2016

Developing An Alternative Way to Analyze NanoString Data

Shu Shen
University of Kentucky, shu.shen@uky.edu
Digital Object Identifier: http://dx.doi.org/10.13023/ETD.2016.385

STUDENT AGREEMENT:

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

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

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

REVIEW, APPROVAL AND ACCEPTANCE

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

Shu Shen, Student

Dr. Arnold J.Stromberg, Major Professor

Dr. Constance Wood, Director of Graduate Studies

Developing An Alternative Way to Analyze NanoString Data

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Arts and Sciences
at the University of Kentucky

By
Shu Shen
Lexington, Kentucky

Director: Dr. Arnold J.Stromberg, Professor of Statistics
Lexington, Kentucky

2016

ABSTRACT OF DISSERTATION

Developing An Alternative Way to Analyze NanoString Data

Nanostring technology provides a new method to measure gene expressions. It's more sensitive than microarrays and able to do more gene measurements than RT-PCR with similar sensitivity. This system produces counts for each target gene and tabulates them. Counts can be normalized by using an Excel macro or nSolver before analysis. Both methods rely on data normalization prior to statistical analysis to identify differentially expressed genes. Alternatively, we propose to model gene expressions as a function of positive controls and reference gene measurements. Simulations and examples are used to compare this model with Nanostring normalization methods. The results show that our model is more stable, efficient, and able to control false positive proportions. In addition, we also derive asymptotic properties of a normalized test of control versus treatment.

KEYWORDS: NanoString nCounter data, normalization, linear model, asymptotic properties

Author's Signature: Shu Shen

Date: July 28, 2016

Developing An Alternative Way to Analyze NanoString Data

By
Shu Shen

Director of Dissertation: Dr. Arnold J.Stromberg

Director of Graduate Studies: Dr. Constance Wood

Date: July 28, 2016

ACKNOWLEDGMENTS

I would like to express my deepest appreciation and gratitude to my advisors, Dr. Arnold J.Stromberg, for his guidance, insights and support that made possible this work. And also I would like to thank Dr. Constance Wood, without her patient help and brilliant ideas, I would never have got the theoretical part done.

Next, I wish to thank other committee members: Dr. Heather Bush, Dr. William Griffith, Dr. Ruriko Yoshida and Dr. Timothy McClintock, for their friendly comments and advice.

Finally I would like to thank my parents, their love and support helped me overcome so many difficulties. And I'm also very grateful to my dearest friends, for their encouragement and precious suggestions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Overview

The NanoString nCounter platform was developed by NanoString Technologies as a new technology to do gene expression studies. Comparing to traditional microarray and RT-PCR, besides the ability of dealing more samples without reducing sensibility, the system also shows excellent robustness and reproducibility(Malkov et al., 2009; Veldman-Jones et al., 2015a; Veldman-Jones et al., 2015b). And it's flexible, fast and simple: researchers only need to add RNA samples to the platform and then press the start button, the system will produce the gene expressions automatically. This system is based on a novel digital molecular barcoding technology and all the outcomes are counts. We refer to the counts as NanoString nCounter data.

1.2 Data Production

The NanoString nCounter platform produces data in three steps according to nCounter Brochure(2015).

First step: Hybridization.
Each gene of interest is a target sequence and a corresponding capture probe and reporter probe are used to detect the target. The capture probe has a biotin part which is used to immobilize the sample and the reporter probe is bar coded for future counting. The capture probes and reporter probes for all the target sequences are hybridized to the sample which will create some tripartite structures: a target mRNA bound to its capture and reporter probes.

Second step: Sample processing.
This step is performed by the nCounter Prep Station: excess probes are removed and hybridized probes are bound to the cartridge.

Third step: Count.
Sample cartridges are placed in the nCounter Digital Analyzer for automatic digital counting. Barcodes are counted and tabulated for each target molecule. All the outputs are counts.

1.3 Data File

Tabulated output consists of three parts: file attributes, lane attributes and reporter counts.

File attributes contains the basic information of the file, such as file name, ID, owner and date while the lane attributes shows the information for each lane, such as lane ID and binding density.

Reporter counts is the most important part of the table and it also consists of three parts: positive control, negative control and endogenous, gene counts.

Different assay types have different numbers of positive controls and negative controls. Take mRNA gene expression as an example. Under such a condition, 6 positive controls and 8 negative controls are included. Each positive control is at one of the following concentrations: 128 fm, 32 fm, 8 fm, 2 fm, 0.5 fm and 0.125 fm. This range corresponds to most mRNA expression levels. The rest are endogenous genes. Be-

sides genes we are interested in, there are genes called reference genes. Those genes are pre-selected from a certain reference gene library provided by the company. They are expected to be similar for all samples.

1.4 Data Normalization

In gene expression study, normalization is a common method(Vandesompele et al., 2002; Bullard, 2010; Garmire, 2012). According to nCounter Expression Data Analysis Guide(2012), due to slight differences in data production, NanoString suggests normalizing the data by using the internal controls before data analysis. Recently, there are two ways to do the normalization. One is using an Excel macro and the other is using a software called nSolver.

1.4.1 Using Excel worksheet

Here are three steps if normalization is performed by using Excel worksheet based on nCounter Expression Data Analysis Guide(2012): positive control normalization, reference gene normalization and assessing background.

Positive control normalization
This step normalizes all the variation related to the platform.

1. For each lane, compute the summation of all the positive controls.
2. Compute the average of all the summations from previous step.
3. Divide this average by each lane's positive control summation to get that lane's positive control norm factor.
4. Multiply all the gene expressions in a lane by that lane's positive control norm

factor to get the positive control normalized gene expressions.

Reference gene normalization

Reference genes are pre-selected genes which are supposed not to express vary between samples. This step is used to adjust all the genes to these consistent reference genes. This step starts with the positive control normalized gene expressions.

1. For each lane, compute the geometric mean of the reference genes. The reason for using geometric mean is reference genes are always expressed at difference levels and geometric mean is less sensitive to the variation.
2. Compute the average of all the geometric means from previous step.
3. Divide this average by each lane's reference gene geometric mean to get that lane's reference gene norm factor.
4. Multiply all the gene expressions in a lane by that lane's reference gene norm factor to get the reference gene normalized gene expressions.

Assessing background

Negative controls are probes for which no target is expected to be present and thus, they can be used to estimate the background of the experiment. Several ways are introduced to determine the background, such as mean of negative controls or mean plus standard deviation of negative controls. Once the background threshold is decided, subtract them from the gene expressions will complete the normalization process.

There are things to note when normalization is performed by using Excel worksheet. For example, in the positive control normalization step, instead of using the summation of positive controls, the official user guide also lists average or geometric mean of positive controls as alternative options.

1.4.2 Using nSolver v1.1

nSolver v1.1 which is developed by NanoString, can be used to do normalization and other analyses.

The first step is importing data and selecting the data type. Two kinds of files can be imported through the main menu: RCC files and CodeSet RLF file.

After the raw data is imported, choose 'Study' from the drop down menu to start a new study page. In this page, we are able to specify the properties for this study, such as name, group, research area and description. Then select 'Experiment' from the drop down menu to set up a new experiment to start the data analysis. As the new page pops out, properties of the experiment can be specified.

Next step is adding samples. If different assay types of data are stored, select the appropriate type from the left of the page will help to locate the data quickly. Then select the 'CodeSet' from the navigation tree and all the samples will show up in the table view. If here are some samples need to be excluded, select them and press the 'Exclude Selected' button, then these samples will become hidden. All samples visible in the table view will be used for the experiment when the 'Next' button is pressed. If anything need to be specified for the samples, we can also sample annotations.

Then the following step is selecting normalization type and parameters for the experiment. Two normalization types are available: Positive Control Normalization and CodeSet Content Normalization which is using the reference genes. nSolver v1.1 makes these two kinds of normalization optional: we can choose to use both or just

one or even neither of them. Suppose we want to use both to normalize the raw data and next we will select the normalization parameters.

For the positive control normalization, positive controls show up automatically in a window view, but besides using them all, nSolver makes it possible to choose part of them to compute the norm factor. And two options are provided for norm factor computation: mean and geometric mean. Once this is decided, we should choose the parameters for the CodeSet Content Normalization. Unlike the positive controls, if the reference genes are not marked as 'House Keeping' in the raw data set, they will not show up automatically. In this case, we have to pick those reference genes and transfer them into the Normalization Codes window. Once again, both mean and geometric mean can be used to compute the norm factor.

When everything is set, last two steps are to specify baseline data for creating fold change estimates if 'Build ratio' is selected and to assign names to the fold change data. Experiment results are listed in the left side of the window and they could be exported in different formats.

1.4.3 Formulize the normalization process

In section 1.4.1 and 1.4.2, we showed how to do normalization step by step by using Excel workflow and nSolver v1.1. In this section, we will formulize the positive control normalization and the reference gene normalization steps which are performed in Excel workflow.

Denotation:

Suppose we have m samples. Denote positive controls as p_{ij}, negative controls as n_{ij}, reference genes as r_{ij} and non-reference genes as g_{ij} for row/gene i and sample j.

Positive Control Normalization

Raw positive controls

$$\begin{pmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{61} & \cdots & p_{6m} \end{pmatrix}$$

1. Summation for the jth lane:

$p_{.j} = p_{1j} + \cdots + p_{6j}, j = 1, ..., m$

2. Average of all the summations:

$\frac{\sum_j p_{.j}}{m} = \frac{p_{..}}{m}$, $p_{..} = \sum_{ij} p_{ij}$

3. Norm factor for each lane:

$\frac{p_{..}}{mp_{.j}}, j = 1, ..., m$

Suppose here are t non-reference genes and s reference genes, then the positive control normalized gene expressions would be:

$$\begin{pmatrix} g_{11} \times \frac{p_{..}}{mp_{.1}} & \cdots & g_{1m} \times \frac{p_{..}}{mp_{.m}} \\ \vdots & \ddots & \vdots \\ g_{t1} \times \frac{p_{..}}{mp_{.1}} & \cdots & g_{tm} \times \frac{p_{..}}{mp_{.m}} \\ r_{11} \times \frac{p_{..}}{mp_{.1}} & \cdots & r_{1m} \times \frac{p_{..}}{mp_{.m}} \\ \vdots & \ddots & \vdots \\ r_{s1} \times \frac{p_{..}}{mp_{.1}} & \cdots & r_{sm} \times \frac{p_{..}}{mp_{.m}} \end{pmatrix}$$

Reference Gene Normalization

1. Geometric mean of reference genes for the jth lane:

$\frac{p_{..}}{mp_{.j}} \times (\prod_{i=1}^{s} r_{ij})^{\frac{1}{s}}, j = 1, ..., m$

2. Average of all the geometric means:

$\frac{p_{..}}{m^2} \sum_{j=1}^{m} \frac{r_j}{p_{.j}}$, where $r_j = (\prod_{i=1}^{s} r_{ij})^{\frac{1}{s}}, j = 1, ..., m$

3. Norm factor for each lane j:

$\frac{p_{.j}}{mr_j} \sum_{j=1}^{m} \frac{r_j}{p_{.j}}, j = 1, ..., m$

Then the reference gene normalized gene expressions would be:

$$\begin{pmatrix} g_{11} \times \frac{p_{..}}{m^2 r_1} \times \sum_{j=1}^{m} \frac{r_j}{p_{.j}} & \cdots & g_{1m} \times \frac{p_{..}}{m^2 r_m} \times \sum_{j=1}^{m} \frac{r_j}{p_{.j}} \\ \vdots & \ddots & \vdots \\ g_{t1} \times \frac{p_{..}}{m^2 r_1} \times \sum_{j=1}^{m} \frac{r_j}{p_{.j}} & \cdots & g_{tm} \times \frac{p_{..}}{m^2 r_m} \times \sum_{j=1}^{m} \frac{r_j}{p_{.j}} \\ r_{11} \times \frac{p_{..}}{m^2 r_1} \times \sum_{j=1}^{m} \frac{r_j}{p_{.j}} & \cdots & r_{1m} \times \frac{p_{..}}{m^2 r_m} \times \sum_{j=1}^{m} \frac{r_j}{p_{.j}} \\ \vdots & \ddots & \vdots \\ r_{s1} \times \frac{p_{..}}{m^2 r_1} \times \sum_{j=1}^{m} \frac{r_j}{p_{.j}} & \cdots & r_{sm} \times \frac{p_{..}}{m^2 r_m} \times \sum_{j=1}^{m} \frac{r_j}{p_{.j}} \end{pmatrix}$$

After these two steps of normalization, the original gene expression g_{ij} is transformed to $g_{ij} \times \frac{p_{..}}{m^2 r_j} \times \sum_j \frac{r_j}{p_{.j}}$, where $p_{..} = \sum_{ij} p_{ij}$, $r_j = (\prod_{i=1}^{s} r_{ij})^{\frac{1}{s}}, j = 1, ..., m$

1.5 Summary

Normalization, which is introduced by NanoString, is used to correct the differences caused by the data production steps. The above two ways are used to use to achieve the goal: using Excel worksheet is more intuitive while using nSolver v1.1 is simpler and faster.

One thing to note is nSolver v1.1 does not have "assessing background" step and the reason for this is the company thinks there is no single appropriate algorithm to estimate background and it is not necessarily applicable for all downstream analyses.

Therefore if we normalize a raw data set by using both Excel and nSolver v1.1, we will get different results. Even if we only apply the first two normalization steps and choose the same normalization parameters for both methods, we will not expect the exactly the same normalized data. This is because there are some minor differences in the algorithm used by nSolver v1.1 such as the number of significant digits used in the calculations. All of these reveals a problem of normalization: the normalized data relies a lot on the parameters and method chosen by researchers which means we cannot expect a consistent result.

Take a real data set for example. In this 2x2 design experiment, 12 cell samples are used to measure the expressions of 236 genes. Table 1.1 lists the differences of significant genes(p-value<0.05) for two kinds of normalized data: the first one is normalized without background correction and the second one is normalized with background correction. Both of them are done by using Excel Workflow.And from this table, we could see the significant genes are not exactly the same based on these two differently normalized data sets.

Table 1.1: This table shows the numbers of significant genes for overall ANOVA and 6 pairwise T-test comparisons

	no background correction	with background correction	significant genes in common
overall	145	144	133
T-test 1	104	98	96
T-test 2	76	78	72
T-test 3	77	79	73
T-test 4	89	89	83
T-test 5	125	140	118
T-test 6	81	78	72

It's possible that the differences of the normalized data are too minor to cause a huge different in the results. But we don't want the analysis to be affected by unstable normalized data and this brings us an idea of looking for an alternative way to analyze

the NanoString data without normalization. This new way should be more consistent and reliable. In this case, building a model to pick up those differentially expressed (DE) genes is a good way to proceed.

1.6 Methods

In the following study, we will focus on the simplest and also the most popular experimental design, a two sample comparison: gene expressions are presented under two different conditions, let's say treatment and control. The usual way of doing analysis is after normalization, a two sample t-test is performed for each gene. Then some other rules are accordingly applied to the p-value list to identify DE genes, such as False Discovery Rate (FDR) control or a fold change (FC) cutoff.

False Discovery Rate Control

Table 1.2: This table shows the definition of m hypothesis tests

	H_0 is True	H_1 is True	Total
Declared significant	V	S	R
Declared non-significant	U	T	m-R
Total	m_0	m-m_0	m

Based on definition Table 1.2, the proportion of false discoveries among the discoveries is $\frac{V}{R}$, and the false discovery rate is $E(\frac{V}{R})$, where $\frac{V}{R}$ is defined to be 0 when R=0. And we want to keep this value below a certain threshold. False discovery rate (FDR) control is a statistical method used in multiple hypothesis testing to correct for multiple comparisons. It's designed to control the expected proportion of incorrectly rejected null hypotheses ("false discoveries"). In gene studies, it's used to control the rate of non DE genes among detected changes. In order to control FDR for multiple testing, two general types of adjustment were developed: one is a single-step adjustment which is accessing significance of all p-values with a single criterion and another one is step-

down adjustment which is applying different criterion for each of the ordered p-values.

Single-step adjustment example

Bonferroni adjustment:

To keep the overall significant level still at α, use $\alpha^{'} = \frac{\alpha}{m}$ as the significant level for the individual tests. This adjustment is the most naive way to address the issue, but it's too conservative.

Step-down adjustment example

Benjamini- Hochberg procedure (Benjamini, Yoav and Hochberg, Yosef, 1995):

Consider multiple testing $H_1, ..., H_m$ and their corresponding p-values $P_1, ..., P_m$.

1. Order p-values in an increasing order and denote them as $P_{(1)}, ..., P_{(m)}$ and their corresponding hypothesis testing as $H_{(1)}, ..., H_{(m)}$.
2. For a given level q, let k be the largest i such that $P_{(i)} \leq \frac{i}{m}q$.
3. Reject all $H_{(i)}, i = 1, ..., k$.

This procedure controls the FDR at level q when those m tests are independent and also in various scenarios of dependence.

Besides these methods, other control procedures have also been introduced for gene expression data(Benjamini et al., 2000; Fernando et al., 2004; Benjamini et al., 2006).

Fold Change

A fold change is a measurement to describe how much a quantity changes going from an initial to a final value. When speaking of the fold change of a gene, the standard definition is(Witten et al., 2007)

$$FC_i = \frac{\overline{x_{i.}}}{\overline{y_{i.}}}$$

where x_{ij} and y_{ij} are the raw gene expressions for gene i in replicate j in treatment group and control group. And a value greater than 1 indicates an increase in gene expression and a value smaller than 1 means a decrease in gene expression. Much like p-values, there are some cutoff rules to help with picking up DE genes(Dalman et al., 2012). And methods which use both p-values and fold changes to identify significant genes have also been developed(Peart et al., 2005; Patterson et al., 2006; Huggins et al., 2008; Mccarthy et al., 2009).

p-value histogram

In gene studies, after all the p-values are obtained, drawing a p-value histogram is a good way to see if any problems exist in the analysis. What we expect is a right skewed histogram which indicates that there are some differentially expressed genes. A uniformly distributed p-value histogram indicates very few, if any, genes are differentially expressed. Usually these two shapes do not indicate any violation of the assumptions of methods applied to the data. If we see a histogram has one or more humps or it is left skewed, this would suggest an inappropriate statistical method was used to compute p-values. And in gene expression study, a p-value histogram can also be used to estimate false discovery rate(Storey and Tibshirani, 2003; Nettleton et al., 2006)

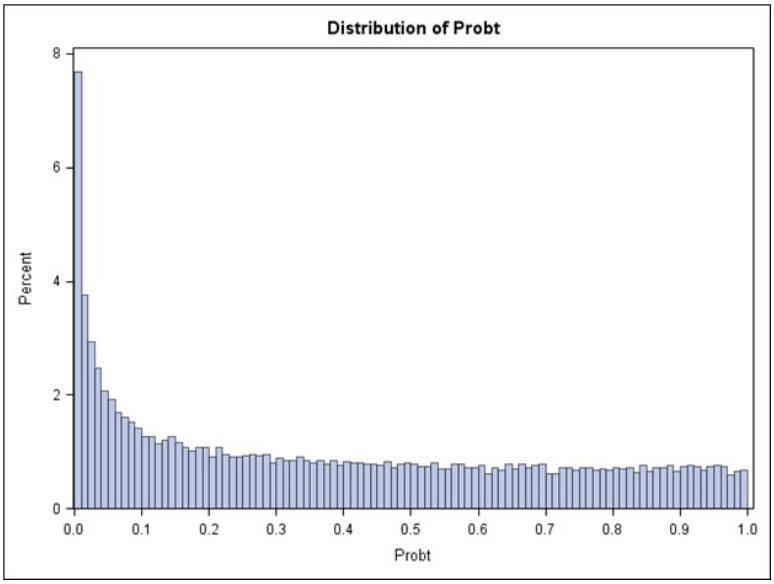

Figure 1.1: Right skewed p-value histogram.

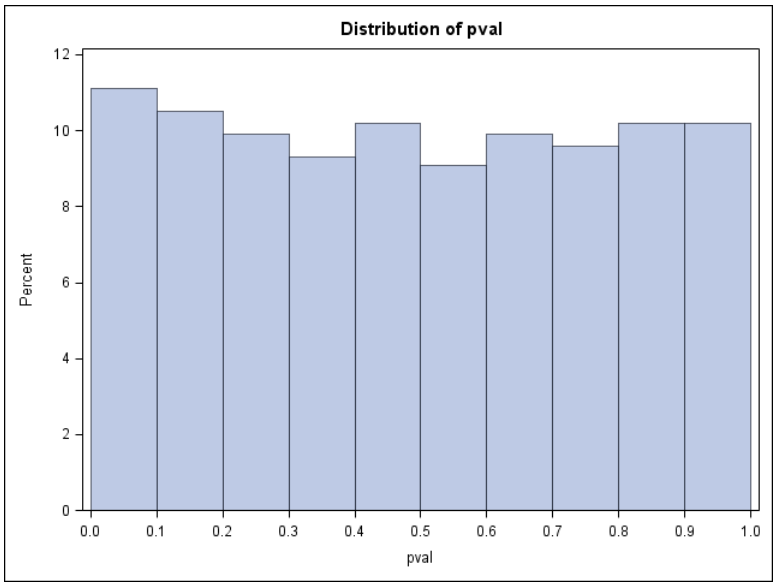

Figure 1.2: Uniformly distributed p-value histogram.

False discovery rate control(Pounds, 2006), fold change and p-value histogram are all general ways to deal with p-values in gene expression study. And the challenge here is NanoString technology has only been used for a few years, therefore studies for the properties of this kind of data are really rare. Most publications are using normalization and statistical approaches for data analysis(Vaes et al., 2014; Lee et al., 2015). And some particular R packages are also based on normalization and t-test, such as NanoStriDE(Brumbaugh, 2011) and NanoStringNorm(Waggott, 2012). Our goal is to build a general statistical model, so existing models developed for other types of gene expression data could be improved.

For example, linear model(Smyth, 2005; Wettenhall, James M and Smyth, Gordon K, 2004; Wettenhall et al., 2006), mixture model(Pan et al., 2003) and etc have been developed for Microarray data. And among these, linear model is a widely used one. Suppose y_{ij} is the *log* transformed gene expression of gene i in array $j(i = 1, ..., n; j = 1, ..., p, p+1, ..., p+q)$. The first p arrays are controls and the last q arrays are treatments. A general statistical model is

$$y_{ij} = a_i + b_i x_j + \epsilon_{ij}$$

where $x_j = 1$ for control arrays and $x_j = 0$ for treatment arrays, and ϵ_{ij} are random errors with mean 0. To determine if a gene is differentially expressed, it's equivalent to test

$$H_0 : b_i = 0 \ vs \ H_1 : b_i \neq 0$$

Chapter 2 Linear Model

Our basic idea is to use a linear model to detect DE genes for NanoString data. For each gene, fit a linear model to the raw data.

$$y_{ij} = \alpha + \beta_1 trt_i + \beta_2 pos_{ij} + \beta_3 ref_{ij} + \epsilon_{ij}$$

Let i denote the group type: $i = 1$ for control groups and $i = 2$ for treatment groups. For group i, let j denote the jth sample and $j = 1, ..., n_i$. y_{ij} is the corresponding raw gene expression. Let trt_1 indicate control groups and trt_2 indicate treatment groups. Set $trt_1 = 0$ and $trt_2 = 1$. pos_{ij} is the summation of positive controls for group i and sample j. ref_{ij} is the geometric mean of reference genes for group i and sample j. Assume genes are mutually independent and $\epsilon_{ij} \overset{i.i.d.}{\sim} N(0, \sigma^2)$. And detecting DE genes is equivalent to test:

$$H_0 : \beta_1 = 0 \ vs \ H_1 : \beta_1 \neq 0$$

2.1 Simulated and real data

In order to compare the traditional method (first normalize the raw data and then perform two sample t-test) with this linear model, we will use simulated and real data to check their performances. For simulated data, we're able to define and control DE genes and check the accuracy of both methods. For real data, since the true DE genes are unknown, we will see if the result obtained by linear model will be similar to the result derived by the traditional method.

2.1.1 Simulation 1

This simulation is a general one and it's used to check if the linear model works for the most common assumption for simulated data. We will generate 48 groups (half of them are control and the other half are treatment), 6 positive controls, 6 reference genes and 1000 non-reference genes. Negative controls will not be generated since we will only perform positive control normalization and reference gene normalization.

1. Simulate N (100, 1) for all positive controls.
2. Simulate N (100, 1) for all reference genes.
3. Fix the number of DE genes to be 250. Simulate N (100, 1) for control groups and N (μ, 1) for treatment groups. We use μ=100.5, 101 and 102 to see how the results will differ.
4. Simulate N (100, 1) for the rest genes.

After obtaining the simulated data, compute p-values(significant genes will be identified by using $\alpha = 0.05$) for all the genes by using both methods. Repeat this simulation 1000 times and count the number of rejected null hypotheses and false positives. Calculate the proportions of false positives by using $\frac{number\ of\ false\ positives}{number\ of\ rejected\ null\ hypotheses}$ and denote them as mSFDR for linear model method and nSFDR for normalization method.

Simulation results:

In the following graphs(from Figure 2.1 to Figure 2.6), here're some interesting patterns of mSFDR and nSFDR:

1. For each value of μ, the value of mSFDR is quite stable.
2. When μ increases, there is a left shift tendency of mSFDRs which means the linear

model picks less false positives. And also the range of mSFDRs shrinks.

3. Values of nSFDRs are quite spread out and even when the value of μ increases, this range shrinks, but it's still quite large comparing to the linear model result.

4. Comparing the distributions of the number of rejected H_0 for both methods, we can see on average, linear model tends to reject a few more null hypotheses.

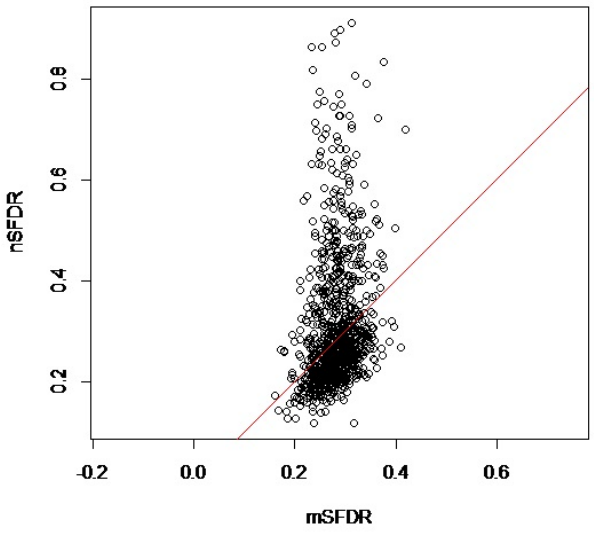

Figure 2.1: $\mu = 100.5$.

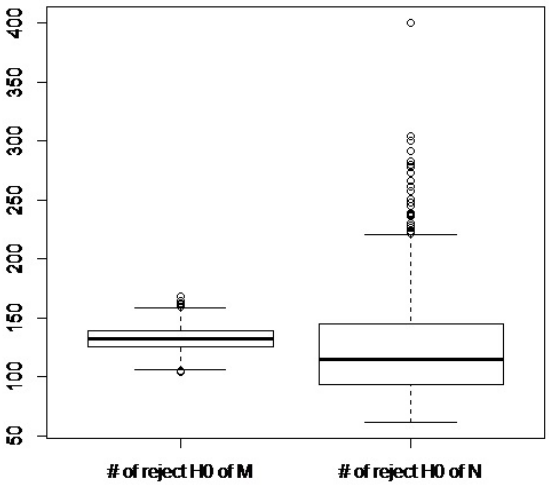

Figure 2.2: $\mu = 100.5$, M: model method; N: normalization method

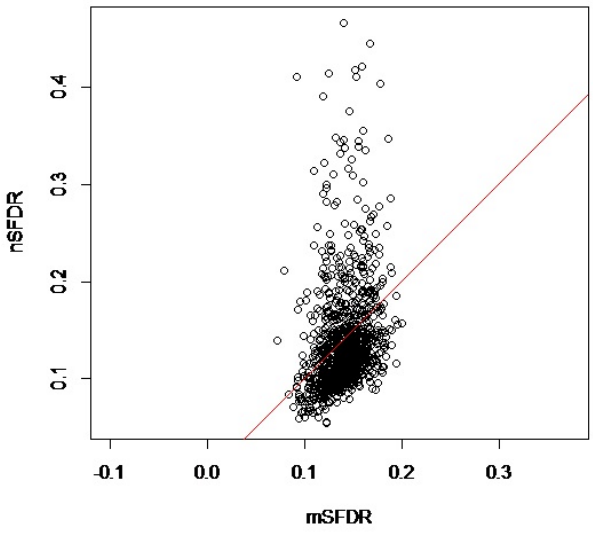

Figure 2.3: $\mu = 101$.

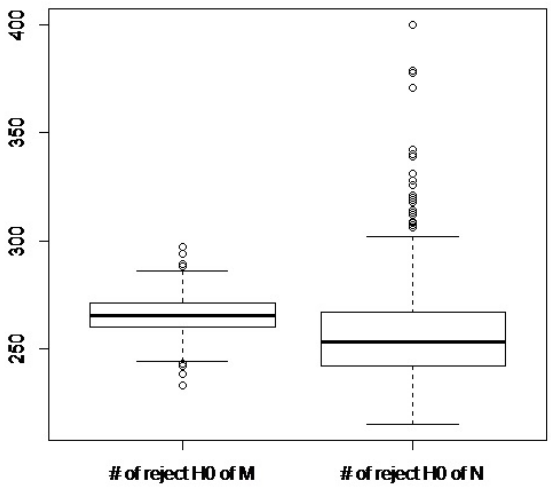

Figure 2.4: $\mu = 101$, M: model method; N: normalization method

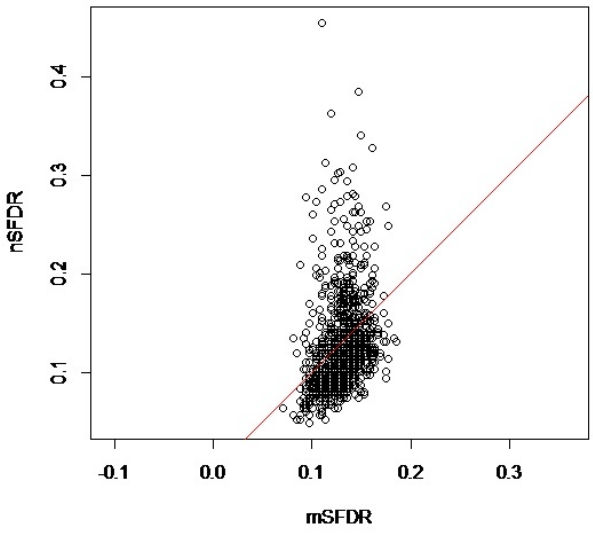

Figure 2.5: $\mu = 102$.

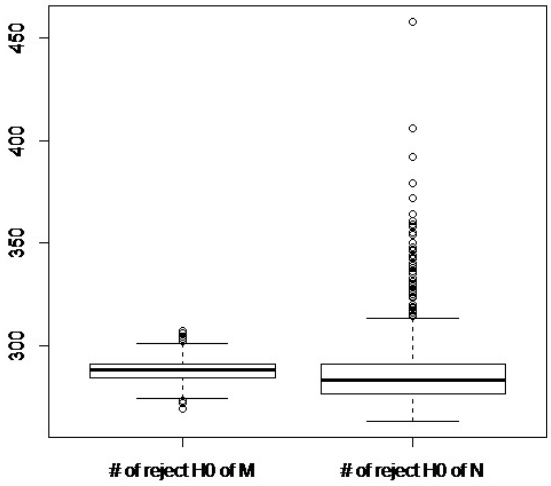

Figure 2.6: $\mu = 102$, M: model method; N: normalization method

As stated in previous chapter, when there are multiple hypothesis tests(in simulation 1, we perform testing 1000 times), in order to control the FDR, we need to do some adjustments. Next we will apply Bonferroni adjustment and B-H adjustment to p-values(here we use the p-values list for $\mu = 101$ as an example) and compare the results to see how the adjustments work(from Figure 2.7 to Figure 2.9).

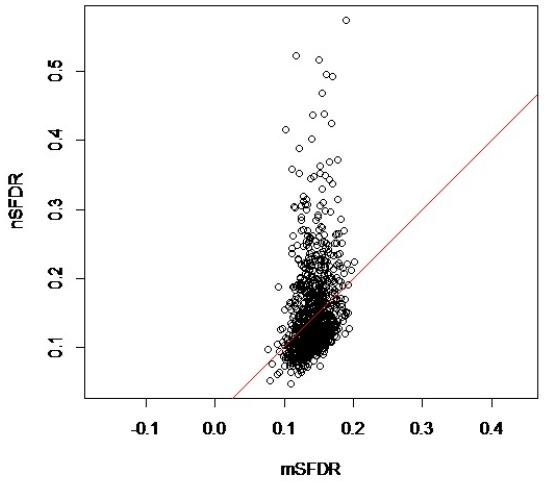

Figure 2.7: $\mu = 101$,un-adjusted p-values.

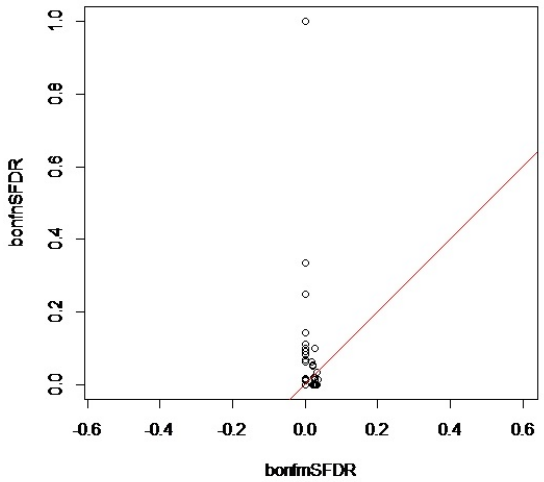

Figure 2.8: $\mu = 101$,Bonferroni adjusted p-values.

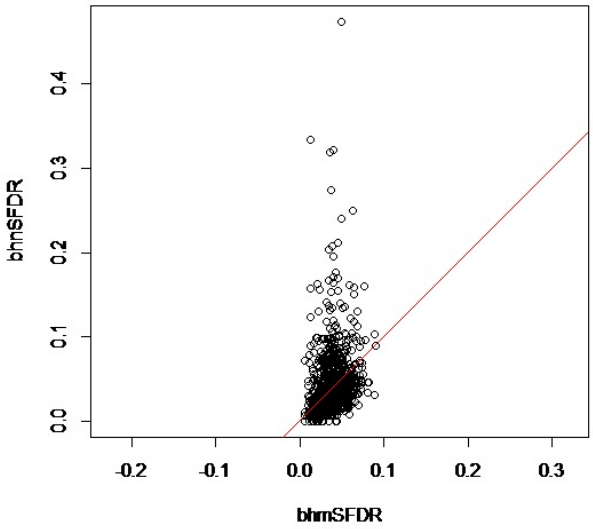

Figure 2.9: $\mu = 101$,B-H adjusted p-values.

In above three graphs, they still maintain the properties that mSFDRs are stable and nSFDRs are spreading out within a large range, no matter what p-values are used.

If we only consider the proportion of false positives, Bonferroni adjusted p-values did the best job. But the price is that way less significant genes were picked out (on average 41 genes were identified as significant by linear model and 32 by t-test). This shows how conservative Bonferroni adjustment is.

The ranges of mSFDRs given by B-H adjusted p-values and un-adjusted p-values are (0, 0.1) and (0.1, 0.2). Therefore, B-H adjustment did a good job in reducing the proportion of false positives. In order to see if these two kinds of p-values can pick up enough DE genes, the following graph(Figure 2.10) lists four boxplots for the numbers of true positives. It reveals that B-H adjusted p-values identifies less true positives than un-adjusted p-values, but considering the smaller proportion of false positives, B-H adjustment is still a reasonable.

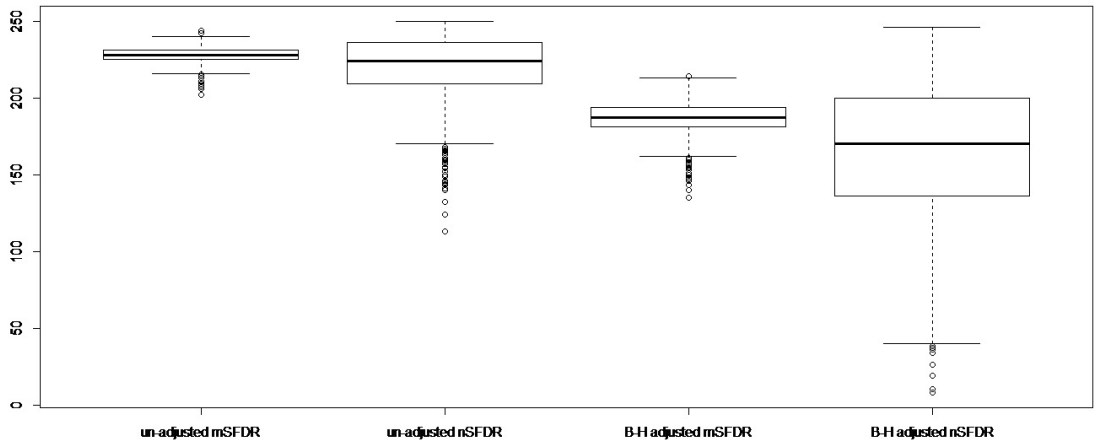

Figure 2.10: $\mu = 101$, number of true positives of different methods.

In simulation 1, we generated 1000 genes each time with a high proportion (25%) of DE genes, the proportion of false discoveries is used as a criteria to see which way is better. Results show that linear model is more stable and B-H adjusted p-value can help to reduce the proportion of false discoveries. One thing to note is that in real NanoString data, although it can test up to 800 genes, usually a data set only contains 100-200 genes with an unknown proportion of DE genes. And we would like to know when the numbers of genes and DE genes are limited, which way is more effective. We will generate 200 genes with 12.5% DE genes and $\mu = 101$ while other settings stay the same.

The following two graphs(Figure 2.11 and Figure 2.12) show that linear model picks up more true positives with stable proportion(between 0.1 and 0.4) of false positives.

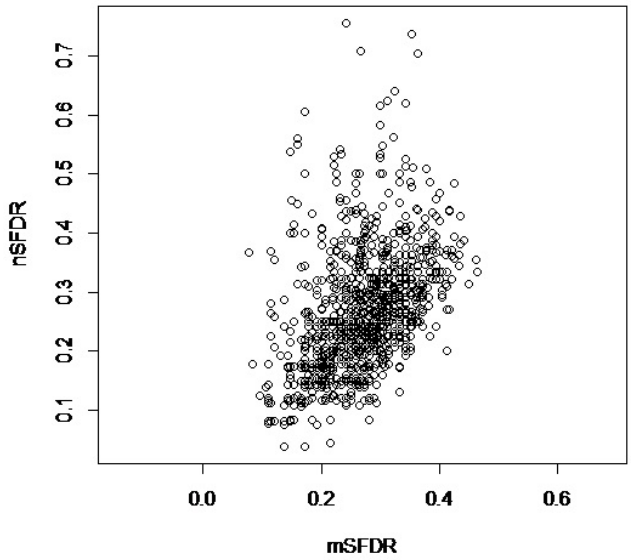

Figure 2.11: $\mu = 101$, 200 genes with 12.5% DE genes.

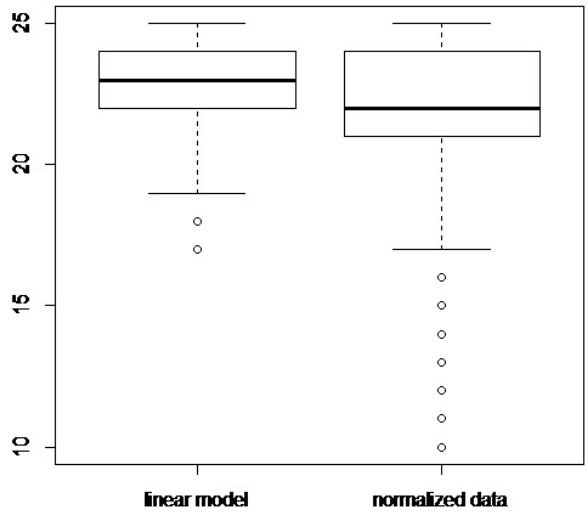

Figure 2.12: number of true positives.

2.1.2 Simulation 2

Simulation 1 doesn't take any properties of positive controls and reference genes into account. Such as for each lane, the values of 6 positive controls are listed in a decreasing order from pos1 to pos6. And besides this, here's another interesting property of those values. We know positive controls are at known concentrations: 128 fm, 32 fm, 8 fm, 2 fm, 0.5 fm and 0.125 fm. In those real data set examples, values for 0.5 fm and 0.125 fm are small and appear random. But for the rest concentrations, expression of 2 fm is approximately equal to 4*expression of 0.5 fm, expression of 8 fm is approximately equal to 4*expression of 2 fm, and so on. We would like to include this property in this new simulation which will make the simulated data closer to real data.

For reference genes, even they are supposed to express uniformly across all the lanes, but each of them is always expressed at different levels. And the same thing happens to non-reference genes too.

Although NanoString nCounter platform can measure up to 800 genes, but in real data, wetypically only have seen about 150-200 genes and the sample size is not as many as simulation 1. All of these will be considered in this new simulation.

Simulation settings:
12 control groups and 12 treatment groups
1. Positive controls: for each lane,
expression of 0.125 fm=a random number from 1 to 10
expression of 0.5 fm =expression of 0.125 fm + a random number from 1 to 5
expression of 2 fm=4* expression of 0.5 fm + a random number from 1 to 10
expression of 8 fm=4* expression of 2 fm + a random number from 1 to 10
expression of 32 fm=4* expression of 8 fm + a random number from 1 to 100

expression of 128 fm=4* expression of 32 fm + a random number from 1 to 100
This ensures positive controls are following an increasing pattern as the concentration increases.

2. Reference genes: They are typically expressed at different levels. We define 4 levels: 1-99, 100-999, 1000-9999, 10000-99999 and randomly assign reference genes to these levels. Once the levels are determined, generate random numbers from each level to make them the expressions of that reference gene across all the lanes. This method will make reference genes look like those in real data, but will not guarantee each reference gene express uniformly across all the lanes.

3. DE genes: 30 DE genes in total. For each of them, generate a random number from 1 to 1000 and denote it as μ. Simulate $N(\mu, 1)$ for control groups and $N(\mu + k, 1)$ for treatment groups.

4. Non-DE genes: 170 non-DE genes in total. For each of them, generate a random number from 1 to 1000 and denote it as $\mu^{'}$. Simulate $N(\mu^{'}, 1)$ for all the lanes.

5. Compute p-values by using both methods and use 0.05 cutoff. And don't do p-value adjustment this time since we only have 200 genes.
Repeat this simulation 1000 times.

Simulation results:
In this simulation, we have a much smaller size of measured genes and also DE genes. Besides controlling false discoveries, we also want to see how many true positives can be identified. Instead of using proportions, we this time use counts and the reason is there are a lot of zeros for the number of rejected H_0 for t-test method.

K=2 Case:

For normalized data with t-test (Figure 2.13), there are two extreme clusters. The left cluster indicates small number of false positives, but the number of true positives in this cluster is also limited. The right cluster can pick up 20-30 true positives, but it also picks over 150 false positives. For linear model (Figure 2.14), the numbers of false positives are almost always under 20 and the numbers of true positives are always greater than 20.

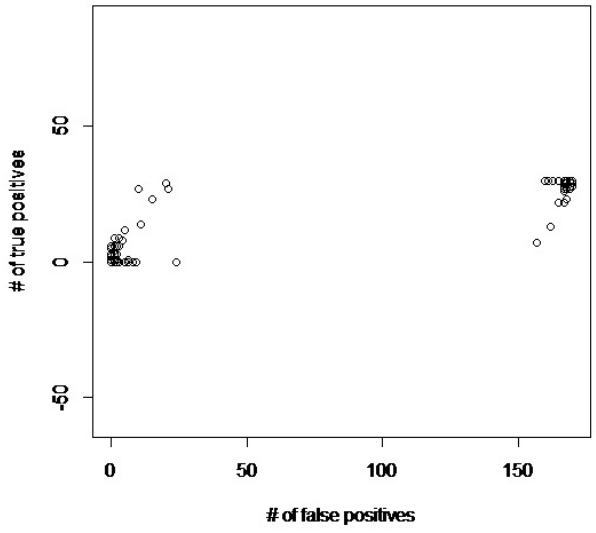

Figure 2.13: k=2, normalized data with t-test.

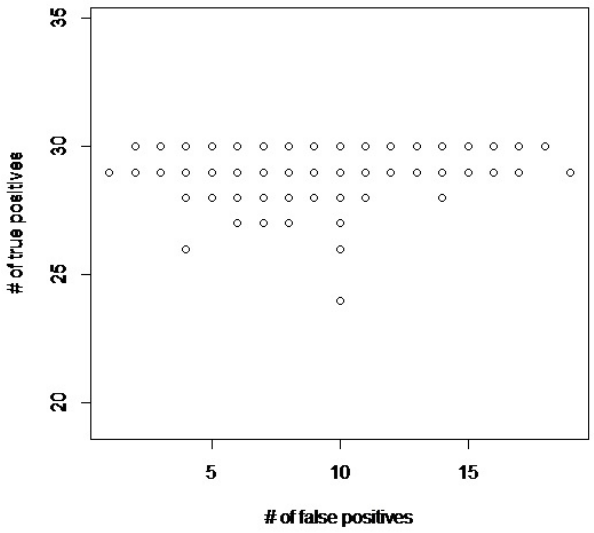

Figure 2.14: k=2, linear model.

K=1 Case:

For normalized data with t-test (Figure 2.15), two extreme clusters still exist. Comparing to the left cluster in Figure 2.13, this left cluster also shrinks to the point $(0, 0)$. And the right cluster indicates over 150 false positives with almost all the true positives are picked out. For linear model (Figure 2.16), we can see the number of false positives are always less than 20 and in most cases, more than 10 true positives could be picked up (Figure 2.20). Figure 2.18, the distribution of true positives picked up by t-test, shows in most cases, only less than 3 true positives can be identified. Therefore, linear model is still much better in this situation.

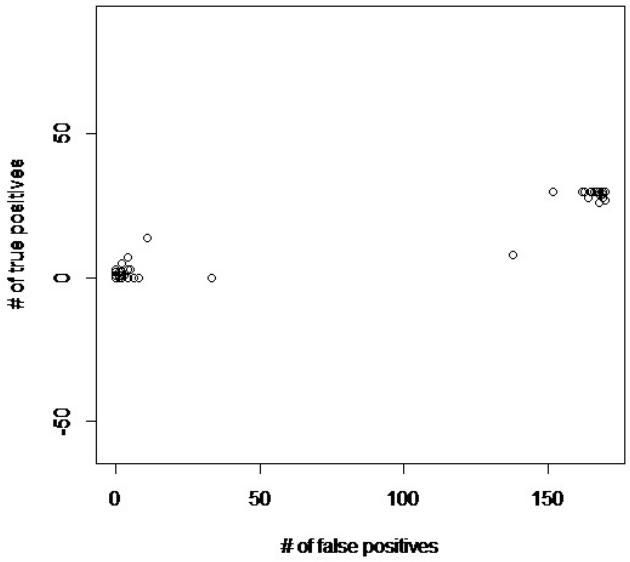

Figure 2.15: k=1, normalized data with t-test.

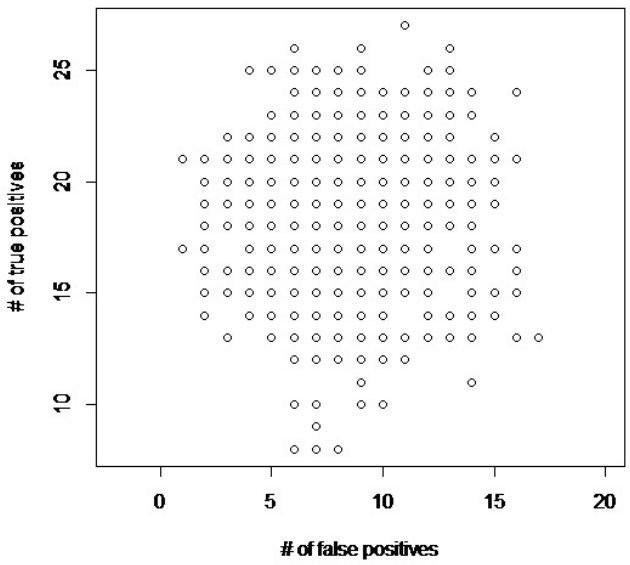

Figure 2.16: k=1, linear model.

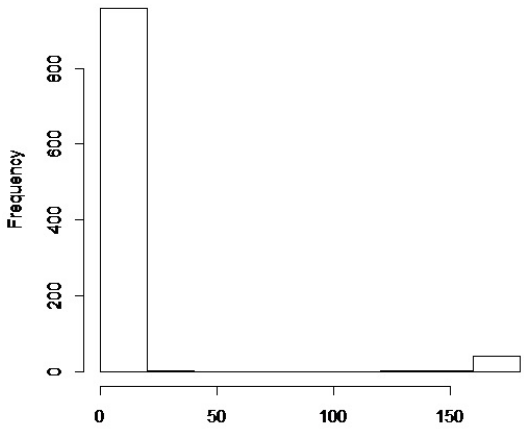

Figure 2.17: k=1, normalized data with t-test, number of false positives

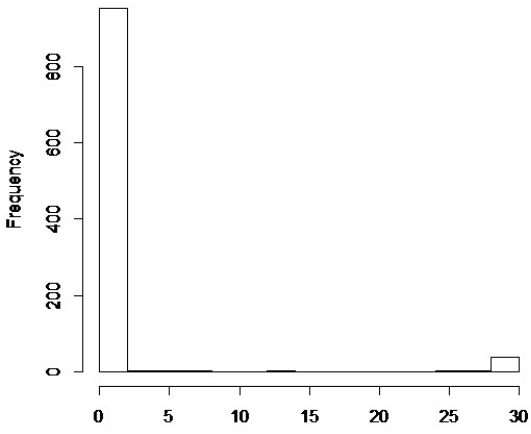

Figure 2.18: k=1, normalized data with t-test, number of true positives

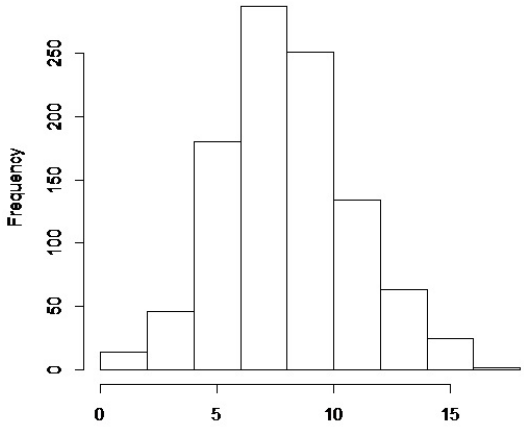

Figure 2.19: k=1, linear model, number of false positives

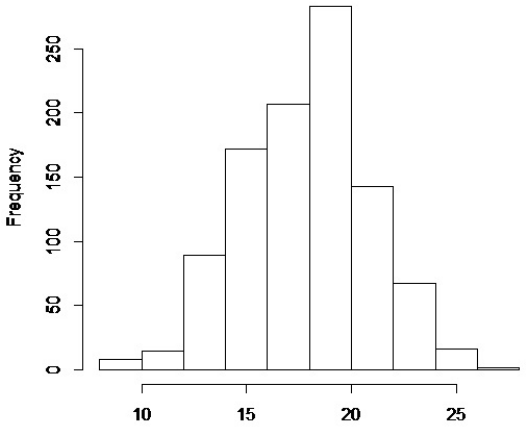

Figure 2.20: k=1, linear model, number of true positives

K=0.5 Case:

For normalized data with t-test (Figure 2.21), two extreme clusters are still there. Figure 2.23 and figure 2.24 provide the details of these two clusters. We can see the left cluster indicates less than 25 false positives with less than 5 true positives and the right cluster indicates over 150 false positives with almost all the true positive. For linear model (Figure 2.22, 2.25 and 2.26), with less than 20 false positives, the number of true positives are between 0 and 15. And in most cases, this number will be greater than 5.

Therefore,when k=0.5, linear model is still a better choice.

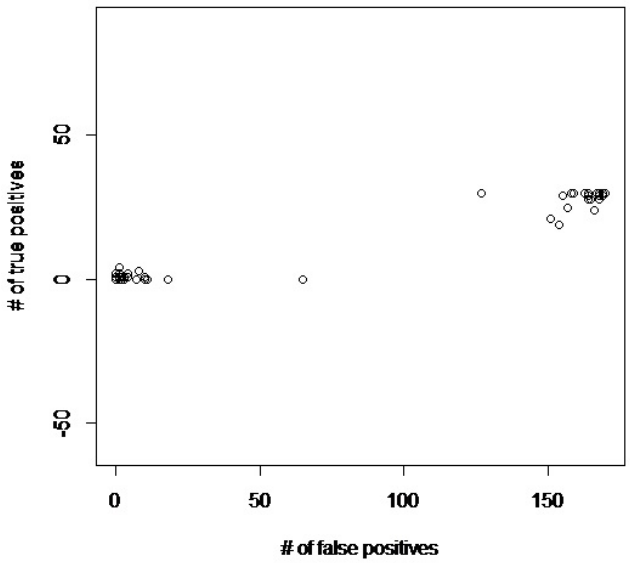

Figure 2.21: k=0.5, normalized data with t-test.

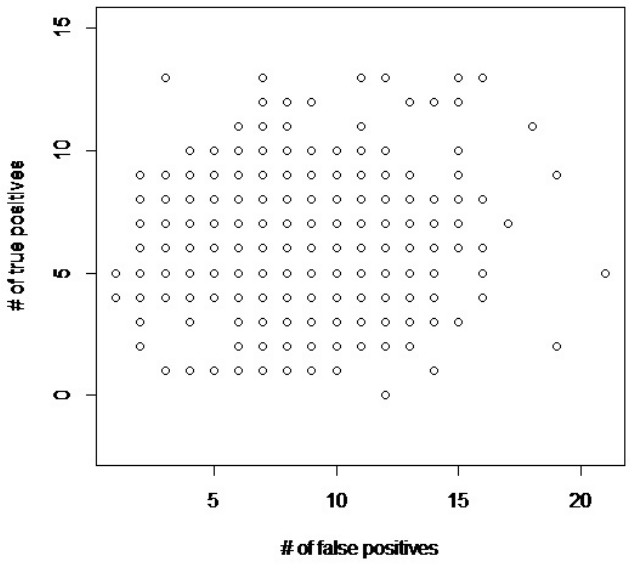

Figure 2.22: k=0.5, linear model.

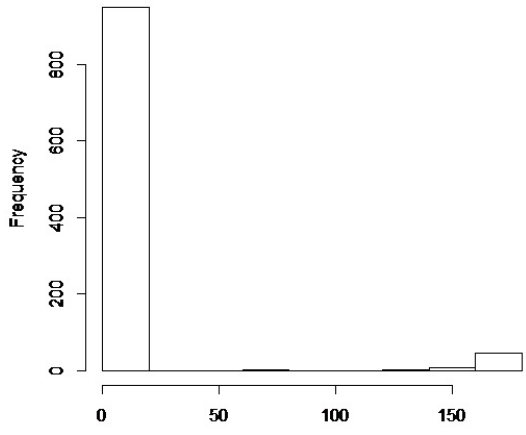

Figure 2.23: k=0.5, normalized data with t-test, number of false positives

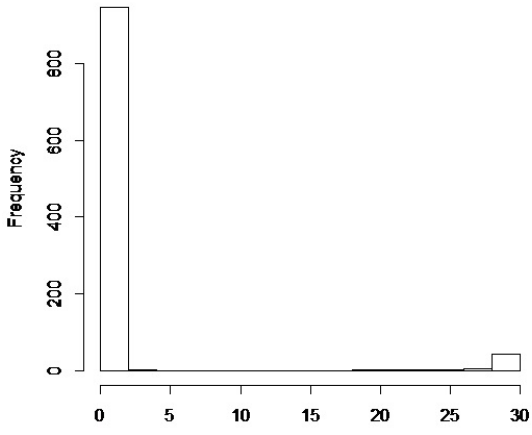

Figure 2.24: k=0.5, normalized data with t-test, number of true positives

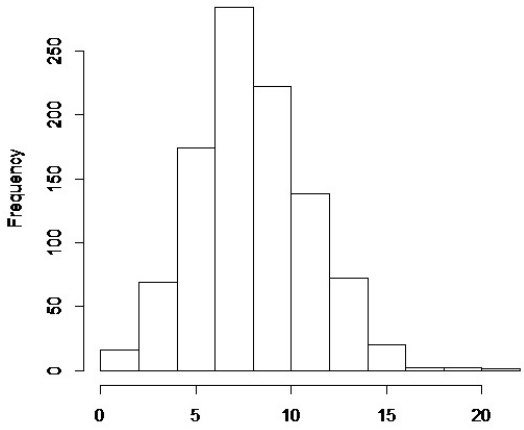

Figure 2.25: k=0.5, linear model, number of false positives

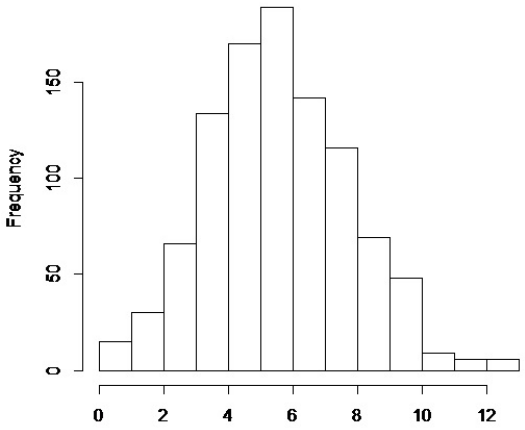

Figure 2.26: k=0.5, linear model, number of true positives

To handle the problem of reference genes mentioned before, we will use an ideal situation: for each of them, assume it has the same expression for all the lanes. Make the difference(k) between control and treatment to be 2 and 0.5, and compare two methods by taking a look at the numbers of false positives and true positives. According to table 2.1 and table 2.2, we could say both methods almost behave the same. This result shows traditional methods is highly affected by bad reference genes and very likely to fail in picking out DE genes. The linear model often provides much better result.

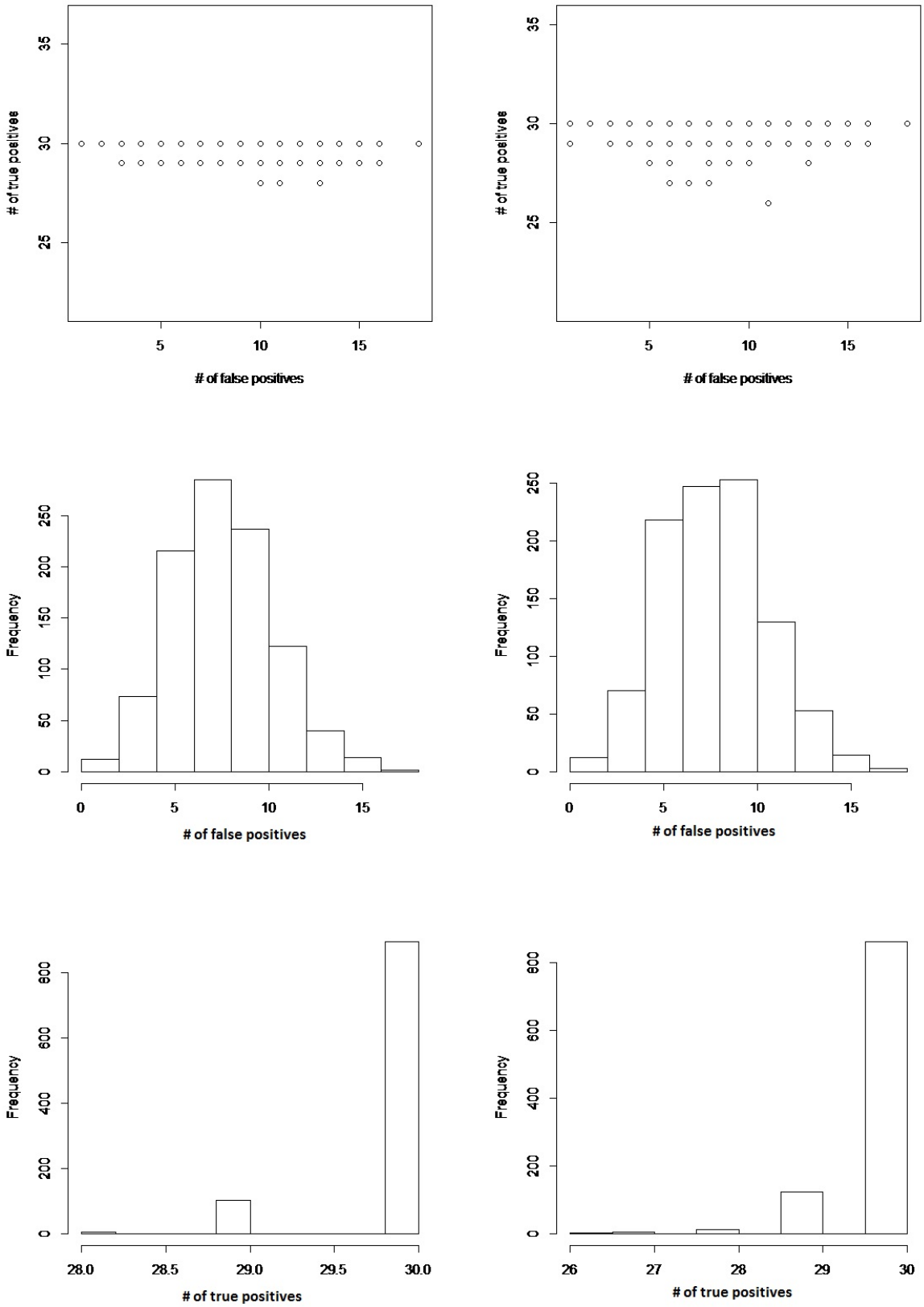

Table 2.1: k=2, left: normalized data with t-test; right: linear model

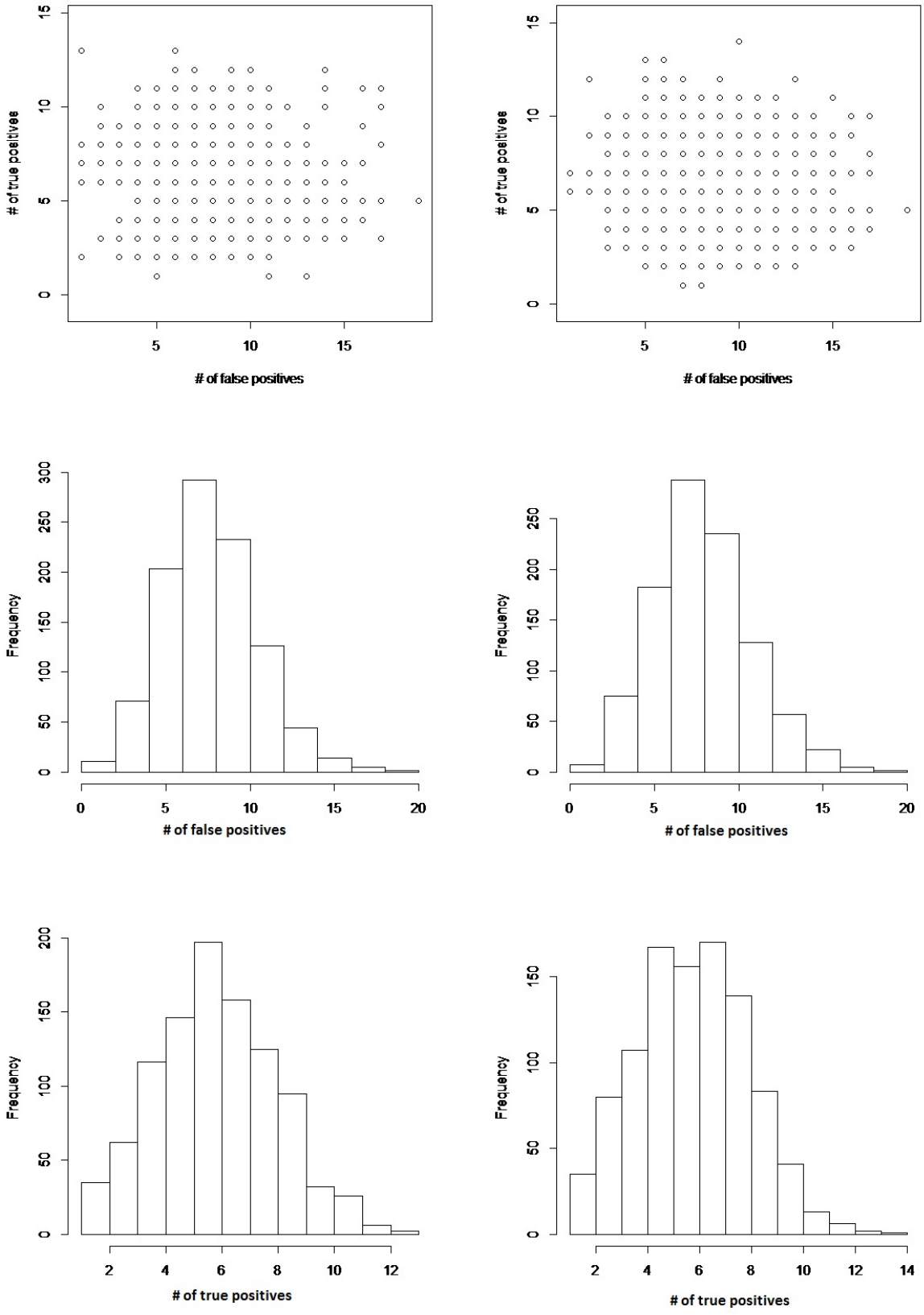

Table 2.2: k=0.5, left: normalized data with t-test; right: linear model

As the linear models all have the same coefficients pos and ref, we may have a dependency problem. In order to check if this problem exists, under the setting of simulation 2 with k=1, for a single simulated sample, we pull out the p-values for pos and ref and use a K-S test to see if they are approximately a uniform distribution on [0,1]. If a uniform distribution of the p-values is unreasonable, a mixed model may be needed for the dependency variables. After doing this simulation for 1000 times, the p-values for all the K-S tests are listed in table 2.3, specifically only 48 K-S tests for pos variable p-values are less than 0.05 and 50 K-S tests for ref variable p-values are less than 0.05.

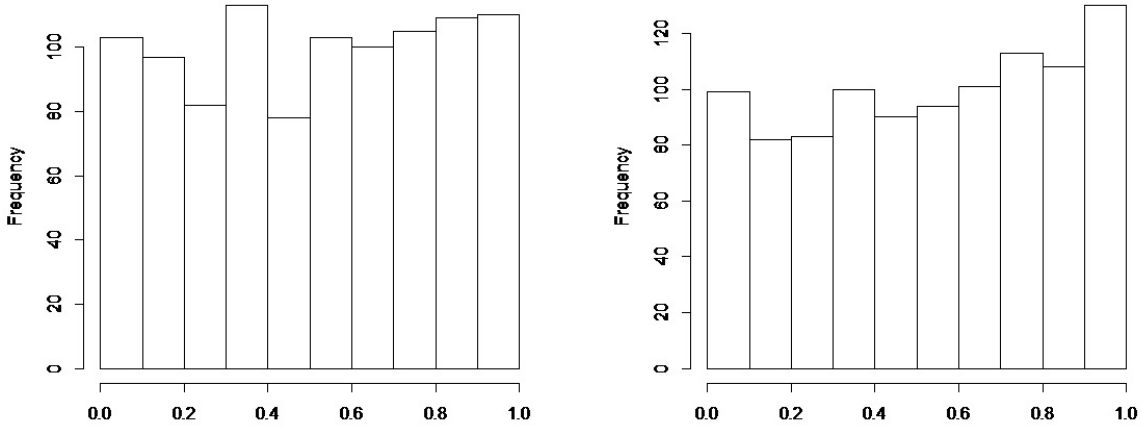

Table 2.3: left: K-S test results for pos; right: K-S test results for ref

2.1.3 Two Sample Example

In section 2.3.1 and section 2.3.2, we did two different kinds of simulations. And in this section, we will use a real data set to compare these two methods.

In this data, experiment was taken on 12 mice: half of them were healthy and the other half had a disease. 185 genes were measured and 6 of them were reference genes. So the traditional method will be normalized data with two sample t-test and our linear model will function as described before.

Normalized data with t-test picks out 28 significant genes(with 0.05 cutoff) and none of the reference genes are significant. P-value histogram(Figure 2.27) has a right-skewed shape which means t-test works properly.

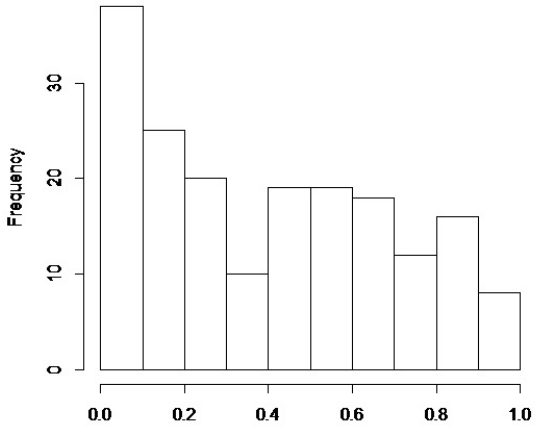

Figure 2.27: normalized data with t-test

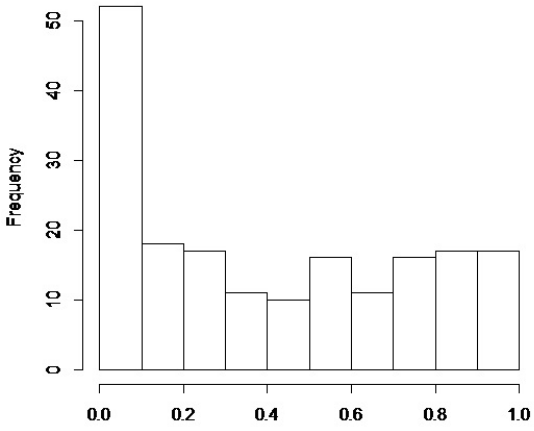

Figure 2.28: linear model

Linear model picks out 35 significant genes(with 0.05 cutoff) and two of them are reference genes which are Gapdh and Pgk1. P-value histogram(Figure 2.28) also has

a right-skewed shape with a better tail: it's more even and smoother, while the tail of Figure 2.27 has more bumps.

Comparing both significant gene lists, 19 genes are shared by both lists. And those two reference genes picked out by linear model suggest a question: are they truely significant genes? Because ideal reference genes have a property that they don't vary much across different treatments and from simulation 2, we do know that bad reference genes can affect results. Since the choice of reference genes is flexible and we can remove those having too much variation, therefor we will try to remove Gapdh and Pgk1(leaving 4 reference genes in the analysis) and see what changes will occur to the significant genes.

After the removal, normalized data with t-test picks out 38 significant genes(with 0.05 cutoff) and p-value histogram(Figure 2.29) still has a right-skewed shape, but a much smoother tail. Those 4 reference genes which are used to normalize data are not identified as significant. If we check the removed two reference genes: p-value of Pgk1 is 0.012(which is significant) and p-value of Gapdh is 0.165. And comparing this new list with previous one for t-test, they have 20 genes in common.

Linear model picks out 34 significant genes(with 0.05 cutoff) and p-value histogram(Figure 2.30) shows a right-skewed shape. None of the 4 used reference genes are picked out and Gapdh(p-value: 0.009) and Pgk1(p-value: 0.003) are still identified as significant. These 34 significant genes are all included in previous 35 genes which are picked out by linear model. And after removal, 27 genes are shared by both methods.

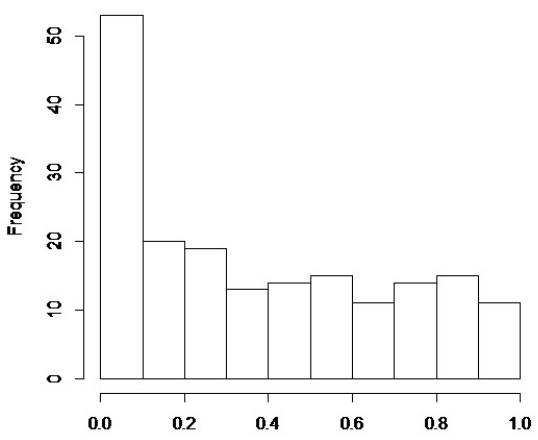

Figure 2.29: normalized data with t-test (remove 2 reference genes)

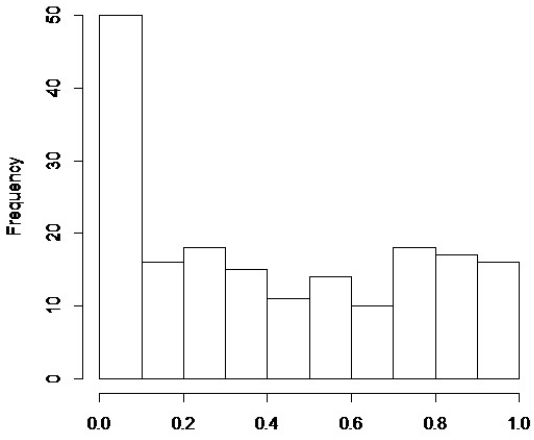

Figure 2.30: linear model (remove 2 reference genes)

Table 2.4: This table shows the numbers of significant genes before and after removal

	before removal	after removal	shared genes
T-test	28	38	20
Linear model	35	34	34
Shared genes	19	27	

Table 2.5: This table shows the significance of Gapdh and Pgk1

	before removal	after removal
T-test	Gapdh>0.05, Pgk1>0.05	Gapdh>0.05, Pgk1<0.05
Linear model	Gapdh<0.05, Pgk1<0.05	Gapdh<0.05, Pgk1<0.05

Conclusion:

In this example the linear model found that two of the reference genes were differentially expressed. This caused the normalization method to lose power for detecting genes that are differentially expressed. Without the linear model, the power of the study is compromised. With the linear model, more differentially expressed genes are identified include the two reference genes. If the two differentially expressed reference genes are removed, the linear model and normalization methods are similar, but the

problem with the reference genes can only be detected by the linear model method.

Chapter 3 Asymptotic Properties of a Normalized Test of Control versus Treatment

In this chapter, we derive the large sample distribution of the normalized test for differences between control and treated groups. Here we assume that the number of controls(m_1) and the number of treated(m_2) tend to ∞ at the same rate;i.e. $\frac{m_1}{m_1+m_2} \to \lambda$, $0 < \lambda < \infty$, as $m_1, m_2 \to \infty$.

Define

$$Y_{m_1} = \begin{pmatrix} \frac{1}{m_1}\sum_{j=1}^{m_1} p_{\cdot j}^{C} \\ \frac{1}{m_1}\sum_{j=1}^{m_1} \frac{r_j}{p_{\cdot j}} C \\ \frac{1}{m_1}\sum_{j=1}^{m_1} \frac{g_j}{r_j} C \end{pmatrix},$$

$$Y_{m_2} = \begin{pmatrix} \frac{1}{m_2}\sum_{j=1}^{m_2} p_{\cdot j}^{T} \\ \frac{1}{m_2}\sum_{j=1}^{m_2} \frac{r_j}{p_{\cdot j}} T \\ \frac{1}{m_2}\sum_{j=1}^{m_2} \frac{g_j}{r_j} T \end{pmatrix},$$

$$\mu^C = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_C \end{pmatrix},$$

and

$$\mu^T = \begin{pmatrix} \mu_4 \\ \mu_5 \\ \mu_T \end{pmatrix}.$$

From the multivariate CLT for i.i.d random vectors (Serfling(1980) Th 1.9.1B),

$$\sqrt{m_1}(Y_{m_1} - \mu^C) \xrightarrow{d} N(0, \Sigma^C), as\ m_1 \to \infty,$$

where

$$\Sigma^C = Var \begin{pmatrix} p_{.1}^C \\ \frac{r_1}{p_{.1}}{}^C \\ \frac{g_1}{r_1}{}^C \end{pmatrix} = \begin{pmatrix} Var(p_{.1}^C) & E(r_1^C) - \mu_1\mu_2 & E(p_{.1}^C \frac{g_1}{r_1}{}^C) - \mu_1\mu_C \\ E(r_1^C) - \mu_1\mu_2 & Var(\frac{r_1}{p_{.1}}{}^C) & E(\frac{g_1}{p_{.1}}{}^C) - \mu_2\mu_C \\ E(p_{.1}^C \frac{g_1}{r_1}{}^C) - \mu_1\mu_C & E(\frac{g_1}{p_{.1}}{}^C) - \mu_2\mu_C & Var(\frac{g_1}{r_1}{}^C) \end{pmatrix},$$

and

$$\sqrt{m_2}(Y_{m_2} - \mu^T) \xrightarrow{d} N(0, \Sigma^T), as\ m_2 \to \infty,$$

where

$$\Sigma^T = Var \begin{pmatrix} p_{.1}^T \\ \frac{r_1}{p_{.1}}{}^T \\ \frac{g_1}{r_1}{}^T \end{pmatrix} = \begin{pmatrix} Var(p_{.1}^T) & E(r_1^T) - \mu_4\mu_5 & E(p_{.1}^T \frac{g_1}{r_1}{}^T) - \mu_4\mu_T \\ E(r_1^T) - \mu_4\mu_5 & Var(\frac{r_1}{p_{.1}}{}^T) & E(\frac{g_1}{p_{.1}}{}^T) - \mu_5\mu_T \\ E(p_{.1}^T \frac{g_1}{r_1}{}^T) - \mu_4\mu_T & E(\frac{g_1}{p_{.1}}{}^T) - \mu_5\mu_T & Var(\frac{g_1}{r_1}{}^T) \end{pmatrix}.$$

Define

$$X_n = \sqrt{n} \left[\begin{pmatrix} Y_{m_1} \\ \\ Y_{m_2} \end{pmatrix} - \mu \right],$$

where

$$\mu = \begin{pmatrix} \mu^C \\ \\ \mu^T \end{pmatrix}.$$

Lemma 1. Suppose $\frac{m_1}{n} \to \lambda$, $0 < \lambda < 1$, where $n = m_1 + m_2$,then $X_n \xrightarrow{d} N(0, \Sigma)$, as $n \to \infty$, where

$$\Sigma = \begin{pmatrix} \frac{1}{\lambda}\Sigma^C & 0 \\ \\ 0 & \frac{1}{1-\lambda}\Sigma^T \end{pmatrix}.$$

Proof. First consider

$$\sqrt{n}(Y_{m_1} - \mu^C) = \frac{\sqrt{n}}{\sqrt{m_1}}\sqrt{m_1}(Y_{m_1} - \mu^C).$$

Note that

$$\frac{\sqrt{n}}{\sqrt{m_1}} \to \sqrt{\frac{1}{\lambda}}, as\ n \to \infty.$$

And by Serfling(1980) Th 1.9.1B,

$$\sqrt{m_1}(Y_{m_1} - \mu^C) \xrightarrow{d} N(0, \Sigma^C), as\ n \to \infty.$$

By Slutsky's Theorem, it follows that

$$\sqrt{n}(Y_{m_1} - \mu^C) \xrightarrow{d} N(0, \frac{1}{\lambda}\Sigma^C), as\ n \to \infty.$$

Similarly, we can get $\sqrt{n}(Y_{m_2} - \mu^T) \xrightarrow{d} N(0, \frac{1}{1-\lambda}\Sigma^T)$, as $n \to \infty$.

Since Y_{m_1} and Y_{m_2} are independent, then

$$X_n = \sqrt{n}\left(\begin{bmatrix} Y_{m_1} \\ \\ Y_{m_2} \end{bmatrix} - \mu\right) \xrightarrow{d} N\left(0, \begin{pmatrix} \frac{1}{\lambda}\Sigma^C & 0 \\ \\ 0 & \frac{1}{1-\lambda}\Sigma^T \end{pmatrix}\right), as\ n \to \infty.$$

Also define

$$c_n = \frac{m_1}{m_1+m_2}\mu_2 + \frac{m_2}{m_1+m_2}\mu_5 \ and \ c = \lambda\mu_2 + (1-\lambda)\mu_5,$$
$$d_n = \frac{m_1}{m_1+m_2}\mu_1 + \frac{m_2}{m_1+m_2}\mu_4 \ and \ d = \lambda\mu_1 + (1-\lambda)\mu_4.$$

Let M_{NC} be the mean of normalized control groups, that is,

$$M_{NC} = \frac{p_{..}}{n^2} \times \sum_{j=1}^{n} \frac{r_j}{p_{.j}} \times \frac{1}{m_1} \sum_{j=1}^{m_1} \frac{g_j}{r_j}^C.$$

Let M_{NT} be the mean of normalized treatment groups, that is,

$$M_{NT} = \frac{p_{..}}{n^2} \times \sum_{j=1}^{n} \frac{r_j}{p_{.j}} \times \frac{1}{m_2} \sum_{j=1}^{m_2} \frac{g_j}{r_j}^T.$$

Lemma 2. If $\frac{m_1}{n} \to \lambda$, $0 < \lambda < 1$, then

$$\begin{aligned} M_{NC} =& c\lambda\mu_C(\frac{1}{m_1}\sum p_{.j}^C - \mu_1) + c(1-\lambda)\mu_C(\frac{1}{m_2}\sum p_{.j}^T - \mu_4) \\ &+ d\lambda\mu_C(\frac{1}{m_1}\sum \frac{r_j}{p_{.j}}^C - \mu_2) + d(1-\lambda)\mu_C(\frac{1}{m_1}\sum \frac{r_j}{p_{.j}}^T - \mu_5) \\ &+ dc(\frac{1}{m_1}\sum \frac{g_j}{r_j}^C - \mu_C) + d_n c_n \mu_C + o_p(n^{-\frac{1}{2}}), as\ n \to \infty, \end{aligned}$$

and

$$\begin{aligned}M_{NT} =& c\lambda\mu_T(\frac{1}{m_1}\sum p_{.j}^{C} - \mu_1) + c(1-\lambda)\mu_T(\frac{1}{m_2}\sum p_{.j}^{T} - \mu_4)\\ &+ d\lambda\mu_T(\frac{1}{m_1}\sum \frac{r_j}{p_{.j}}^{C} - \mu_2) + d(1-\lambda)\mu_T(\frac{1}{m_1}\sum \frac{r_j}{p_{.j}}^{T} - \mu_5)\\ &+ dc(\frac{1}{m_1}\sum \frac{g_j}{r_j}^{T} - \mu_T) + d_n c_n \mu_T + o_p(n^{-\frac{1}{2}}), as\ n \to \infty.\end{aligned}$$

Proof. Write

$$X_n = \sqrt{n}\left(\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} - \mu\right)$$

Then, by Lemma 1, $X_n \xrightarrow{d} N(0, \Sigma)$. But

$$\begin{aligned}M_{NC} =& (\frac{m_1}{m_1+m_2}y_1 + \frac{m_2}{m_1+m_2}y_4)(\frac{m_1}{m_1+m_2}y_2 + \frac{m_2}{m_1+m_2}y_5)y_3\\ =& [\frac{m_1}{m_1+m_2}(y_1-\mu_1) + \frac{m_2}{m_1+m_2}(y_4-\mu_4) + \frac{m_1}{m_1+m_2}\mu_1 + \frac{m_2}{m_1+m_2}\mu_4]\\ &\times[\frac{m_1}{m_1+m_2}(y_2-\mu_2) + \frac{m_2}{m_1+m_2}(y_5-\mu_5) + \frac{m_1}{m_1+m_2}\mu_2 + \frac{m_2}{m_1+m_2}\mu_5]y_3\\ =& R_{n_1}R_{n_2}y_3 + R_{n_1}c_n y_3 + d_n R_{n_2} y_3 + d_n c_n y_3,\end{aligned}$$

where

$$R_{n_1} = \frac{m_1}{m_1+m_2}(y_1-\mu_1) + \frac{m_2}{m_1+m_2}(y_4-\mu_4)$$

and

$$R_{n_2} = \frac{m_1}{m_1+m_2}(y_2-\mu_2) + \frac{m_2}{m_1+m_2}(y_5-\mu_5).$$

Note that $R_{n_1}R_{n_2}y_3$ is $o_p(n^{-\frac{1}{2}})$, because

$$\begin{aligned}\sqrt{n}R_{n_1} =& \sqrt{n}(\frac{m_1}{m_1+m_2}(y_1-\mu_1) + \frac{m_2}{m_1+m_2}(y_4-\mu_4)) \\ =& \sqrt{n}\frac{m_1}{n}(y_1-\mu_1) + \sqrt{n}\frac{m_2}{n}(y_4-\mu_4) \\ =& \sqrt{n}\frac{\sqrt{m_1}}{n}\sqrt{m_1}(y_1-\mu_1) + \sqrt{n}\frac{\sqrt{m_2}}{n}\sqrt{m_2}(y_4-\mu_4)\end{aligned}$$

and

$$\sqrt{m_1}(y_1-\mu_1) \xrightarrow{d} N(0,\sigma_1^2), \sqrt{m_2}(y_4-\mu_4) \xrightarrow{d} N(0,\sigma_4^2),$$
$$\sqrt{n}\frac{\sqrt{m_1}}{n} \to \sqrt{\lambda} \; and \; \sqrt{n}\frac{\sqrt{m_2}}{n} \to \sqrt{1-\lambda}, as \; n \to \infty.$$

Since y_1 and y_4 are independent, $\sqrt{n}R_{n_1} \xrightarrow{d} N(0, \lambda\sigma_1^2 + (1-\lambda)\sigma_4^2)$,

and $R_{n_2} \xrightarrow{p} 0$, $y_3 \xrightarrow{p} \mu_C$. Therefore, $R_{n_1}R_{n_2}y_3 = o_p(n^{-\frac{1}{2}})$.

Also, $R_{n_1}c_ny_3 = R_{n_1}(c_n-c)y_3 + R_{n_1}cy_3$, $R_{n_1}(c_n-c)y_3$ is $o_p(n^{-\frac{1}{2}})$, because

$$\sqrt{n}R_{n_1} \xrightarrow{d} N(0, \lambda\sigma_1^2 + (1-\lambda)\sigma_4^2), \; c_n - c \to 0 \; and \; y_3 \xrightarrow{p} \mu_C.$$

$d_nR_{n_2}y_3 = (d_n-d)R_{n_2}y_3 + dR_{n_2}y_3$, $(d_n-d)R_{n_2}y_3 = o_p(n^{-\frac{1}{2}})$.

$d_nc_ny_3 = (d_nc_n-dc)(y_3-\mu_C) + dc(y_3-\mu_C) + d_nc_n\mu_C$,$(d_nc_n-dc)(y_3-\mu_C) = o_p(n^{-\frac{1}{2}})$,

because

$$d_nc_n - dc \to 0 \; and \; \sqrt{n}(y_3-\mu_C) \xrightarrow{d} N(0, \frac{1}{\lambda}\sigma_C^2).$$

Then

$$\begin{aligned}
M_{NC} =& R_{n_1} c y_3 + R_{n_2} d y_3 + dc(y_3 - \mu_C) + d_n c_n \mu_C + o_p(n^{-\frac{1}{2}}) \\
=& (R_{n_1} c + R_{n_2} d)(y_3 - \mu_C) + \mu_C (R_{n_1} c + R_{n_2} d) \\
& + dc(y_3 - \mu_C) + d_n c_n \mu_C + o_p(n^{-\frac{1}{2}}) \\
& (R_{n_1} c + R_{n_2} d)(y_3 - \mu_C) \ is \ o_p(n^{-\frac{1}{2}}) \\
=& \mu_C (R_{n_1} c + R_{n_2} d) + dc(y_3 - \mu_C) + d_n c_n \mu_C + o_p(n^{-\frac{1}{2}}),
\end{aligned}$$

where

$$\begin{aligned}
R_{n_1} =& \frac{m_1}{m_1 + m_2}(y_1 - \mu_1) + \frac{m_2}{m_1 + m_2}(y_4 - \mu_4) \\
=& (\frac{m_1}{m_1 + m_2} - \lambda)(y_1 - \mu_1) + (\frac{m_2}{m_1 + m_2} - (1 - \lambda))(y_4 - \mu_4) \\
& + \lambda(y_1 - \mu_1) + (1 - \lambda)(y_4 - \mu_4) \\
& (\frac{m_1}{m_1 + m_2} - \lambda)(y_1 - \mu_1) \ is \ o_p(n^{-\frac{1}{2}}) \\
& (\frac{m_2}{m_1 + m_2} - (1 - \lambda))(y_4 - \mu_4) \ is \ o_p(n^{-\frac{1}{2}}) \\
=& \lambda(y_1 - \mu_1) + (1 - \lambda)(y_4 - \mu_4) + o_p(n^{-\frac{1}{2}}).
\end{aligned}$$

Similarly,

$$R_{n_2} = \lambda(y_2 - \mu_2) + (1 - \lambda)(y_5 - \mu_5) + o_p(n^{-\frac{1}{2}})$$

and

$$\begin{aligned}
\mu_C (R_{n_1} c + R_{n_2} d) =& \mu_C [c\lambda(y_1 - \mu_1) + c(1 - \lambda)(y_4 - \mu_4) + c * o_p(n^{-\frac{1}{2}}) \\
& + d\lambda(y_2 - \mu_2) + d(1 - \lambda)(y_5 - \mu_5) + d * o_p(n^{-\frac{1}{2}})] \\
=& c\lambda \mu_C (y_1 - \mu_1) + c(1 - \lambda)\mu_C (y_4 - \mu_4) \\
& + d\lambda \mu_C (y_2 - \mu_2) + d(1 - \lambda)\mu_C (y_5 - \mu_5) + o_p(n^{-\frac{1}{2}}).
\end{aligned}$$

Then

$$\begin{aligned}
M_{NC} =& c\lambda\mu_C(y_1-\mu_1)+c(1-\lambda)\mu_C(y_4-\mu_4)+d\lambda\mu_C(y_2-\mu_2)\\
&+d(1-\lambda)\mu_C(y_5-\mu_5)+dc(y_3-\mu_C)+d_nc_n\mu_C+o_p(n^{-\frac{1}{2}})\\
=& c\lambda\mu_C(\frac{1}{m_1}\sum p_{\cdot j}^{C}-\mu_1)+c(1-\lambda)\mu_C(\frac{1}{m_2}\sum p_{\cdot j}^{T}-\mu_4)\\
&+d\lambda\mu_C(\frac{1}{m_1}\sum \frac{r_j}{p_{\cdot j}}^{C}-\mu_2)+d(1-\lambda)\mu_C(\frac{1}{m_1}\sum \frac{r_j}{p_{\cdot j}}^{T}-\mu_5)\\
&+dc(\frac{1}{m_1}\sum \frac{g_j}{r_j}^{C}-\mu_C)+d_nc_n\mu_C+o_p(n^{-\frac{1}{2}}).
\end{aligned}$$

And we can get M_{NT} in a similar way.

The next lemma uses Serfling(1980) Theorem A (page 122), which is given below

Theorem A: *Suppose that* $X_n=(X_{n1},...,X_{nk})$ *is* $AN(\mu,b_n^2\Sigma)$ *with* Σ *a covariance matrix and* $b_n\to 0$. *Let* $g(x)=(g_1(x),...g_m(x))$, $x=(x_1,...x_k)$, *be a vector valued function for which each component function* $g_i(x)$ *is real-valued and has a nonzero differential* $g_i(\mu;t)$, $t=(t_1,...t_k)$, *at* $x=\mu$. *Put*
$D=\left[\frac{\partial g_i}{\partial x_j}\Big|_{x=\mu}\right]_{m\times k}$.
Then
$g(X_n)$ *is* $AN(g(\mu),b_n^2D\Sigma D')$.

Let

$$Y_n = \begin{pmatrix} \frac{1}{m_1}\sum p_{.j}^{C} - \mu_1 \\ \\ \frac{1}{m_1}\sum \frac{r_j}{p_{.j}}{}^{C} - \mu_2 \\ \\ \frac{1}{m_1}\sum \frac{g_j}{r_j}{}^{C} - \mu_C \\ \\ \frac{1}{m_2}\sum p_{.j}^{T} - \mu_4 \\ \\ \frac{1}{m_2}\sum \frac{r_j}{p_{.j}}{}^{T} - \mu_5 \\ \\ \frac{1}{m_2}\sum \frac{g_j}{r_j}{}^{T} - \mu_T \end{pmatrix} = \begin{pmatrix} y_1 \\ \\ y_2 \\ \\ y_3 \\ \\ y_4 \\ \\ y_5 \\ \\ y_6 \end{pmatrix},$$

$$\begin{aligned} g_1(Y) =& g_1(y_1, ..., y_6) \\ =& c\lambda\mu_C y_1 + c(1-\lambda)\mu_C y_4 + d\lambda\mu_C y_2 + d(1-\lambda)\mu_C y_5 + dcy_3, \end{aligned}$$

and

$$\begin{aligned} g_2(Y) =& g_2(y_1, ..., y_6) \\ =& c\lambda\mu_T y_1 + c(1-\lambda)\mu_T y_4 + d\lambda\mu_T y_2 + d(1-\lambda)\mu_T y_5 + dcy_6. \end{aligned}$$

Lemma 3. If $\frac{m_1}{n} \to \lambda$, $0 < \lambda < 1$, as $n \to \infty$, then

$$\sqrt{n}\begin{pmatrix} g_1(Y_n) \\ \\ g_2(Y_n) \end{pmatrix} \xrightarrow{d} N(0, D\Sigma D^{'}), as\ n \to \infty, where$$

$$D = \left(\left.\frac{\partial g_i}{\partial x_j}\right|_{x=\mu}\right) = \begin{pmatrix} c\lambda\mu_C & d\lambda\mu_C & \text{dc} & c(1-\lambda)\mu_C & d(1-\lambda)\mu_C & 0 \\ \\ c\lambda\mu_T & d\lambda\mu_T & 0 & c(1-\lambda)\mu_T & d(1-\lambda)\mu_T & \text{dc} \end{pmatrix}.$$

Proof. From Lemma 1, we have

$$\sqrt{n}Y_n = \sqrt{n}\begin{pmatrix} y_1 \\ \\ y_2 \\ \\ y_3 \\ \\ y_4 \\ \\ y_5 \\ \\ y_6 \end{pmatrix} \xrightarrow{d} N(0,\Sigma),$$

which is Y_n is $AN(0, n^{-1}\Sigma)$.

Let

$$g(Y) = \begin{pmatrix} g_1(Y) \\ \\ g_2(Y) \end{pmatrix}.$$

According to Theorem A, $g(Y_n)$ is $AN(0, n^{-1}D\Sigma D^{'})$, where

$$D = \left(\left.\frac{\partial g_i}{\partial x_j}\right|_{x=\mu}\right) = \begin{pmatrix} c\lambda\mu_C & d\lambda\mu_C & \text{dc} & c(1-\lambda)\mu_C & d(1-\lambda)\mu_C & 0 \\ \\ c\lambda\mu_T & d\lambda\mu_T & 0 & c(1-\lambda)\mu_T & d(1-\lambda)\mu_T & \text{dc} \end{pmatrix}.$$

That is

$$\sqrt{n}\begin{pmatrix} g_1(Y_n) \\ \\ g_2(Y_n) \end{pmatrix} \xrightarrow{d} N(0, D\Sigma D^{'}), as\ n \to \infty.$$

Lemma 4. As $n \to \infty$,

$$M_{NC} - M_{NT} - (d_n c_n(\mu_C - \mu_T))\ is\ AN(0, \frac{1}{n} vD\Sigma D^{'}v^{'}),\ where$$

$v = (1, -1)$ and

$$D = \begin{pmatrix} c\lambda\mu_C & d\lambda\mu_C & \text{dc} & c(1-\lambda)\mu_C & d(1-\lambda)\mu_C & 0 \\ \\ c\lambda\mu_T & d\lambda\mu_T & 0 & c(1-\lambda)\mu_T & d(1-\lambda)\mu_T & \text{dc} \end{pmatrix}.$$

Proof. Let $G(Z) = z_1 - z_2$,

$$Z = \begin{pmatrix} z_1 \\ \\ z_2 \end{pmatrix}.$$

Since

$$g(Y_n) = \begin{pmatrix} g_1(Y_n) \\ \\ g_2(Y_n) \end{pmatrix}\ isAN(0, n^{-1}D\Sigma D^{'}),$$

then according to Theorem A

$$G\left[\begin{pmatrix} g_1(Y_n) \\ \\ g_2(Y_n) \end{pmatrix}\right] = g_1(Y_n) - g_2(Y_n)\ is\ AN(0, n^{-1}vD\Sigma D^{'}v^{'}),$$

which is $\sqrt{n}(g_1(Y_n) - g_2(Y_n)) \xrightarrow{d} N(0, vD\Sigma D'v')$, where $v = (1, -1)$ and D takes the value from Lemma 3.

Note that from Lemma 2

$$\sqrt{n}(M_{NC} - M_{NT}) = \sqrt{n}(g_1(Y_n) - g_2(Y_n) + d_n c_n(\mu_C - \mu_T) + o_p(n^{-\frac{1}{2}})),$$

and we have $\sqrt{n} * o_p(n^{-\frac{1}{2}}) \xrightarrow{p} 0$.

This implies

$$M_{NC} - M_{NT} - (d_n c_n(\mu_C - \mu_T)) \xrightarrow{d} N(0, \frac{1}{n} vD\Sigma D'v').$$

Chapter 4 Conclusions and Future Work

This dissertation is devoted to the analysis of NanoString nCounter data via a linear regression model. Specifically, we introduce this new type of gene expression data at the very beginning and then present two types of normalization methods. After this, we carefully examine the entire normalization process and find out its potential problems which could affect the statistical analysis results. And then we propose our linear regression method which is stable and effective.

This linear model is developed to discover differentially expressed genes. In order to compare it with traditional method, we simulate two kinds of data and use the proportion of false discoveries as the rule to check which method is better. And in both conditions, the linear model always has more stable and trustable results. Besides simulated data, we also use a real data set to compare the two methods, the genes lists identified by picked out by the traditional method are highly influenced by reference genes while linear model is much less sensitive to them.

Our theoretical research is about the asymptotic properties of the normalized test of control versus Treatment. The results reveal us why the traditional normalization method fails in some cases. Future work could consider asymptotic properties for more complex experimental designs.

Appendices

A. R code for Simulation 1

```
### n1: number of control groups
### n2: number of treatment  groups
### mu: mean of DE treatment
### 250 DE genes and 750 non-DE genes
### 6 positive control, 6 reference genes



simulation1<-function(n1,n2,mu ){
pos<-matrix((rnorm((n1+n2)*6, mean = 100, sd = 1)),nrow=6)
rownames(pos) <- paste("pos",1:6)
ref<-matrix((rnorm((n1+n2)*6, mean = 100, sd = 1)),nrow=6)
rownames(ref) <- paste("ref",1:6)
dectrl<-matrix((rnorm(n1*250, mean = 100, sd = 1)),nrow=250)
detrt<-matrix((rnorm(n2*250, mean = mu, sd = 1)),nrow=250)
de<-cbind(dectrl,detrt)
nonde<-matrix((rnorm((n1+n2)*750, mean = 100, sd = 1)),nrow=750)
gene<-rbind(de,nonde)
rownames(gene) <- paste("gene",1:1000)
design<-c(rep(0, n1), rep(1, n2))
return(list(pos=pos, ref=ref, gene=gene, design=design))
}
##############################################################################
```

```
res<-matrix(0, 1000, 4) ###simulate this data frame 1000 times
for (j in 1:1000){
s<-simulation1(24, 24, 100.5)
pos<-s$pos
ref<-s$ref
gene<-s$gene
design<-s$design
possum<-apply(pos, 2,sum)
geomean = function(x){prod(x)^(1/length(x))}
refgeomean<- apply(ref,2,geomean)

scalar<-mean(possum)* mean(refgeomean/possum)/ refgeomean
scalar<-matrix(scalar, nrow=1)
scalarmatrix= matrix(rep(scalar,1000),nrow=1000,byrow=TRUE)
normalizedgene<-gene* scalarmatrix



mpvalue<-rep(0,1000); ###compute p-values by using linear model
for (i in 1:1000){
  data<-data.frame(gene=gene[i,], trt=design,pos=possum,ref=refgeomean)
  fit<- lm(gene~trt+pos+ref,data=data)
  mpvalue[i]<-summary(fit)$coefficients[2,4];
 }
mR=length(which(mpvalue<0.05)) ### number of rejected H0
mV=length(which(mpvalue[251:1000]<0.05)) ### number of false positives
```

```
npvalue<-rep(0,1000); ###compute p-values by using normalized data
for (i in 1:1000){
  data<-data.frame(ctrl= normalizedgene [i,1:24], trt= normalizedgene [i,25:48])
  npvalue[i]<- t.test(data$ctrl, data$trt)$p.value
 }
nR=length(which(npvalue<0.05)) ### number of rejected H0
nV=length(which(npvalue[251:1000]<0.05)) ### number of false positives



res[j,]=c(mR, mV, nR, nV)
}

mSFDR=res[,2]/res[,1]
nSFDR=res[,4]/res[,3]
plot(mSFDR, nSFDR, asp=1)
abline(0,1,col="red")

boxplot(res[,1], res[,3], names=c("number of rejected H0 of M",
"number of rejected H0 of N"))

###compute "bonferroni" and "BH" adjusted p-value for both methods
bonfmpvalue=p.adjust(mpvalue, "bonferroni")
bonfmR=length(which(bonfmpvalue<0.05)) ### number of rejected H0
bonfmV=length(which(bonfmpvalue[251:1000]<0.05)) ### number of false positives

bhmpvalue=p.adjust(mpvalue, "BH")
bhmR=length(which(bhmpvalue<0.05)) ### number of rejected H0
```

```
bhmV=length(which(bhmpvalue[251:1000]<0.05)) ### number of false positives


bonfnpvalue=p.adjust(npvalue, "bonferroni")
bonfnR=length(which(bonfnpvalue<0.05)) ### number of rejected H0
bonfnV=length(which(bonfnpvalue[251:1000]<0.05)) ### number of false positives


bhnpvalue=p.adjust(npvalue, "BH")
bhnR=length(which(bhnpvalue<0.05)) ### number of rejected H0
bhnV=length(which(bhnpvalue[251:1000]<0.05)) ### number of false positives
```

B. R code for Simulation 2

```
### n1: number of control groups
### n2: number of treatment groups
### 30 DE genes, 170 non-DE genes
### k: the shift between control and treatment for DE genes

simulation2<-function(n1,n2, k){
###generate positive controls, use their concentration property
###and make an increasing pattern
x6<-sample(1:10, n1+n2,replace=T)
x5<-x6 +sample(1:5, n1+n2,replace=T)
x4<-4*x5+ sample(0:10, n1+n2,replace=T)
x3<-4*x4+ sample(0:10, n1+n2,replace=T)
x2<-4*x3+ sample(0:100, n1+n2,replace=T)
x1<-4*x2+ sample(0:100, n1+n2,replace=T)
pos<-rbind(x1, x2, x3, x4, x5, x6)
rownames(pos) <- paste("pos",1:6)

###reference genes are always expressed at different levels, set
###them as: 1-99, 100-999, 1000-9999, 10000-99999
refcar<-rep(0,6);
for (i in 1:6){
x<-sample(1:4, 1,replace=T)
refcar[i]<-x
 }
```

```
ref<-matrix(0, 6, n1+n2)
for (j in 1:6){
if( refcar[j]==1)
{ ref[j, ]= matrix(sample(1:99, n1+n2,replace=T), nrow=1)}
else if (refcar[j]==2)
{ ref[j, ]= matrix(sample(100:999, n1+n2,replace=T), nrow=1)}
else if (refcar[j]==3)
{ ref[j, ]= matrix(sample(1000:9999,n1+n2,replace=T), nrow=1)}
else {ref[j, ]= matrix(sample(10000:99999,n1+n2,replace=T), nrow=1)}
}
rownames(ref) <- paste("ref",1:6)

###generate DE genes
de<-matrix(0, 30, n1+n2)
for (t in 1:30){
mu1<-sample(1:1000, 1,replace=T)
de[t, 1:n1]<-rnorm(n1, mu1, sd=1)
de[t, n1+1: n2]<-rnorm(n2, mu1+k, sd=1)
 }

###generate non-DE genes
nonde<-matrix(0, 170, n1+n2)
for (h in 1:170){
mu2<-sample(1:1000, 1,replace=T)
nonde[h, ]<-rnorm(n1+n2, mu2, sd=1)
}
```

```
gene<-rbind(de,nonde)
rownames(gene) <- paste("gene",1:200)
design<-c(rep(0, n1), rep(1, n2))
return(list(pos=pos, ref=ref, gene=gene, design=design))
}

res<-matrix(0, 1000, 4)
for (j in 1:1000){
s<-simulation2(12, 12, 2)
pos<-s$pos
ref<-s$ref
gene<-s$gene
design<-s$design
possum<-apply(pos, 2,sum)
geomean = function(x){prod(x)^(1/length(x))}
refgeomean<- apply(ref,2,geomean)

scalar<-mean(possum)* mean(refgeomean/possum)/ refgeomean
scalar<-matrix(scalar, nrow=1)
scalarmatrix= matrix(rep(scalar,200),nrow=200,byrow=TRUE)
normalizedgene<-gene* scalarmatrix

mpvalue<-rep(0,200);
for (i in 1:200){
  data<-data.frame(gene=gene[i,], trt=design,pos=possum,ref=refgeomean)
  fit<- lm(gene~trt+pos+ref,data=data)
  mpvalue[i]<-summary(fit)$coefficients[2,4];
```

```
 }
mR=length(which(mpvalue<0.05)) ### number of rejected H0
mV=length(which(mpvalue[31:200]<0.05)) ### number of false positives

npvalue<-rep(0,200);
for (i in 1:200){
  data<-data.frame(ctrl= normalizedgene [i,1:12], trt= normalizedgene [i,13:24])
  npvalue[i]<- t.test(data$ctrl, data$trt)$p.value
 }
nR=length(which(npvalue<0.05)) ### number of rejected H0
nV=length(which(npvalue[31:200]<0.05)) ### number of false positives

res[j,]=c(mR, mV, nR, nV)
}

mR<-res[,1]
mV<-res[,2]
nR<-res[,3]
nV<-res[,4]
plot(nV, nR-nV, xlab ="# of false positives", ylab ="# of true positives",asp=1)
plot(mV, mR-mV, xlab ="# of false positives", ylab ="# of true positives",asp=1)
```

Bibliography

[1] Vladislav A Malkov, Kyle A Serikawa, Noel Balantac, James Watters, Gary Geiss, Afshin Mashadi-Hossein, and Thomas Fare. Multiplexed measurements of gene signatures in different analytes using the nanostring ncounter assay system. *BMC research notes*, 2(1):1, 2009.

[2] Margaret H Veldman-Jones, Zhongwu Lai, Mark Wappett, Chris G Harbron, J Carl Barrett, Elizabeth A Harrington, and Kenneth S Thress. Reproducible, quantitative, and flexible molecular subtyping of clinical dlbcl samples using the nanostring ncounter system. *Clinical Cancer Research*, 21(10):2367–2378, 2015.

[3] Margaret H Veldman-Jones, Roz Brant, Claire Rooney, Catherine Geh, Hollie Emery, Chris G Harbron, Mark Wappett, Alan Sharpe, Michael Dymond, J Carl Barrett, et al. Evaluating robustness and sensitivity of the nanostring technologies ncounter platform to enable multiplexed gene expression analysis of clinical samples. *Cancer research*, 75(13):2587–2593, 2015.

[4] NanoString Technologies. *nCounter Brochure*, 2015.

[5] Jo Vandesompele, Katleen De Preter, Filip Pattyn, Bruce Poppe, Nadine Van Roy, Anne De Paepe, and Frank Speleman. Accurate normalization of real-time quantitative rt-pcr data by geometric averaging of multiple internal control genes. *Genome biology*, 3(7):1, 2002.

[6] James H Bullard, Elizabeth Purdom, Kasper D Hansen, and Sandrine Dudoit. Evaluation of statistical methods for normalization and differential expression in mrna-seq experiments. *BMC bioinformatics*, 11(1):1, 2010.

[7] Lana Xia Garmire and Shankar Subramaniam. Evaluation of normalization methods in mammalian microrna-seq data. *RNA*, 18(6):1279–1288, 2012.

[8] NanoString Technologies. *nCounter Expression Data Analysis Guide*, 2012.

[9] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300, 1995.

[10] Yoav Benjamini and Yosef Hochberg. On the adaptive control of the false discovery rate in multiple testing with independent statistics. *Journal of educational and Behavioral Statistics*, 25(1):60–83, 2000.

[11] RL Fernando, D Nettleton, BR Southey, JCM Dekkers, MF Rothschild, and M Soller. Controlling the proportion of false positives in multiple dependent tests. *Genetics*, 166(1):611–619, 2004.

[12] Yoav Benjamini, Abba M Krieger, and Daniel Yekutieli. Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*, 93(3):491–507, 2006.

[13] Daniela Witten and Robert Tibshirani. A comparison of fold-change and the t-statistic for microarray data analysis. *Analysis*, 17, 2007.

[14] Mark R Dalman, Anthony Deeter, Gayathri Nimishakavi, and Zhong-Hui Duan. Fold change and p-value cutoffs significantly alter microarray interpretations. *BMC bioinformatics*, 13(2):1, 2012.

[15] Melissa J Peart, Gordon K Smyth, Ryan K van Laar, David D Bowtell, Victoria M Richon, Paul A Marks, Andrew J Holloway, and Ricky W Johnstone. Identification and functional significance of genes regulated by structurally different histone deacetylase inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, 102(10):3697–3702, 2005.

[16] Tucker A Patterson, Edward K Lobenhofer, Stephanie B Fulmer-Smentek, Patrick J Collins, Tzu-Ming Chu, Wenjun Bao, Hong Fang, Ernest S Kawasaki, Janet Hager, Irina R Tikhonova, et al. Performance comparison of one-color and two-color platforms within the microarray quality control (maqc) project. *Nature biotechnology*, 24(9):1140–1150, 2006.

[17] CE Huggins, AA Domenighetti, ME Ritchie, N Khalil, JM Favaloro, Joseph Proietto, GK Smyth, Salvatore Pepe, and LMD Delbridge. Functional and metabolic remodelling in glut4-deficient hearts confers hyper-responsiveness to substrate intervention. *Journal of molecular and cellular cardiology*, 44(2):270–280, 2008.

[18] Davis J McCarthy and Gordon K Smyth. Testing significance relative to a fold-change threshold is a treat. *Bioinformatics*, 25(6):765–771, 2009.

[19] John D Storey and Robert Tibshirani. Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences*, 100(16):9440–9445, 2003.

[20] Dan Nettleton, JT Gene Hwang, Rico A Caldo, and Roger P Wise. Estimating the number of true null hypotheses from a histogram of p values. *Journal of Agricultural, Biological, and Environmental Statistics*, 11(3):337–356, 2006.

[21] Stanley B Pounds. Estimation and control of multiple testing error rates for microarray studies. *Briefings in bioinformatics*, 7(1):25–36, 2006.

[22] Evelien Vaes, Mona Khan, and Peter Mombaerts. Statistical analysis of differential gene expression relative to a fold change threshold on nanostring data of mouse odorant receptor genes. *BMC bioinformatics*, 15(1):1, 2014.

[23] Hee Jin Lee, Jeong-Ju Lee, In Hye Song, In Ah Park, Jun Kang, Jong Han Yu, Jin-Hee Ahn, and Gyungyub Gong. Prognostic and predictive value of nanostring-based immune-related gene signatures in a neoadjuvant setting of

triple-negative breast cancer: relationship to tumor-infiltrating lymphocytes. *Breast cancer research and treatment*, 151(3):619–627, 2015.

[24] Christopher D Brumbaugh, Hyunsung J Kim, Mario Giovacchini, and Nader Pourmand. Nanostride: normalization and differential expression analysis of nanostring ncounter data. *BMC bioinformatics*, 12(1):1, 2011.

[25] Daryl Waggott, Kenneth Chu, Shaoming Yin, Bradly G Wouters, Fei-Fei Liu, and Paul C Boutros. Nanostringnorm: an extensible r package for the pre-processing of nanostring mrna and mirna data. *Bioinformatics*, 28(11):1546–1548, 2012.

[26] Gordon K Smyth. Limma: linear models for microarray data. In *Bioinformatics and computational biology solutions using R and Bioconductor*, pages 397–420. Springer, 2005.

[27] James M Wettenhall and Gordon K Smyth. limmagui: a graphical user interface for linear modeling of microarray data. *Bioinformatics*, 20(18):3705–3706, 2004.

[28] James M Wettenhall, Ken M Simpson, Keith Satterley, and Gordon K Smyth. affylmgui: a graphical user interface for linear modeling of single channel microarray data. *Bioinformatics*, 22(7):897–899, 2006.

[29] Wei Pan, Jizhen Lin, and Chap T Le. A mixture model approach to detecting differentially expressed genes with microarray data. *Functional & integrative genomics*, 3(3):117–124, 2003.

[30] Robert J Serfling. *Approximation theorems of mathematical statistics*, volume 162. John Wiley & Sons, 2009.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30]

Vita

SHU SHEN

Education

Master in Statistics, University of Kentucky, 2010

Bachelor in Statistics, University of Science and Technology of China, 2008

Research Experience

Research Assistant *2011-2016*

Department of Statistics, University of Kentucky

Teaching Assistant *2008-2011*

Department of Statistics, University of Kentucky