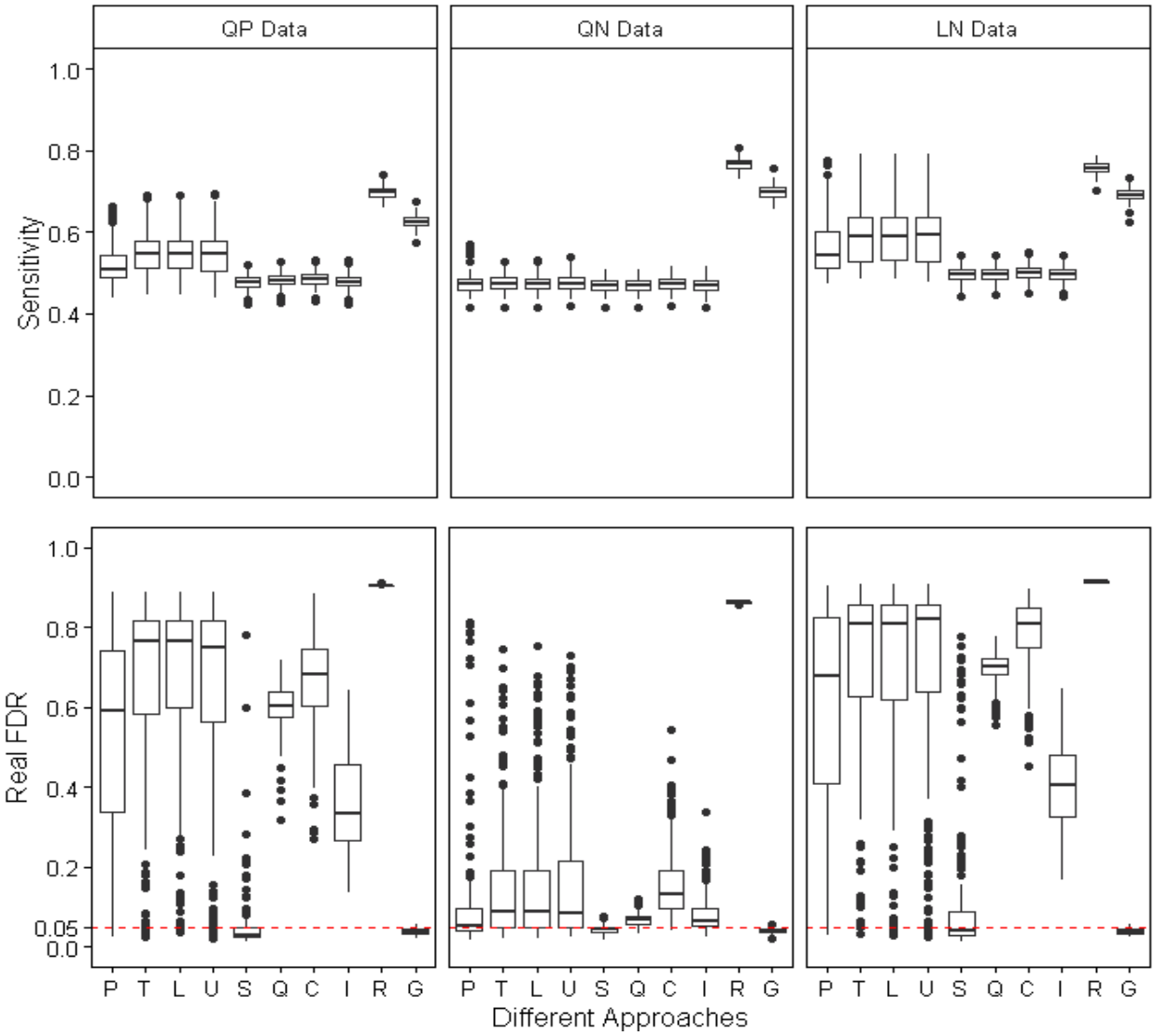

Figure C 11. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, High Fold Change, Varied Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

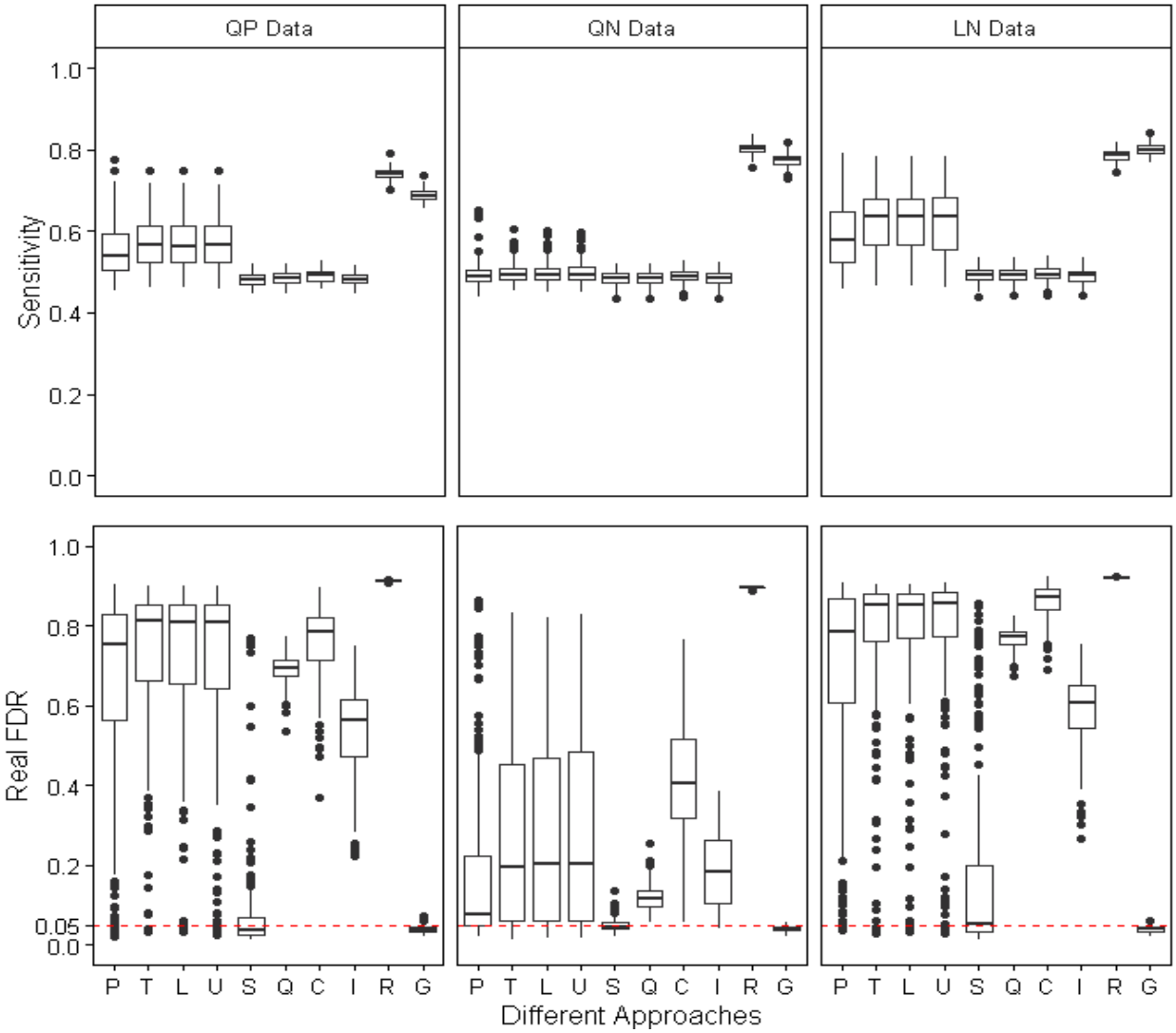

Figure C 12. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, High Fold Change, Varied Gene Length, LIMMA)

P = RPKM data without further normalization, T = RPKM data after the TMM method, L = RPKM data after the RLE method, U = RPKM data after the UQ method, S = RPKM data after the scaling normalization, Q = RPKM data after the quantile normalization, C = RPKM data after the cyclic loess normalization, I = RPKM data after the invariant set normalization, R = Raw count data with the TMM method, G = Raw count data with the gene length and TMM normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

APPENDIX D

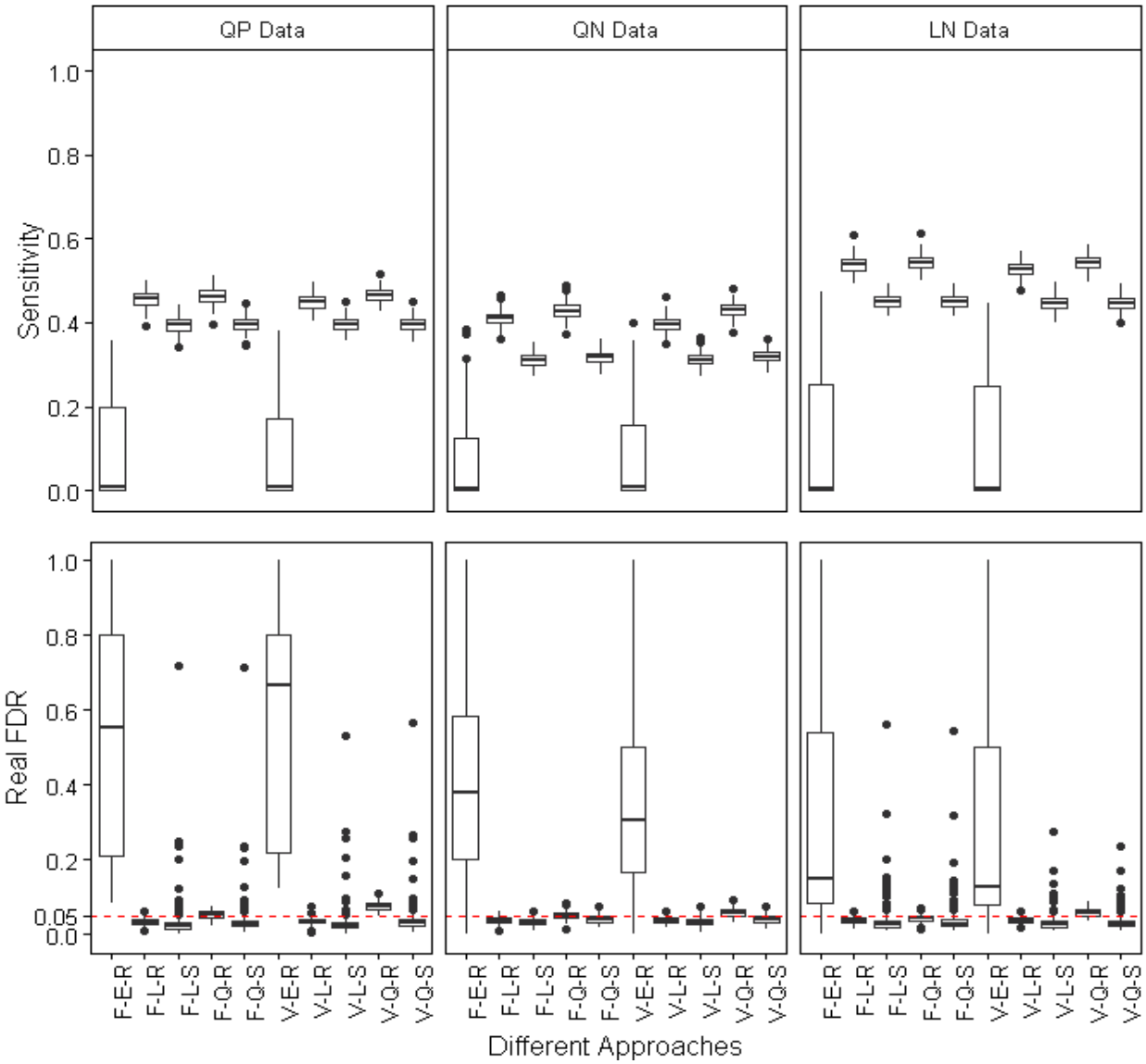

Figure D 1. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, Low Fold Change)

The first part of the labels on horizontal axis indicates the gene length setting: F = Fixed Gene Length; V = Varied Gene Length. The second part indicates the statistical method: E = edgeR; L = LIMMA; Q = QuasiDE. The last part indicates the type of data: R = Raw count data normalized by the TMM method when gene length is fixed; Raw count data normalized by the gene length and the TMM method when gene length is varied. S = RPKM data with the scaling normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

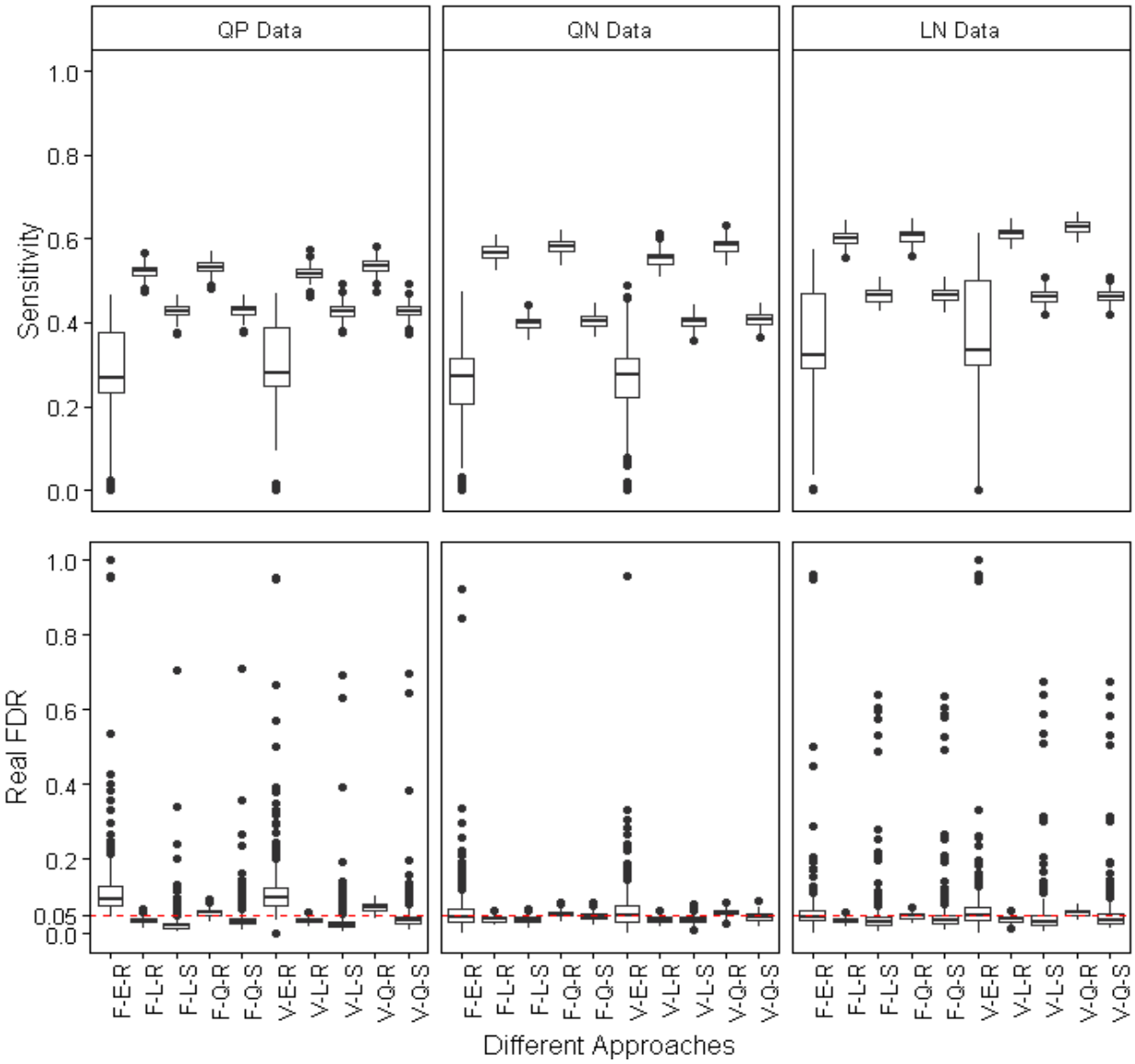

Figure D 2. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 20 vs. 20, Low Fold Change)

The first part of the labels on horizontal axis indicates the gene length setting: F = Fixed Gene Length; V = Varied Gene Length. The second part indicates the statistical method: E = edgeR; L = LIMMA; Q = QuasiDE. The last part indicates the type of data: R = Raw count data normalized by the TMM method when gene length is fixed; Raw count data normalized by the gene length and the TMM method when gene length is varied. S = RPKM data with the scaling normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

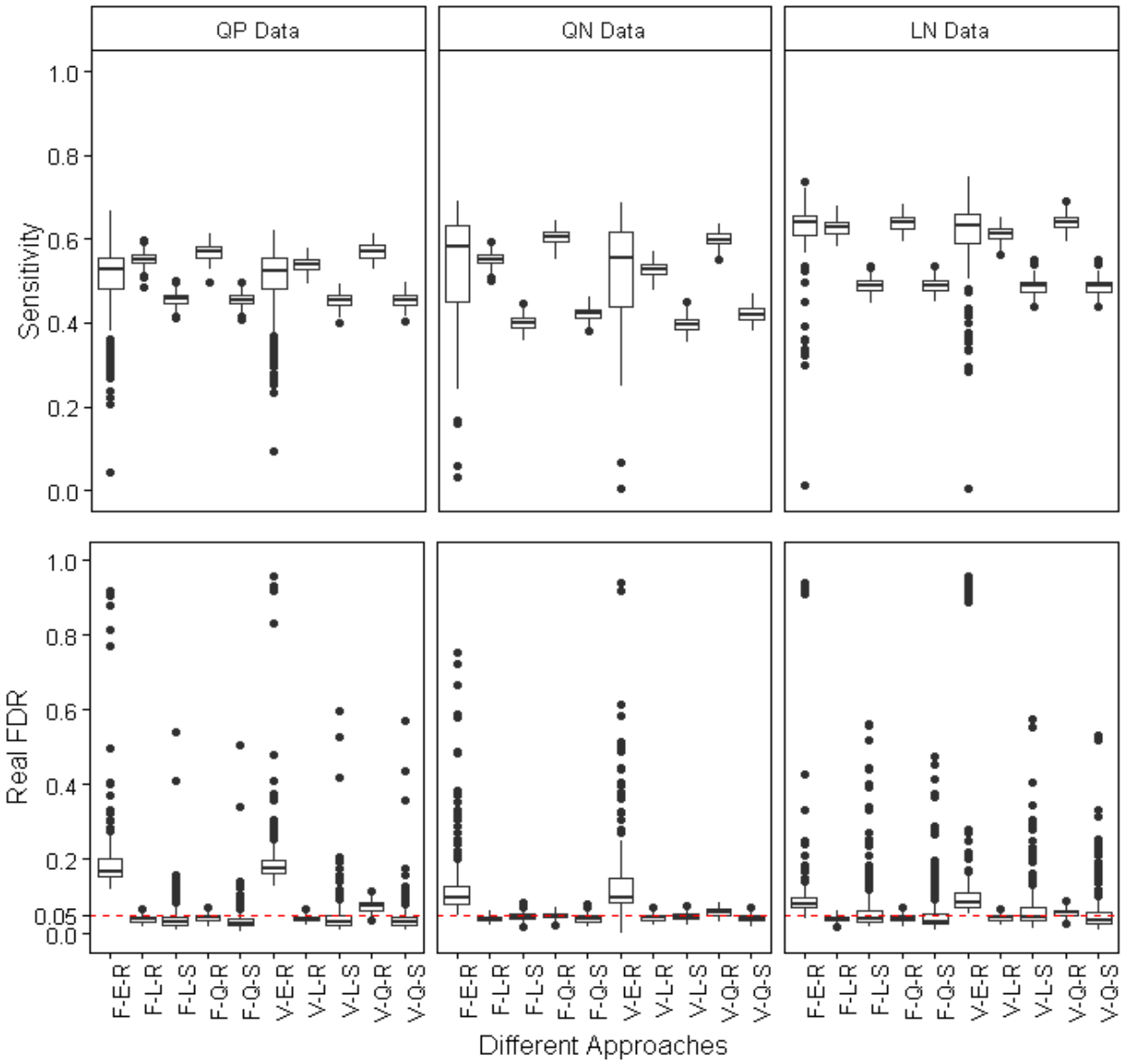

Figure D 3. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 5 vs. 5, High Fold Change)

The first part of the labels on horizontal axis indicates the gene length setting: F = Fixed Gene Length; V = Varied Gene Length. The second part indicates the statistical method: E = edgeR; L = LIMMA; Q = QuasiDE. The last part indicates the type of data: R = Raw count data normalized by the TMM method when gene length is fixed; Raw count data normalized by the gene length and the TMM method when gene length is varied. S = RPKM data with the scaling normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.

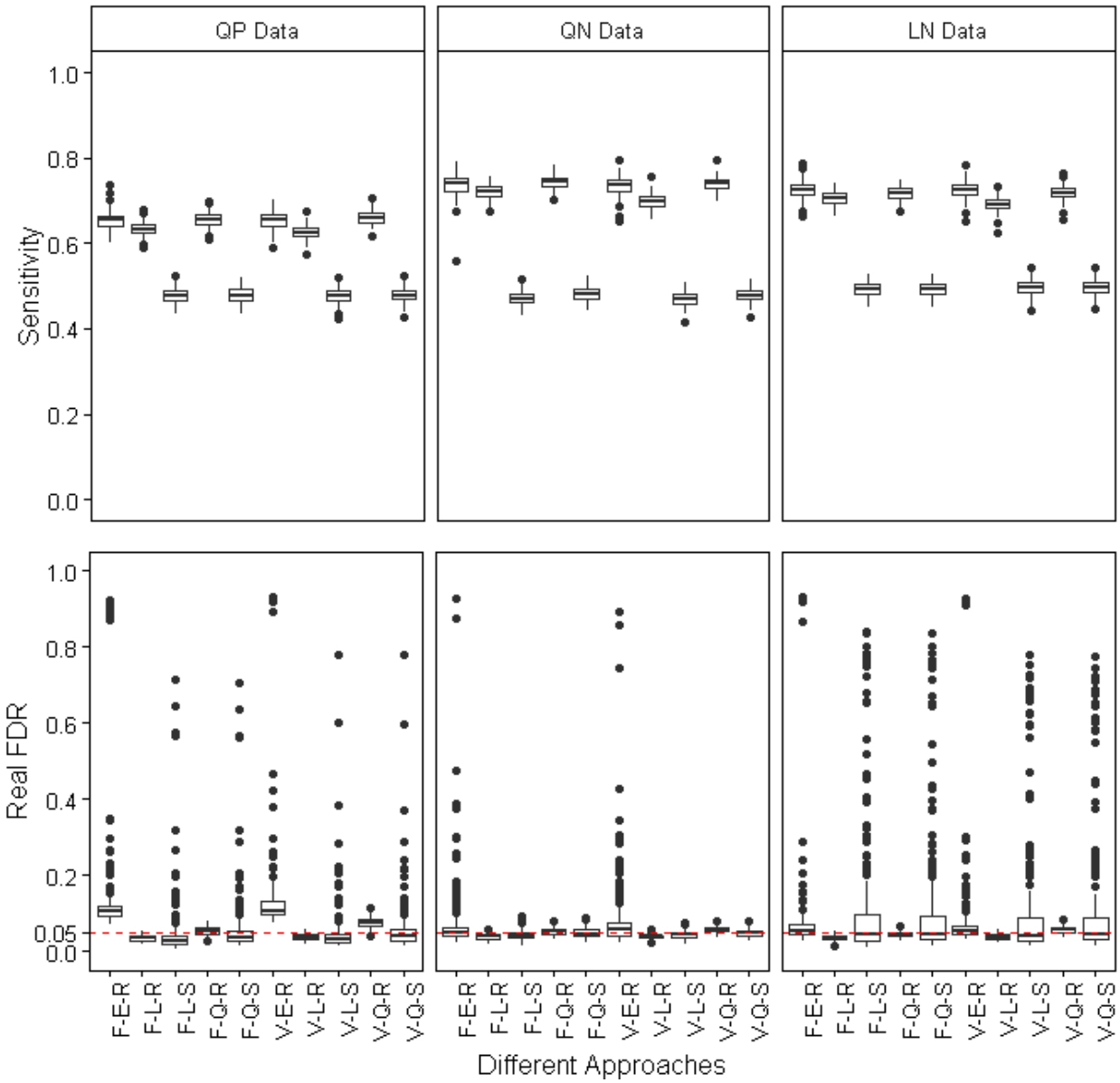

Figure D 4. Sensitivity and Real FDR by Different Approaches for Different Types of Data (Sample Size 10 vs. 10, High Fold Change)

The first part of the labels on horizontal axis indicates the gene length setting: F = Fixed Gene Length; V = Varied Gene Length. The second part indicates the statistical method: E = edgeR; L = LIMMA; Q = QuasiDE. The last part indicates the type of data: R = Raw count data normalized by the TMM method when gene length is fixed; Raw count data normalized by the gene length and the TMM method when gene length is varied. S = RPKM data with the scaling normalization. In the second row of panels, the red reference line is the nominal FDR we are willing to allow.