Randomized Quantile Residual for Assessing Generalized Linear Mixed Models with Application to Zero-Inflated Microbiome Data

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon

By
Wei Bai

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Mathematics and Statistics
142 Mcclean Hall, 106 Wiggins Road
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5E6
Canada
OR
Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

Abstract

In microbiome research, it is often of interest to investigate the impact of clinical and environmental factors on microbial abundance, which is often quantified as the total number of unique operational taxonomic units (OTUs). The important features of OTU count data are the presence of a large number of zeros and skewness in the positive counts. A common strategy to handle excessive zeros is to use zero-inflated models or zero-modified (hurdle) models. Moreover, subjects in microbiome data often have clustering structure, for example humans from the same family or plants from the same plot; as a result, random effects should be included to account for the clustering effects.

Model diagnosis is an essential step to ensure that a fitted model is adequate for the data. However, diagnosing zero-inflated counts models is still a challenging research problem. Pearson and deviance residuals are often used in practice for diagnosing counts models, despite wide recognition that these residuals are far from normality when applied to count data. Randomized quantile residual (RQR) was proposed in literature to circumvent the above problems in traditional residuals. The key idea of the RQR is to randomize the lower tail probability into a uniform random number between the discontinuity gap of cumulative density function (CDF). It can be shown that RQRs are normally distributed under the true model. To the best of our knowledge, RQR has not been applied to diagnose zero inflated or modified mixed effects models. In this thesis project, we have developed generic R functions that can compute RQRs for zero-inflated and zero-modified mixed effects models based on fitting outputs of `glmmTMB`. We have tested our functions using datasets generated from zero-modified Poisson (ZMP) and zero-modified negative binomial (ZMNB) models. Our simulation studies show that RQRs are normally distributed under the true model. In GOF tests, the type 1 error rates are close to the nominal level 0.05, and the powers of rejecting the wrong models are very good. We have also applied RQR to assess 8 models for a real human microbiome OTU dataset and concluded that ZMNB or zero-inflated negative binomial (ZINB) models provide adequate fits to the dataset.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to my supervisors, Dr. Longhai Li and Dr. Cindy Feng, for their continuing support and help. Their guidance and encouragement steer me in the right direction and give me strength to pursue advanced knowledge. Their preciseness and diligence in science lead me to be hardworking as well. It is a great honor to work with my supervisors. This thesis could not have been done without their advice and guidance.

I sincerely thank Dr. Shahedul Khan and Dr. Laura Wright on my advisory committee for their encouragement and insightful comments. I would also like to thank the Professor Wei Xu from the University of Toronto for providing the microbiome data for this research. I would like to thank the Department of Mathematics and Statistics at the University of Saskatchewan for academic and financial support to my MSc study. I am grateful to all of the professors, graduate students, and staff in the Department of Mathematics and Statistics.

I would especially like to express very profound gratitude to my parents for their deep love and strong support during my whole life. I love them forever.

Thanks for everything.

Contents

List of Tables

List of Figures

List of Abbreviations

GLMM	Generalized Linear Mixed model
NB	Negative Binomial
OTUs	Operational Taxonomic Units
ZIP	zero-inflated Poisson
ZINB	zero-inflated Negative Binomial
ZMP	zero-modified Poisson
ZMNB	zero-modified Negative Binomial
CDF	Cumulative Distribution Function
PDF	Probability Density Function
PMF	Probability Mass Function
RQRs	Randomized Quantile Residuals
LOOCV	Leave-One-Out Cross-Validation

Chapter 1

Introduction

Metagenomics [1] is a study of microbial communities collected directly from the enviroment by applying genome sequencing methodology. Unlike traditional sequencing technology, which relies on cloning cultivation, the new technology called Next-Generation Sequencing (NGS) [2] provides lower cost for quantifying the microbial communities because cultivation is not necessary. Metagenomics studies have been applied to a variety of areas, such as human health [3], environmental science [4] and industrial production [5].

Despite the ability to generate massive metagenomic sequencing data, the study of microbiome is still challenging. One challenge is that typical data in microbiome, called operational taxonomic units (OTUs), are usually over-dispersed and zero-excessive. The reason for the excessive zeros is either due to absence of OTUs (structural zeros), or presence of OTU but with low frequency which results in observed counts below detection limits (sampling zeros)[6]. One way to deal with excessive zeros is to use a zero-inflated model [7], which is a mixture of a regular count regression model, such as Poisson or negative binomial model, as well as a component to accommodate the excessive zeros. Another way is to use a zero-modified model, also called a hurdle model [8], with one part being a binomial model to determine whether a zero or non-zero outcome occurs, and the other part being the truncated counts regression model to model the positive count data. Unlike zero-inflated models, zero-modified models do not make a distinction between structural and sampling zeros [9].

Moreover, subjects in microbiome data often have clustering structure, for example humans from the same family or plants from the same plot. To model the association of the abundance of OTU with such environment factors, random effects are often used to account for the clustering structure in microbiome study[2]. Most of the previous studies resort to linear mixed models (LMMs) by treating transformed data as normally distributed or negative

binomial mixed models (NBMMs). Such methods may not adequately model zero-inflation and over-dispersion of the response variable. Zero-inflated mixed models and zero-modified mixed models are therefore proposed as alternatives for modelling microbiome count data [9].

Diagnosing various non-normal regression models is essential but still challenging. Pearson and deviance residuals are often used to diagnose model inadequacy. In normal regression, both residuals are normally distributed under the true model; however, in non-normal regression, both Pearson and deviance residuals are far from normality. In particular, in modelling discrete response variables with distinct values or when the number of observations for each covariate pattern are small, Pearson and deviance residuals cluster on curves due to the discreteness, producing little meaningful information for model diagnosis [10].

Randomized quantile residual (RQR) was proposed by Dunn and Smyth [11] to overcome the challenges of diagnosing models for modeling discrete outcomes. The central idea of the RQR is to randomize the lower tail probability (i.e., value of cumulative distribution function (CDF), also called predictive p-value) into a uniform random number between the discrete gap of the CDF. RQR is easy to calculate because it only requires inverting the fitted CDF for each observation y_i and finding the corresponding standard normal quantile. Nevertheless, to the best of our knowledge, RQR has not been applied to diagnose mixed effect models for modeling counts data with excessive zeros and clustering structure.

The contributions of this thesis are to (1) demonstrate how to apply RQR to diagnose mixed effects models for modeling count data with excessive zeros and clustering structure and (2) develop generic R functions that can compute RQRs for zero-inflated and zero-modified GLMM models. We have tested our functions using datasets generated from zero-modified Poisson (ZMP) and zero-modified negative binomial (ZMNB) models. Our simulation studies show that RQRs are normally distributed under the true model and normality of RQRs are theoretically proved by previous research [12]; in GOF tests, the probabilities of rejecting the true model (type 1 error rates) are close to the nominal level 0.05, and the powers of rejecting the wrong models are very good. We have also applied RQRs to assess the model fits in a real microbiome data application and we have found that ZMNB and ZINB provide adequate fits to the OTU counts data.

Chapter 2

Methodology

2.1 Models

2.1.1 Generalized Linear Mixed model (GLMM)

The generalized linear mixed model (GLMM) is an extension of the generalized linear model (GLM)[13] by adding random effects into the linear predictor portion of a GLM to account for the within cluster correlation. The difference between GLM and GLMM is that the former only includes fixed effects while the latter includes both fixed and random effects. GLMM [14] involves three specifications:

- A probability distribution for the response given a mean function and other parameters
- A link function for linking the mean of response to linear predictor
- A linear predictor based on fixed and random variables

As an example of traditional GLM, the exponential family of distributions can be written in the form:

$$f(y_i;\theta_i,\phi) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i,\phi)\right\}, \tag{2.1}$$

where a, b, and c are some functions. θ_i and ϕ are called the canonical parameter and the dispersion parameter respectively.

The link function provides the relationship between the linear predictors and the expected value of the response variables:

$$g(E(y)) = \eta = X\beta + Zu, \tag{2.2}$$

where X is a n by p design matrix containing values of independent variables; $\beta = (\beta_1, \beta_2, \ldots, \beta_p)^T$ is a p-dimensional vector of unknown regression coefficients; Z is a n by q matrix analogous to the fixed effects design matrix X. $u = (u_1, u_2, \ldots, u_q)^T$ is an unobserved random effects vector, which are assumed to be normally distributed $u \sim N(0, G)$, where G is a positive definite variance-covariance matrix.

The expectation of the response can be written as

$$E(y) = g^{-1}(X\beta + Zu) = g^{-1}(\eta). \tag{2.3}$$

Next, we will introduce several types of GLMMs for modeling counts outcome, particularly, the zero-modified and zero-inflated models for handling response variable with excessive zeros.

Poisson Mixed Effects Model

Suppose y_i, $i = 1, \cdots, n$ is a discrete random variable which follows a Poisson distribution with parameter μ_i, We use $\text{dpois}(y_i; \mu_i)$ to represent PMF and $\text{ppois}(y_i; \mu_i)$ to represent CDF. Its probability mass function is then given by

$$\text{dpois}(y_i; \mu_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}. \tag{2.4}$$

The expected mean and variance of y_i are as follows:

$$E(y_i) = \mu_i \tag{2.5}$$

$$V(y_i) = \mu_i. \tag{2.6}$$

Typically we use logarithm to link μ_i to a linear predictor of X_i and Z_i:

$$\log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u, \tag{2.7}$$

where offset_i is a structural predictor. The coefficient of offset is not estimated by the model but is assumed to be 1; thus, the values of the offset are simply added to the linear predictor of the target.

Negative Binomial Mixed Effects Model

Suppose y_i has a negative binomial (NB) distribution with mean μ_i and shape parameter k, we use dnbinom$(y_i; \mu_i, k)$ to represent PMF and pnbinom$(y_i; \mu_i, k)$ to represent CDF. Its probability mass function is the written as,

$$\text{dnbinom}(y_i; \mu_i, k) = \frac{\Gamma(y_i + k)}{\Gamma(k)\Gamma(y_i + 1)} \left(\frac{\mu_i}{\mu_i + k}\right)^{y_i} \left(\frac{k}{\mu_i + k}\right)^{k}, \tag{2.8}$$

where the shape parameter k controls over-dispersion. The expected mean and variance of y_i are

$$E(y_i) = \mu_i \tag{2.9}$$

$$V(y_i) = \mu_i + \frac{\mu_i^2}{k}. \tag{2.10}$$

Typically we use logarithm to link μ_i to a linear predictor of X_i and Z_i:

$$\log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u. \tag{2.11}$$

2.1.2 Zero-Inflated Mixed Effects Models

Zero-inflated models are based on zero-inflated probability distributions which are able to describe count datasets with excessive zeros. The zero-inflated Poisson (ZIP) model consists of two components to distinguish two different zero generating processes. The first part is a binary distribution that generates structural zeros. The second part is a Poisson distribution that generates counts, some of which may be zeroes and are often interpreted as "sampling zeros". The ZIP model defined by [15], can be expressed as follows:

$$y_i \sim \begin{cases} 0 & \text{with probability } p_i \\ \text{Poisson}(\mu_i) & \text{with probability } 1 - p_i, \end{cases} \tag{2.12}$$

where μ_i is the mean of the Poisson model and p_i is the probability of zeros for ith observation belonging to excessive zero component. We denote the PMF and the CDF for a ZIP model

by $\text{dzip}(y_i; \mu_i, k, p_i)$ and $\text{pzip}(y_i; \mu_i, k, p_i)$ respectively, which can be written as:

$$\text{dzip}(y_i = 0) = p_i + (1 - p_i) \times e^{-\mu_i} \tag{2.13}$$

$$\text{dzip}(y_i = j) = (1 - p_i)\frac{e^{-\mu_i}\mu_i^j}{j!}, \text{ for } j > 0 \tag{2.14}$$

$$\text{pzip}(y_i = J; \mu_i, p_i) = \sum_{j=0}^{J} \text{dzip}(y_i = j) = p_i + (1 - p_i)\text{ppois}(J, \mu_i). \tag{2.15}$$

The mean and variance of a ZIP random variable can be calculated by

$$E(y_i) = (1 - p_i) \times \mu_i \tag{2.16}$$

$$V(y_i) = (1 - p_i) \times \left(\mu_i + p_i \times {\mu_i}^2\right). \tag{2.17}$$

The ZIP mixed effects model with log link function is:

$$\log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u \tag{2.18}$$

$$\text{logit}(p_i) = \log(\frac{p_i}{1 - p_i}) = \tilde{X}_i\tilde{\beta} + \tilde{Z}_i\tilde{u}, \tag{2.19}$$

Zero-inflated negative binomial (ZINB) model can be defined analogously. Let $\text{dzinb}(y_i; \mu_i, k, p_i)$ and $\text{pzinb}(y_i; \mu_i, k, p_i)$ denote the PMF and CDF for ZINB, respectively.

$$\text{dzinb}(y_i = 0) = p_i + (1 - p_i) \times \left(\frac{k}{k + \mu_i}\right)^k \tag{2.20}$$

$$\text{dzinb}(y_i = j) = (1 - p_i) \times \text{dnbinom}(j, \mu_i, k), \text{ for } j > 0 \tag{2.21}$$

$$\text{pzinb}(y_i = J; \mu_i, k, p_i) = \sum_{j=0}^{J} \text{dzinb}(y_i = j) = p_i + (1 - p_i)\text{pnbinom}(J, \mu_i, k). \tag{2.22}$$

The function $\text{dnbinom}(y_i, \mu_i, k)$ is given in Equation (2.8). The mean and variance of the ZINB are

$$E(y_i) = (1 - p_i) \times \mu_i \tag{2.23}$$

$$V(y_i) = (1 - p_i) \times \left(\mu_i + \frac{{\mu_i}^2}{k}\right) + {\mu_i}^2 \times \left({p_i}^2 + p_i\right). \tag{2.24}$$

ZINB mixed effects model with the logit link function for can be then written as:

$$\log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u \tag{2.25}$$

$$\text{logit}(p_i) = \log(\frac{p_i}{1 - p_i}) = \tilde{X}_i\tilde{\beta} + \tilde{Z}_i\tilde{u}, \tag{2.26}$$

2.1.3 Zero-Modified Mixed Effects Models

Zero-modified models are also called hurdle models [8]. Both zero-modified and zero-inflated models can be used to model excess zeros in the response variable. A zero-inflated model treats zeros come from two parts: structural zeros and sampling zeros; while a zero-modified model treats zeros are structural zeros [16]. Zero-modified model is composed of two components, i.e., a binomial component, the probability distribution of a random variable takes the value 0 with probability π_i and the positive value with probability $1 - \pi_i$

$$Pr(Z_i = k) = \begin{cases} \pi_i, & k = 0 \\ 1 - \pi_i, & k = 1. \end{cases} \tag{2.27}$$

And another component for modeling the positive count data, which are often modeled as truncated Poisson or truncated negative binomial, by removing the zero part from the Poisson or NB distribution and the denominator is to renormalize the probability so that it still sums to 1. The conditional PMF for Y [17, 18] is then written as:

$$\begin{cases} Pr(y_j|Z_i = 0) = I(y_j = 0) \\ Pr(y_j|Z_i = 1) = \frac{\text{dpois}(y_j)}{1-\text{dpois}(0)} I(y_j > 0), \end{cases} \tag{2.28}$$

where I is the indicator function. Then the unconditional probability mass function for Y is

$$Pr(Y_j = y_i) = \begin{cases} \pi_i & \text{if } y_i = 0 \\ (1 - \pi_i)\frac{\text{dpois}(y_i)}{1-\text{dpois}(0)} & \text{if } y_i > 0. \end{cases} \tag{2.29}$$

We denote PMF and CDF for ZMP distribution by $\text{dzmp}(y_i; \mu_i, \pi_i)$ and $\text{pzmp}(y_i; \mu_i, \pi_i)$ respectively.

$$\text{dzmp}(y_i = 0) = \pi_i \tag{2.30}$$

$$\text{dzmp}(y_i = j) = (1 - \pi_i)\frac{\text{dpois}(j)}{1 - e^{-\mu_i}}, \text{ for } j > 0 \tag{2.31}$$

$$\text{pzmp}(y_i; \mu_i, \pi_i) = \pi_i + (1 - \pi_i)\frac{\text{ppois}(y_i; \mu_i, \pi_i) - \text{ppois}(0)}{1 - \text{ppois}(0)}, \tag{2.32}$$

where π_i is the probability of structural zeroes. The mean and variance for the ZMP model are:

$$E(y_i) = \frac{1-\pi_i}{1-e^{-\mu_i}} \times \mu_i \tag{2.33}$$

$$V(y_i) = \frac{1-\pi_i}{1-e^{-\mu_i}} \times \left(\mu_i + \mu_i^2\right) - \left(\frac{1-\pi_i}{1-e^{-\mu_i}} \times \mu_i\right)^2. \tag{2.34}$$

The ZMP model with the log link function for the truncated Poisson component and the binomial component with the logit link function are then written as:

$$\log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u \tag{2.35}$$

$$\text{logit}(\pi_i) = \log(\frac{\pi_i}{1-\pi_i}) = \tilde{X}_i\tilde{\beta} + \tilde{Z}_i\tilde{u}, \tag{2.36}$$

ZMNB model can be defined analogously [7]. Let $\text{dzmnb}(y_i; \mu_i, k, \pi_i)$ and $\text{pzmnb}(y_i; \mu_i, k, \pi_i)$ denote the PMF and CDF for ZMNB distribution, respectively:

$$\text{dzmnb}(y_i = 0) = \pi_i \tag{2.37}$$

$$\text{dzmnb}(y_i = j) = (1-\pi_i)\frac{\text{dnbinom}(y_i)}{1-\text{pnbinom}(0)}, \text{ for } j > 0 \tag{2.38}$$

$$\text{pzmnb}(y_i; \mu_i, k, \pi_i) = \pi_i + (1-\pi_i)\frac{\text{pnbinom}(y_i; \mu_i, k, \pi_i) - \text{pnbinom}(0)}{1-\text{pnbinom}(0)}, \tag{2.39}$$

where $\text{pnbinom}(0) = \left(\frac{k}{k+\mu_i}\right)^{-k}$. The mean and variance of ZMNB random variable can be calculated by

$$E(y_i) = \frac{1-\pi_i}{1-p_0} \times \mu_i \tag{2.40}$$

$$V(y_i) = \frac{1-\pi_i}{1-p_0} \times \left(\mu_i + \mu_i^2 + \frac{{\mu_i}^2}{k}\right) - \left(\frac{1-\pi_i}{1-p_0} \times \mu_i\right)^2, \tag{2.41}$$

where $p_0 = \text{dnbinom}(0)$. The ZMNB model with a component of truncated NB regression and a binomial model can be written as:

$$\log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u \tag{2.42}$$

$$\text{logit}(\pi_i) = \log(\frac{\pi_i}{1-\pi_i}) = \tilde{X}_i\tilde{\beta} + \tilde{Z}_i\tilde{u}, \tag{2.43}$$

2.2 Parameters Estimation

Package `glmmTMB` [19] is for fitting linear and generalized linear mixed models with various extensions, including zero-inflation. To maximize speed and flexibility, the models are fitted using maximum likelihood estimation via `TMB` (Template Model Builder). Automatic differentiation was used for gradients and Laplace approximation was used for random effects [20]. The package evaluates and maximizes the Laplace approximation of the marginal likelihood where the random effects are automatically integrated out. This approximation, and its derivatives, are obtained using automatic differentiation of the joint likelihood. Here is the review of the Laplace approximation for random effects models [21]. Let $f(u, \theta)$ denote the negative joint log-likelihood of the data and the random effects. The function $f(u, \theta)$ is provided by the TMB user in the form of C++ source code. The `TMB` package implements maximum likelihood estimation and uncertainty calculations for θ and u. The maximum likelihood estimate for θ maximizes

$$L(\theta) = \int \exp(-f(u, \theta))du. \tag{2.44}$$

The random effects u have been integrated out and the marginal likelihood $L(\theta)$ is the likelihood of the data. We use $\hat{u}(\theta)$ to denote the minimizer of $f(u, \theta)$; i.e.,

$$\hat{u}(\theta) = \arg\min_{u} \ f(u, \theta). \tag{2.45}$$

We use $H(\theta)$ to denote the Hessian of $f(u, \theta)$ with respect to u and evaluated at $\hat{u}(\theta)$; i.e.,

$$H(\theta) = f''_{uu}(\hat{u}(\theta), \theta). \tag{2.46}$$

The Laplace approximation for the marginal likelihood $L(\theta)$ is

$$L_*(\theta) = \sqrt{2\pi}^n \det(H(\theta))^{1/2} \exp(-f(\hat{u}, \theta)). \tag{2.47}$$

Our estimate of θ minimizes the negative log of the Laplace approximation; i.e.,

$$-\log L_*(\theta) = -n\log\sqrt{2\pi} + 1/2\log\det(H(\theta)) + f(\hat{u},\theta). \tag{2.48}$$

Given a computer algorithm that defines a function, Automatic Differentiation (AD) can be used to compute derivatives of the function. Source transformation and operator overloading are two different approaches to AD [22].

2.3 Residuals for Checking Models

2.3.1 Pearson Residuals

Examining model goodness of fit (GOF) is an essential step in building a regression model to ensure the fitted regression model is valid. An important way to assess GOF is to examine residuals of a regression model. Pearson residuals can be used to measure the GOF of the model [23], which is the raw residual divided by square root of the variance. The Pearson residuals can be defined as

$$r_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\widehat{V(y_i)}}}, \tag{2.49}$$

where $\hat{\mu}_i$ is the fitted value and $\widehat{V(y_i)}$ is the estimated variance of y_i respectively. The specific formulations of Pearson residuals for some common counts regression models are presented in Table 2.1.

2.3.2 Deviance Residuals

Deviance residuals can be used to measure the GOF of the model as well [24]. The Deviance residuals can be defined as the difference between the log-likelihood functions of the saturated model and the fitted model. The likelihood ratio statistic is

$$2\left\{l\left(\boldsymbol{y};\tilde{\boldsymbol{\mu}}\right) - l\left(\boldsymbol{y};\hat{\boldsymbol{\mu}}\right)\right\} \tag{2.50}$$

where $l(\boldsymbol{y};\hat{\boldsymbol{\mu}})$ and $l(\boldsymbol{y};\tilde{\boldsymbol{\mu}})$ are the log-likelihood for fitted and saturated model, respectively. Deviance residual is defined as signed square root of the component of $D(\boldsymbol{y};\hat{\boldsymbol{\mu}})$ [12], i.e.

$$d_i = \text{sgn}(y_i - \hat{\mu}_i)\sqrt{2\left\{\omega_i\left[y_i(\tilde{\theta}_i - \hat{\theta}_i) - b(\tilde{\theta}_i) + b(\hat{\theta}_i)\right]\right\}} \tag{2.51}$$

where $D(\boldsymbol{y};\hat{\boldsymbol{\mu}}) = \sum_i d_i^2$.

However, it is hard to define deviance residuals when the model is complex because it is not easy to find the saturated model. Therefore, we do not include deviance residuals in the simulation study and real data application.

Table 2.1: Pearson residuals for different models

Model	Pearson Residuals
Poisson	$r_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i}}$
NB	$r_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i + \hat{\mu}_i^2/k}}$
ZIP	$r_i = \frac{y_i - (1-\hat{p}_i)\hat{\mu}_i}{\sqrt{(1-\hat{p}_i)\left(\hat{\mu}_i + \hat{p}_i\hat{\mu}_i^2\right)}}$
ZINB	$r_i = \frac{y_i - (1-\hat{p}_i)\hat{\mu}_i}{\sqrt{(1-\hat{p}_i)\left(\hat{\mu}_i + \frac{\hat{\mu}_i^2}{k}\right) + \hat{\mu}_i^2\left(\hat{p}_i^2 + \hat{p}_i\right)}}$
ZMP	$r_i = \frac{y_i - \frac{1-\hat{\pi}_i}{1-e^{-\hat{\mu}_i}}\hat{\mu}_i}{\sqrt{\frac{1-\hat{\pi}_i}{1-e^{-\hat{\mu}_i}}\left(\hat{\mu}_i + \hat{\mu}_i^2\right) - \left(\frac{1-\hat{\pi}_i}{1-e^{-\hat{\mu}_i}}\hat{\mu}_i\right)^2}}$
ZMNB	$r_i = \frac{y_i - \frac{1-\hat{\pi}_i}{1-p_0}\hat{\mu}_i}{\sqrt{\frac{1-\hat{\pi}_i}{1-p_0}\left(\hat{\mu}_i + \hat{\mu}_i^2 + \frac{\hat{\mu}_i^2}{k}\right) - \left(\frac{1-\hat{\pi}_i}{1-\text{dnbinom}(0)}\hat{\mu}_i\right)^2}}$

2.3.3 Problems with Traditional Residuals

Examining residuals is the key way to check the model fit. However, residuals are not normally distributed under the true model. In theory, the deviance residual is supposed to be much more normal than the Pearson residual, and as $\phi \to 0$ relative to the μ_i, both Pearson and deviance residuals converge to a normal distribution. However, when ϕ/μ_i is

large, none of the residuals are normally distributed even if they have the true fitted value μ_i. That is, for a normal linear model, the residuals are normally distributed and have equal variances. However, in non-normal models, the residuals such as Pearson and deviance residuals are not normal. The reason is that the response variable is discrete and usually have few distinct values [11]. For example, when the mean of Poisson data is close to zero, Pearson and deviance residuals may form nearly parallel curves corresponding to different response values [10]. Therefore, the residual plots are not able to provide useful information as we want.

The overall GOF test using chi-squares is not well-calibrated. Quantitative assessment of the overall GOF with Pearson and deviance residuals are often based on χ^2 approximation for their sampling distributions. The *Pearson* χ^2 *statistic* is written as, $X^2 = \sum_{i=1}^{n} r_i^2$, and the *deviance* ($\chi^2$ *statistic)* is written as, $D = \sum_{i=1}^{n} d_i^2$. The asymptotic distribution of D and X^2 under the true model is often assumed to be χ^2_{n-p}, where n is the sample size and p is the number of parameters. However, the use of this asymptotic distribution for both X^2 and D lacks theoretical underpinning.

2.3.4 Randomized Quantile Residuals

To overcome the difficulties of using traditional residuals for diagnosing regression models for discrete outcomes, RQR [11] was proposed by inverting the fitted distribution function for each response value and finding the equivalent standard normal quantile. Let $F(y; \mu, \phi)$ denote the cumulative distribution function (CDF) for random variable y. If the CDF is continuous, $F(y_i; \mu, \phi)$ is uniformly distributed on the unit interval. RQRs can then be defined as

$$q_i = \Phi^{-1}\{F(y_i; \hat{\mu}_i, \hat{\phi}_i)\}, \tag{2.52}$$

where $\Phi^{-1}()$ is the quantile function of a standard normal distribution. However, if the CDF is discrete, randomization is added to make it continuous. To be more specific, let $p(y; \mu, \phi)$ denote the PMF of y. The CDF can be redefined as:

$$F^*(y; \mu, \phi, u) = \begin{cases} F(y; \mu, \phi), & F \text{ is continuous} \\ F(y^-; \mu, \phi) + u\, p(y; \mu, \phi), & F \text{ is discrete} \end{cases} \tag{2.53}$$

where u is a uniform random variable on $[0,1]$, and $F(Y^-;\mu,\phi)$ is the lower limit of F in y. When F is discrete, we let $a_i = \lim_{y\to y_i^-} F(y;\hat{\mu}_i,\hat{\phi}_i)$ and $b_i = F(y_i;\hat{\mu}_i,\hat{\phi}_i)$, then the randomized quantile residual is

$$q_i = \Phi^{-1}(F_i^*), \tag{2.54}$$

where F_i^* is a uniform random variable on the interval $(a_i, b_i]$, and $q_i \sim N(0,1)$. Therefore, the only information that is required for calculating RQRs is the CDF of the response variable. The definition of RQRs for several typical counts regression models are listed below given the CDF and PMF of the considered model.

To demonstrate the idea of RQR in a regression setting with an outcome variable in relation to a covariate of interest [12], we simulated a response variable of size $n = 1000$ from a Poisson model with

$$\log(\mu_i) = -1 + 2\sin(2x_i),$$

where μ_i is the expected mean count for the ith subject and $x_i \sim \mathit{Uniform}(0, 2\pi)$, $i = 1, \cdots, n$. To illustrate how the RQR can help detect non-linearity of the covariate effect, we fit both the true model and a wrong model-Poisson model with mean structure as follows:

$$\log(\mu_i) = \beta_0 + \beta_1 x_i,$$

where x_i is a predictor with linear effect.

The CDF of the response variable Y_i given x_i (under a considered model with parameters estimated with sample) is denoted by $F(k|x_i) = P(Y_i \leq k|x_i)$, for $k = 0, 1, \cdots$. Figure 2.1 shows $F(k|x_i)$ as a function of x_i, with each coloured line representing a CDF curve associated with value k. The distance between two curves $F(k|x_i)$ and $F(k-1|x_i)$ is the theoretical probability of $y_i = k$ given each x_i. Using the randomized lower tail probabilities $F^*(y_i;\hat{\mu}_i,\hat{\phi},u_i)$, each observed y_i is scattered uniformly to a point between the CDF lines associated with $k = y_i - 1$ and $k = y_i$. This randomized scattering of discrete y_i facilities the comparison of the observed frequency of y_i (fraction of points), and the theoretical frequency (distance of two lines). If the observed frequency and the theoretical frequency agree well, the randomly scattered points of $F^*(y_i;\hat{\mu}_i,\hat{\phi},u_i)$ should be uniformly distributed on $(0,1]$ in each neighbourhood of x_i. Figure 2.1 depicts that, under the true model, the randomized lower tail probabilities are uniformly distributed on $(0,1]$ given each x_i; by contrast, under the wrong

model, the randomized lower tail probabilities are not uniformly distributed, exhibiting a non-linear trend [10].

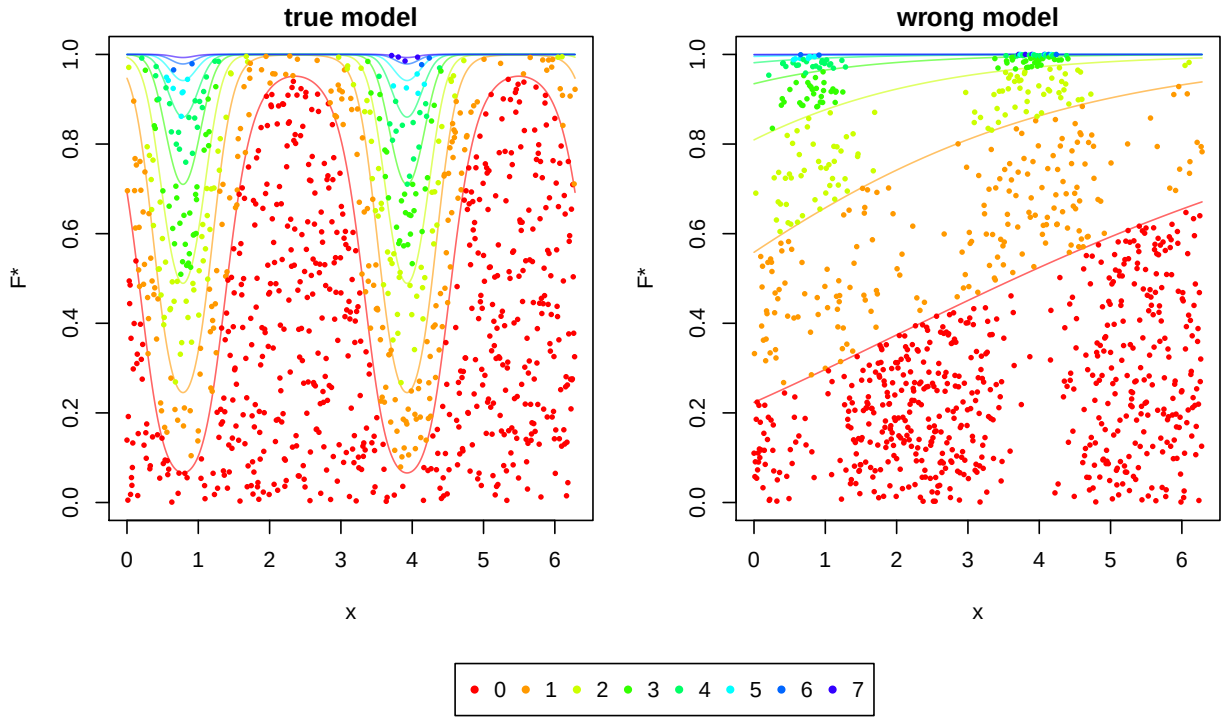


Figure 2.1: An illustration of randomized lower tail probabilities. Each line is a CDF curve of $F(k|x_i)$ versus x_i associated with a value of k. The coloured points show randomized lower tail probabilities $F^*(y_i; \hat{\mu}_i, \hat{\phi}, u_i)$, with colour indicating the value of y_i.

Table 2.2 list the formulae of RQRs for different distributions provided that we can compute the CDF and PMF of the considered models.

We developed two general functions to calculate RQRs for different types of models and these functions are written for outputs by package `glmmTMB`. Function `rqr` is designed for Gaussian, Poisson, NB, ZIP and ZINB models. Function `rqrhurdle` is designed for ZMP and ZMNB models. We just input the fitting results for different types of model, and then these two general functions output the RQRs for the corresponding models. The function is provided in the Appendix A3.1.

Table 2.2: Randomized quantile residuals for different models

Model	Randomized Quantile Residuals
Poisson	$q_i = \Phi^{-1}\Big(\text{ppois}(y_i - 1; \hat{\mu}_i) + u_i \cdot \text{dpois}(y_i; \hat{\mu}_i)\Big)$
NB	$q_i = \Phi^{-1}\Big(\text{pnbinom}(y_i - 1; \hat{\mu}_i, \hat{k}) + u_i \cdot \text{dnbinom}(y_i; \hat{\mu}_i, \hat{k})\Big)$
ZIP	$q_i = \Phi^{-1}\Big(\text{pzip}(y_i - 1; \hat{\mu}_i, \hat{p}_i) + u_i \cdot \text{dzip}(y_i; \hat{\mu}_i, \hat{p}_i)\Big)$
ZINB	$q_i = \Phi^{-1}\Big(\text{pzinb}(y_i - 1; \hat{\mu}_i, \hat{k}, \hat{p}_i) + u_i \cdot \text{dzinb}(y_i; \hat{\mu}_i, \hat{k}, \hat{p}_i)\Big)$
ZMP	$q_i = \Phi^{-1}\Big(\text{pzmp}(y_i - 1; \hat{\mu}_i, \hat{\pi}_i)) + u_i \cdot \text{dzmp}(y_i; \hat{\mu}_i, \hat{\pi}_i)\Big)$
ZMNB	$q_i = \Phi^{-1}\Big(\text{pzmnb}(y_i - 1; \hat{\mu}_i, \hat{k}, \hat{\pi}_i)) + u_i \cdot \text{dzmnb}(y_i; \hat{\mu}_i, \hat{k}, \hat{\pi}_i)\Big)$

2.3.5 Normality Tests

As described in the previous section, we expect the residuals to be roughly normally and independently distributed with a mean of zero and constant variance under a good model. In this section, we review one commonly used normality test: Shapiro-Wilk normality test to determine if a data set is well-modelled by a normal distribution.

Shapiro-Wilk test is a test for normality [25]. The null-hypothesis of this test is that the dataset is normally distributed. Thus, if the p-value is less than the chosen level, then the null hypothesis is rejected and there is evidence that the data are not normally distributed. Whereas if the p-value is greater than the chosen level, then the null hypothesis that the data are normally distributed cannot be rejected. Given an ordered random sample, $y_1 < y_2 < ... < y_n$, the Shapiro-Wilk test statistic is defined as,

$$W = \frac{\left(\sum_{i=1}^{n} a_i y_i\right)^2}{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2}, \tag{2.55}$$

where y_i is the i^{th} order statistic, $\bar{y}$ is the sample mean,

$$\boldsymbol{a}_i = (a_1, ..., a_n) = \frac{m^T V^{-1}}{\left(m^T V^{-1} V^{-1} m\right)^{1/2}}. \tag{2.56}$$

and $\boldsymbol{m} = (m_1, ..., m_n)^T$ are the expected values of the order statistics and $\boldsymbol{V}$ is the covariance

matrix of these order statistics. We usually choose α level of 0.05. As such, if the p-value of the Shaprio-Wilk test for a dataset is less than 0.05, normality of this dataset is rejected; if the p-value is greater than 0.05, the dataset is roughly normally distributed. Shaprio-Wilk test is restricted to sample size $3 \leq n \leq 5000$. Moreover, because of the power of the normality test, p-values of Shapiro-Wilk normality test in replicated experiments are uniformly distribution under the true model.

Chapter 3

Simulation Studies

In this chapter, two synthetic datasets were generated with excess zeros and over-dispersed counts using a ZMP model and a ZMNB model, respectively, which resemble the distributions of OTU in our real application. We assess the GOFs of the true model in comparison with the misspecified models using RQRs and Pearson residuals for a single response variable. Then we replicate the previous steps for all response variables simultaneously to assess the performance of the overall GOF test by testing the normality of the RQRs. The histogram of normality test p-values and probability of rejecting the wrong model are presented for comparing the performance of RQRs and Pearson residuals. Section 3.1 describes data generating process. Section 3.2.1 and section 3.3.1 are for a single response variable while 3.2.2 and section 3.3.2 are for multiple response variables.

3.1 Description of Data Generating Process

A typical microbiome data set consists of the following components, with the data structure provided in Table 3.1:

1. Operational Taxonomic Units (OTUs), Y_{ij}, groups of correlated bacterial taxa at different hierarchical levels;

2. Total sequence read, T_i, the summation of Y_i for each sample, i.e. $T_i = \sum_{j=1}^{m} Y_{ij}$;

3. Fixed factors, X_i, on behalf of host environmental or genetic variables;

4. Random factors, Z_i, on behalf of sample collection identifier in the hierarchical study design.

Table 3.1: Microbiome Data Structure

	Y_1	...	Y_m	log (Total read)	Fixed Factors	Random Factors
sample 1	Y_{11}	...	Y_{1m}	$\log(T_1)$	X_{11} ... X_{1s}	Z_{11} ... Z_{1t}
.	.	...	.	.	...	...
.	.	...	.	.	...	...
.	.	...	.	.	...	...
sample n	Y_{n1}	...	Y_{nm}	$\log(T_n)$	X_{1n} ... X_{ns}	Z_{n1} ... Z_{nt}

We assume that there are m response variables in one typical microbiome data with sample size n in each response variables, and a typical dataset consists of s fixed factors and t random factors.

Parameters settings in the simulation are listed inTable 3.2. We generate a dataset with $n = 800$ samples, each of which contains $m = 3000$ variables. We generate $s = 3$ fixed factors and $t = 3$ random factors, each fixed factor has 5 levels and each random factor has 10 levels. The regression coefficients for the fixed-effects covariates β_i follow a normal distribution with mean $\mu = 0$, and standard deviation $\sigma = 0.1$, and the coefficients for the random effects u_i follow a normal distribution with mean $\mu = 0$, and standard deviation $\sigma = 2$. The total read T_i follows a Poisson distribution with mean $\mu = 300000$. k is the size of ZMNB model, which follows a uniform distribution between 1 and 2.

Table 3.2: Parameter Settings in the Simulation Studies

Parameter	Value
n	800
m	3000
s	3
t	3
β_i	$\mathrm{N}(0, 0.1^2)$
u_i	$\mathrm{N}(0, 2^2)$
k	Unif(1,2)
T_i	Poisson($\mu = 300000$)

3.1.1 Generate Data from ZMP Model

We first describe how to simulate datasets from a ZMP model, which is a two-part model. The first part is a truncated Poisson model and the second part is a logistic model. The mean μ_i^{zmp}, $i = 1, \cdots, n$ of the truncated Poisson model is $\log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u$, where β is the vector of regression coefficients for the host factors X_i and u is the vector of regression coefficients for the sample variables Z_i. We choose $\log(T_i)$ to be offset_i because of relative abundance, however, offset_i still have other different forms [2]. The mean μ_i^{zmp} of the ZMP model can be written more explicitly as:

$$\log(\mu_i^{zmp}) = \log(T_i) + \beta_0 + \beta_{X_i{}^{(1)}} + \beta_{X_i{}^{(2)}} + ... + \beta_{X_i{}^{(s)}} + u_{Z_i{}^{(1)}} + u_{Z_i{}^{(2)}} + ... + u_{Z_i{}^{(t)}}, \quad (3.1)$$

where $\beta_{X_i{}^{(s)}}$ represents the coefficient associated with the sth fixed-effect factor. For observation i, the coefficient of the fixed factor corresponds to the level of fixed factor. $u_{Z_i{}^{(t)}}$ denotes the coefficient associated with the tth random factor. For observation i, the coefficient of the random factor corresponds to the level of random factor. The equation for mean μ_i^{zmp} is an another representation of an additive linear model with indicator variables.

The proportion of zeros, π_i^{zmp} is modeled as a logistic regression model, where $\text{logit}(\pi_i) = \log(\frac{\pi_i}{1-\pi_i}) = \text{offset}_i + \tilde{X}_i\tilde{\beta} + \tilde{Z}_i\tilde{u}$, but here we assume the binomial model without offset_i. $\tilde{\beta}$ is the vector of coefficients for the host factor $\tilde{X}_i$ and $\tilde{u}$ is the vector of coefficients for the sample variable $\tilde{Z}_i$. The logistic component can be written more explicitly as:

$$\log(\frac{\pi_i^{zmp}}{1-\pi_i^{zmp}}) = \tilde{\beta}_0 + \tilde{\beta}_{X_i{}^{(1)}} + \tilde{\beta}_{X_i{}^{(2)}} + ... + \tilde{\beta}_{X_i{}^{(s)}} + \tilde{u}_{Z_i{}^{(1)}} + \tilde{u}_{Z_i{}^{(2)}} + ... + \tilde{u}_{Z_i{}^{(t)}}, \quad (3.2)$$

where $\tilde{\beta}_{X_i{}^{(s)}}$ denotes the coefficient associated with the sth fixed factor. For observation i, the coefficient of the fixed factor corresponds to the level of fixed factor. $\tilde{u}_{Z_i{}^{(t)}}$ denotes the coefficient for the tth random factor.

Below describes the steps for generating one response variable from a ZMP model:

Step 1: A binary variable is generated as an indicator of zeros vs. positive response values with the probability of zeros as π_i, which represents the proportion of zeros in the

response variable. The indicator function can be written as:

$$Z_i = \begin{cases} 0, & \text{with probability } \pi_i^{zmp} \\ 1 & \text{with probability } 1 - \pi_i^{zmp}. \end{cases} \tag{3.3}$$

Step 2: If the indicator $Z_i = 0$, then the response variable $Y_i = 0$.

Step 3: If the indicator $Z_i = 1$, then the response variable follows a truncated Poisson model. That is, $Y_i \sim$ Truncated Poisson(μ_i^{zmp}).

These three steps for generating one response variable simulated from a ZMP model can be realized by the R function rzmpois$(n, \mu_{zmp}, \pi_{zmp})$ based on the package `actuar` [26]. Later, we replicate this process 3000 times to generate 3000 response variables.

3.1.2 Generate Data from ZMNB Model

We also simulated data from a more flexible ZMNB model. The process for generating the response variable from the ZMNB model is similar to the ZMP model. The only difference is that we include $size = k$ as the over-dispersion parameter involved in the NB distribution. The μ_{zmnb} and π_{zmnb} for the ZMNB model is the same, with μ_{zmp} and π_{zmp} in the simulation part, so we keep the step 1 and step 2 the same as described in the previous subsection. However, step 3 is modified as:

Step 3: If the indicator $Z_i = 1$, then the response variable follows truncated NB model. That is, $Y_i \sim$ Truncated Negative Binomial(μ_{zmnb}, k).

The generating process can be realized by the R function rzmnbinom$(n, k, prob, \pi_{zmnb})$ from the R package `actuar`, where $prob = \frac{k}{k+\mu_{zmnb}}$. After one response variable is created, we replicated the the steps 3000 times to generate 3000 response variables.

3.2 Assessing Models for Datasets Simulated from ZMP Model

In this section, we want to compare the ZMP model with the Poisson, ZIP and ZMNB models using RQRs and Pearson residuals. For each simulated dataset, we fit the true model and

other similar models and then compute different types of residuals. To examine the normality of the RQRs and Pearson residuals, we present the quantile-quantile (Q-Q) plots and also apply the Shapiro-Wilk test to test the normality of the residuals.

For all simulated response variables, we replicate the previous steps to see the performance of the overall goodness-of-fit by testing the normality of the RQRs. For replication, we generate 3000 response variables from the ZMP model and expect to see a uniform distribution of the p-values from the Shapiro-Wilk normality test for the RQRs and Pearson residuals. To further evaluate the finite sample properties of RQRs in comparison with Pearson residuals, we conducted the previous investigations using datasets with sample size $n = 200, 400, 800, 1600, 3200$.

3.2.1 Assessing Models for One Response Variable

We first assess model GOF for one response variable simulated from the ZMP model and fit four competing models to the simulated dataset. the R function `glmmTMB` from R package `glmmTMB` was utilized. The four competing models are listed below:

- **ZMP Model (the true model):**

$$y_i \sim \text{ZMP}(\mu_i, \pi_i),\ \log(\mu_i) = \text{offset}_i + X_i\beta + Z_iu,\ \log(\frac{\pi_i}{1-\pi_i}) = X_i\tilde{\beta} + Z_i\tilde{u}; \quad (3.4)$$

- **ZIP Model:**

$$y_i \sim \text{ZIP}(\mu_i, p_i),\ \log(\mu_i) = \text{offset}_i + X_i\beta + Z_iu,\ \log(\frac{p_i}{1-p_i}) = X_i\tilde{\beta} + Z_i\tilde{u}; \quad (3.5)$$

- **ZMNB Model:**

$$y_i \sim \text{ZMNB}(\mu_i, k, \pi_i),\ \log(\mu_i) = \text{offset}_i + X_i\beta + Z_iu,\ \log(\frac{\pi_i}{1-\pi_i}) = X_i\tilde{\beta} + Z_i\tilde{u}; \quad (3.6)$$

- **Poisson Model:**

$$y_i \sim \text{Poisson}(\mu_i),\ \log(\mu_i) = \text{offset}_i + X_i\beta + Z_iu. \quad (3.7)$$

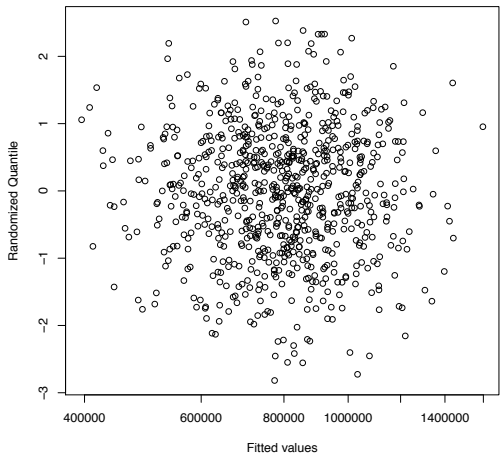
We use RQRs and Pearson residuals to diagnose four competing models to examine the power of the normality test of the residuals for detecting misspecified models. RQRs are

calculated by our created function `rqr` or `rqrhurdle`. Figure 3.1 depicts RQRs versus fitted values. In Figures 3.1a and 3.1b, residuals are randomly scattered around $y = 0$ and do not show any discernible pattern as the standardized fitted value increases, and the standardized residuals are within -3 to 3, which indicates the ZMP model has similar fitting results as the ZIP model and both fit the data well. In Figure 3.1c, no discernible pattern was observed in the RQRs; therefore, the ZMNB model also fits the data well. That is because the ZMP model is a special case of the ZMNB model. However, Figure 3.1d shows that RQRs are clustered at the top and bottom, which demonstrated that the Poisson model cannot handle over-dispersion and excessive zeros. Figure 3.2c indicated that the Pearson residuals fail to provide meaningful information regarding to the GOF of the models, which is not surprising, as Pearson residuals are theoretically not normally distributed for count regression.

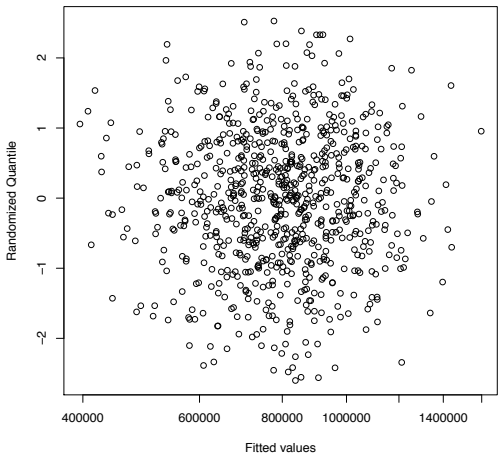
We also depict the Q-Q plots for RQRs in Figure 3.3. Figure 3.3a, 3.3b and 3.3c have straight lines with a slope of 1 and some points are below or above the diagonal line; therefore, the ZMP, ZIP and ZMNB models have the similar results and all fit data well. However, Q-Q plots for Poisson model are depicted as two separate lines with a substantial gap, which clearly indicates that the Poisson model is an undesirable model. Q-Q plots for the Pearson residuals are depicted in Figure 3.4, all of which are curves; therefore it is very challenging to visually check whether the model is good or not. Hence, RQRs have better performance and are more informative for diagnosing model GOF compared to Pearson residuals.

3.2.2 Assessing Models for High-Dimensional Response Variables

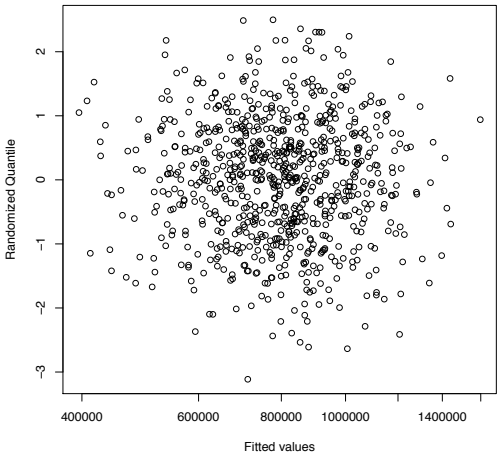
To examine the performance of the GOF of the regression models for all response variables simultaneously by testing the normality of RQRs, we replicate the experiments 3000 times for all response variables simulated from the ZMP model. Figures 3.5a, 3.5b and 3.5c indicate that the p-values of the Shaprio-Wilk normality tests for RQRs under the ZMP, ZIP and ZMNB models are uniformly distributed. Therefore, these three models fit data well for all response variables. However, as shown in Figure 3.5d, the p-values of the for Shapiro-Wilk normality test for RQRs based on the Poisson model are clustered at zero, which indicates that the Poisson model fails to fit the data well. The results for all response variables are consistent with one response variable. Figure 3.6 indicates that the p-values from the Shapiro-



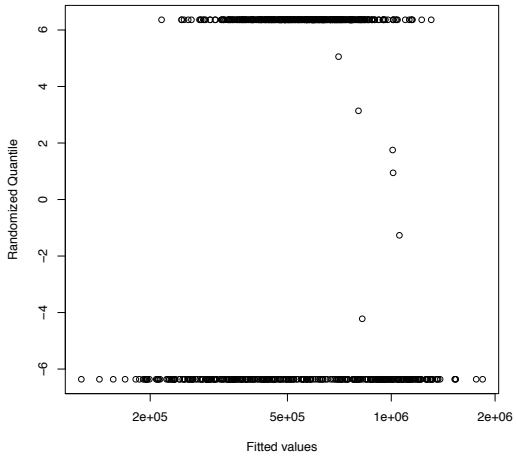
(a) ZMP



(b) ZIP

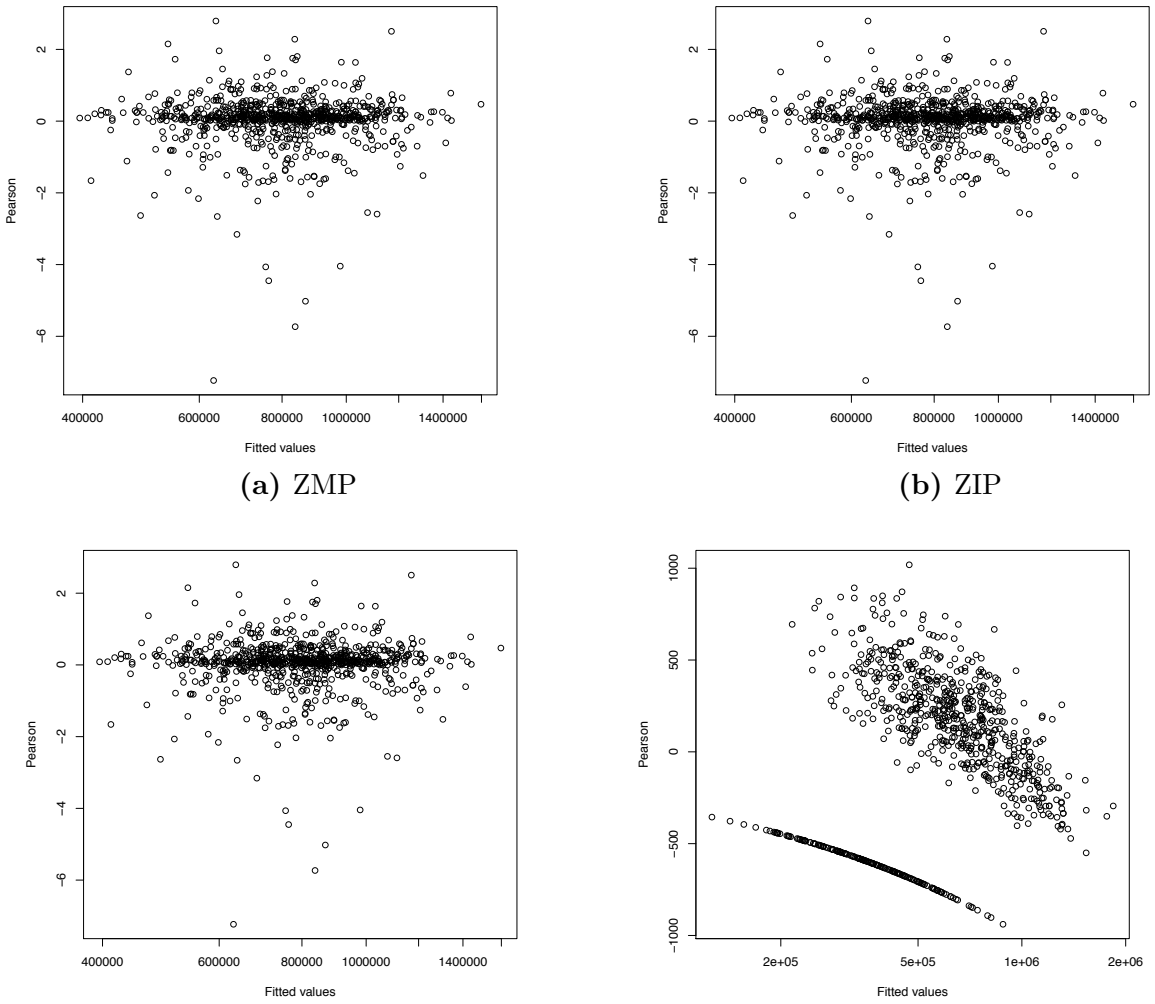


(c) ZMNB



(d) Poisson

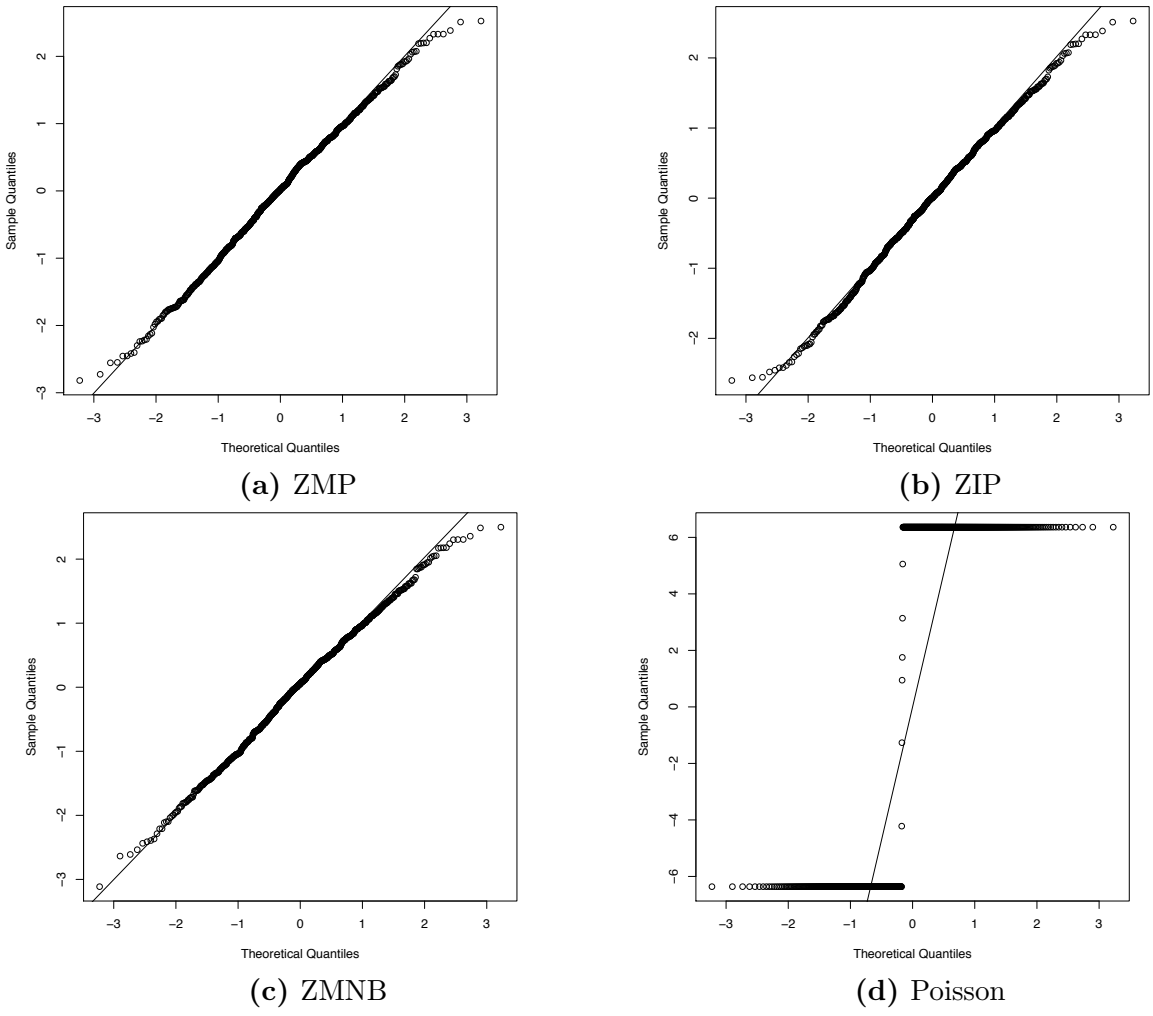
Figure 3.1: RQRs vs. fitted values for ZMP, Poisson, ZIP and ZMNB models when the dataset is simulated from ZMP (n=800).



(a) ZMP **(b)** ZIP

(c) ZMNB **(d)** Poisson

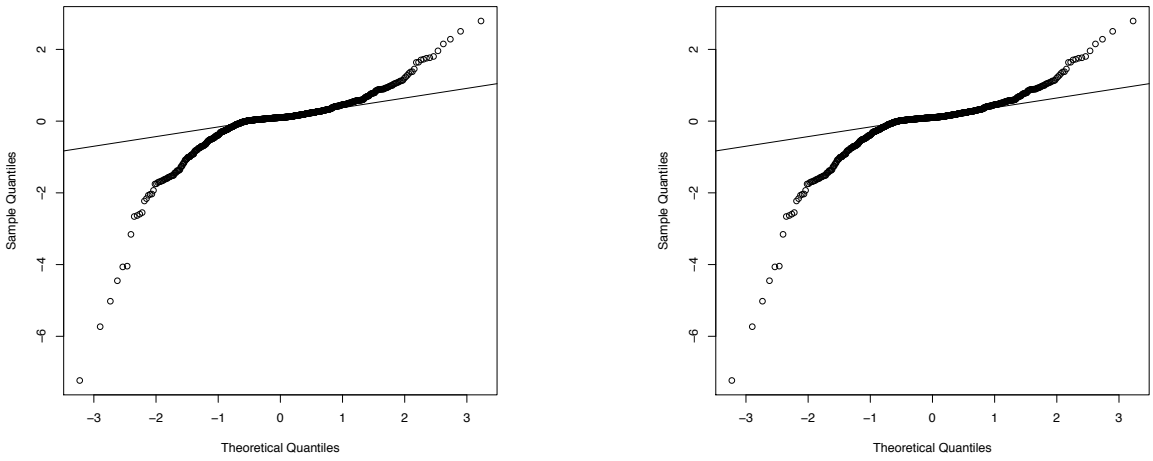
Figure 3.2: Pearson residuals vs. fitted values for ZMP, Poisson, ZIP and ZMNB models when the dataset is simulated from ZMP (n=800).



(a) ZMP **(b)** ZIP

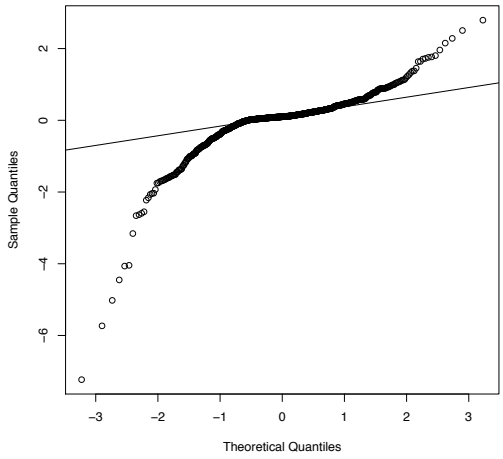
(c) ZMNB **(d)** Poisson

Figure 3.3: Q-Q plots for RQRs for ZMP, Poisson, ZIP and ZMNB models when the dataset is simulated from ZMP (n=800).

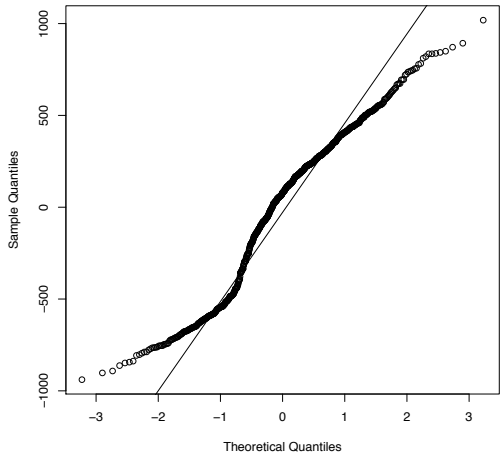


(a) ZMP

(b) ZIP

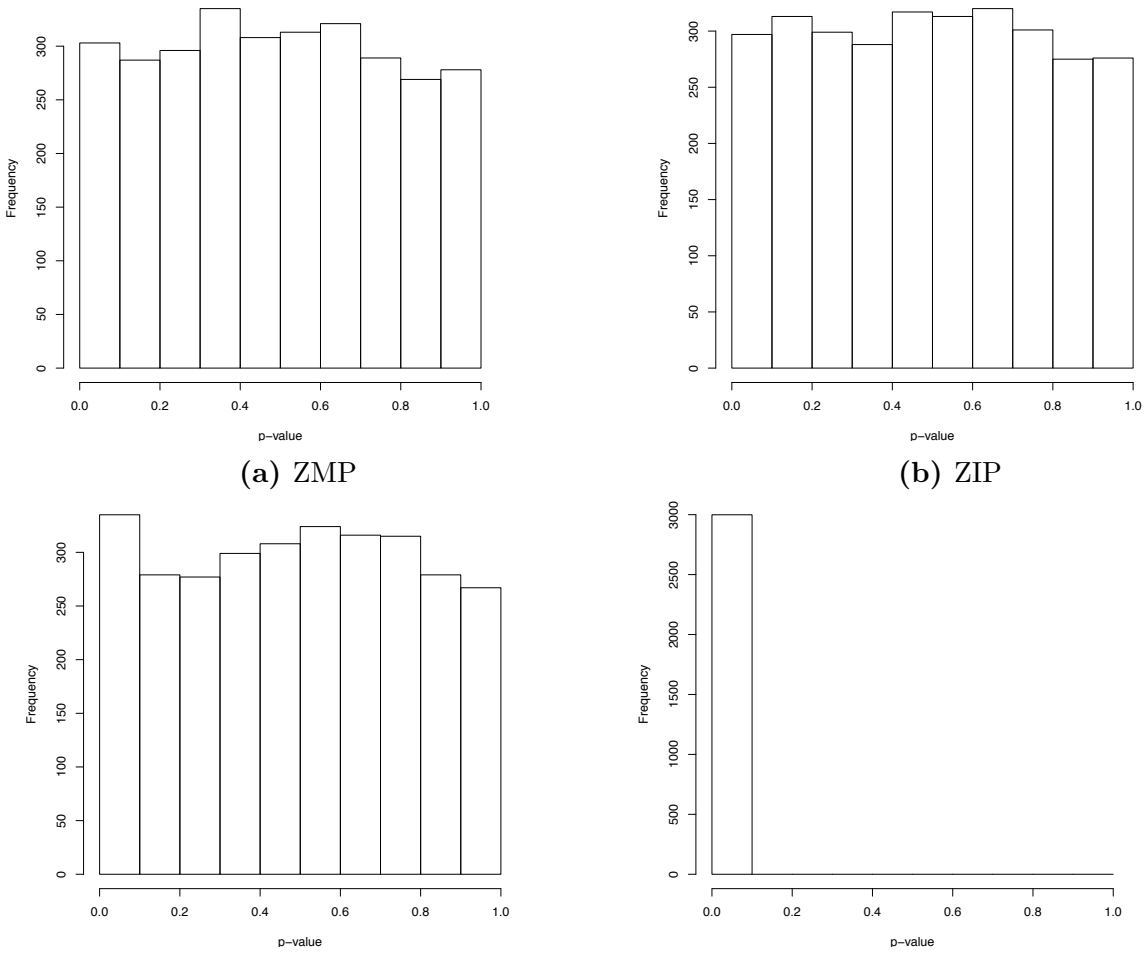


(c) ZMNB



(d) Poisson

Figure 3.4: Q-Q plots for Pearson residuals for ZMP, Poisson, ZIP and ZMNB models when the dataset is simulated from ZMP (n=800).



(a) ZMP **(b)** ZIP

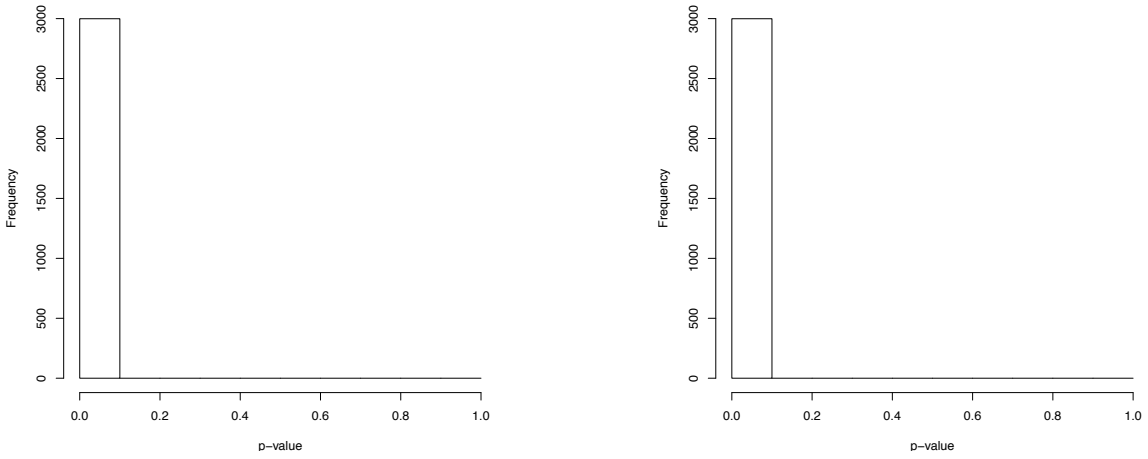
(c) ZMNB **(d)** Poisson

Figure 3.5: Histograms of the p-values for the Shapiro-Wilk test of RQRs for ZMP, Poisson, ZIP and ZMNB models when the dataset is simulated from ZMP (m=3000).

Wilk normality test for the Pearson residuals are all concentrated around zero, therefore Pearson residuals fail to distinguish models.

We also investigated the power of Shaprio-Wilk test for RQRs and Pearson residuals in rejecting incorrectly specified models at varying sample sizes, $n = 200, 400, 800, 1600$ and 3200. Ideally, the type I error of the Shapiro-Wilk test (probability of rejecting the true model) should be around 0.05; while the power (probability of rejecting the wrong model) should be high.

According to Table 3.3, the probability of rejecting zero-modified Poisson model are 0.142, 0.074, 0.068, 0.060 and 0.051 respectively with sample size varying from 200 to 3200. Hence, the type I error rate decreases and approaches to 0.05 as the sample size increases. Despite



(a) ZMP **(b)** ZIP

(c) ZMNB **(d)** Poisson

Figure 3.6: Histograms of the p-values for the Shapiro-Wilk test of Pearson residuals based on the ZMP, Poisson, ZIP and ZMNB models when the dataset is simulated from ZMP (m=3000).

the slightly higher type I error rate, when the sample size is small, i.e., n=200, RQR has substantive appeal in diagnosing models in general. For the ZIP model, the type I error rates are 0.139, 0.090, 0.068, 0.061 and 0.051 at increased sample sizes. As a consequence, both models provide adequate fit to the response variables that are simulated from the ZMP model. The probability of rejecting the ZMNB model are 0.145, 0.102, 0.082, 0.059 and 0.063 respectively, which approaches to 0.05 when the sample size is moderate or large, so the ZMNB model also provides an adequate fit to the data. By comparison, the Shapiro-Wilk test for the RQRs has very good power of rejecting the Poisson model.

Table 3.4 summarizes the probability of rejecting the models for Pearson residuals when the response is simulated from ZMP models with different sample sizes. All models have high probabilities of rejecting models regardless of the sample sizes. For example, the probability of rejecting the true model, i.e., the ZMP model are 0.984, 1.000, 1.000, 1.000 and 1.000 respectively at increased sample sizes. Therefore, Pearson residuals failed to detect the differences among various models.

Table 3.3: Probability of rejecting the models for RQRs when the dataset is simulated from ZMP.

Sample size	ZMP	ZIP	ZMNB	Poisson
200	0.142	0.139	0.145	1.000
400	0.074	0.090	0.102	0.999
800	0.068	0.068	0.082	1.000
1600	0.060	0.061	0.059	1.000
3200	0.051	0.051	0.063	1.000

Table 3.4: Probability of rejecting the models for Pearson residuals when the dataset is simulated from ZMP.

Sample size	ZMP	ZIP	ZMNB	Poisson
200	0.984	0.986	0.983	0.997
400	1.000	1.000	1.000	1.000
800	1.000	1.000	1.000	1.000
1600	1.000	1.000	1.000	1.000
3200	1.000	1.000	1.000	1.000

3.3 Assessing Models for Datasets Simulated from ZMNB Model

In this section, we compare the ZMNB model with the NB, ZINB and ZMP models using RQRs and Pearson residuals. In order to examine the normality of different residuals, we also present the quantile-quantile (Q-Q) plots and apply the Shapiro-Wilk test to test the normality of the residuals.

For all the simulated OTUs, we replicate the previous steps to see the performance of the overall goodness-of-fit by testing the normality of the RQRs. For each replication, we generate 3000 OTUs from the ZMNB model and expect a uniform distribution of the p-values from the Shapiro-Wilk normality test for the RQRs. To further evaluate the behaviour of the RQRs in comparison with Pearson residuals, we conducted the previous investigations using datasets with sample sizes set as $n = 200, 400, 800, 1600$, and 3200.

3.3.1 Assessing Models for One Response Variable

First, we simulate one OUT based on the ZMNB model, and fit all competing models using the function `glmmTMB` from the R package `glmmTMB`. The competing models are listed below:

- **ZMNB Model (the true model):**

$$y_i \sim \text{ZMNB}(\mu_i, k, \pi_i),\ \log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u,\ \log(\frac{\pi_i}{1-\pi_i}) = X_i\tilde{\beta} + Z_i\tilde{u};\ \ (3.8)$$

- **NB Model:**

$$y_i \sim \text{NB}(\mu_i, k), \log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u; \tag{3.9}$$

- **ZINB Model:**

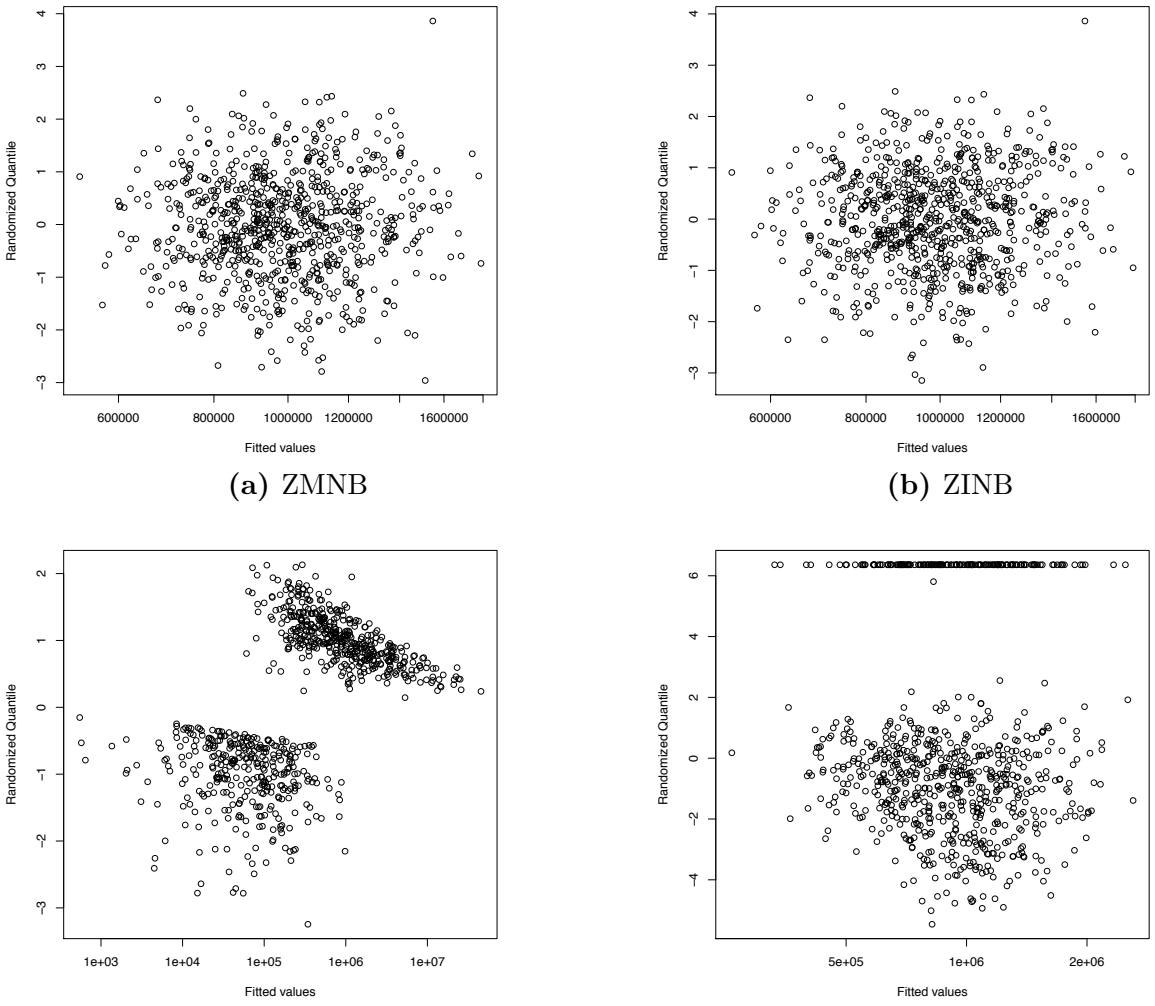
$$y_i \sim \text{ZINB}(\mu_i, k, p_i),\ \log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u,\ \log(\frac{p_i}{1-p_i}) = X_i\tilde{\beta} + Z_i\tilde{u}; \tag{3.10}$$

- **ZMP Model:**

$$y_i \sim \text{ZMP}(\mu_i, \pi_i),\ \log(\mu_i) = \text{offset}_i + X_i\beta + Z_i u,\ \log(\frac{\pi_i}{1-\pi_i}) = X_i\tilde{\beta} + Z_i\tilde{u}. \tag{3.11}$$

RQRs and Pearson residuals were used to diagnose the models. Figure 3.7 depicts the scatter plot of RQRs versus fitted values under the true model. Figure 3.7a and 3.7b indicate that the residuals based on ZINB and ZMNB are randomly scattered around $y = 0$ and do not show any discernible pattern as the standardized fitted value increases and the standardized residuals are within -3 to 3, which indicates ZMNB model has similar fitting results with ZINB model, so both fit the data well. Figures 3.7c and 3.7d indicate that RQRs for the NB model and the ZMP model are not normally distributed with substantial gaps and asymmetry. Therefore, neither model fit the data well. Figure 3.8 shows Pearson residuals versus fitted values, which demonstrates that Pearson residuals fail to distinguish the true and the other competing models, since all the plots are not symmetric with a vast majority of the points concentrating at the bottom.

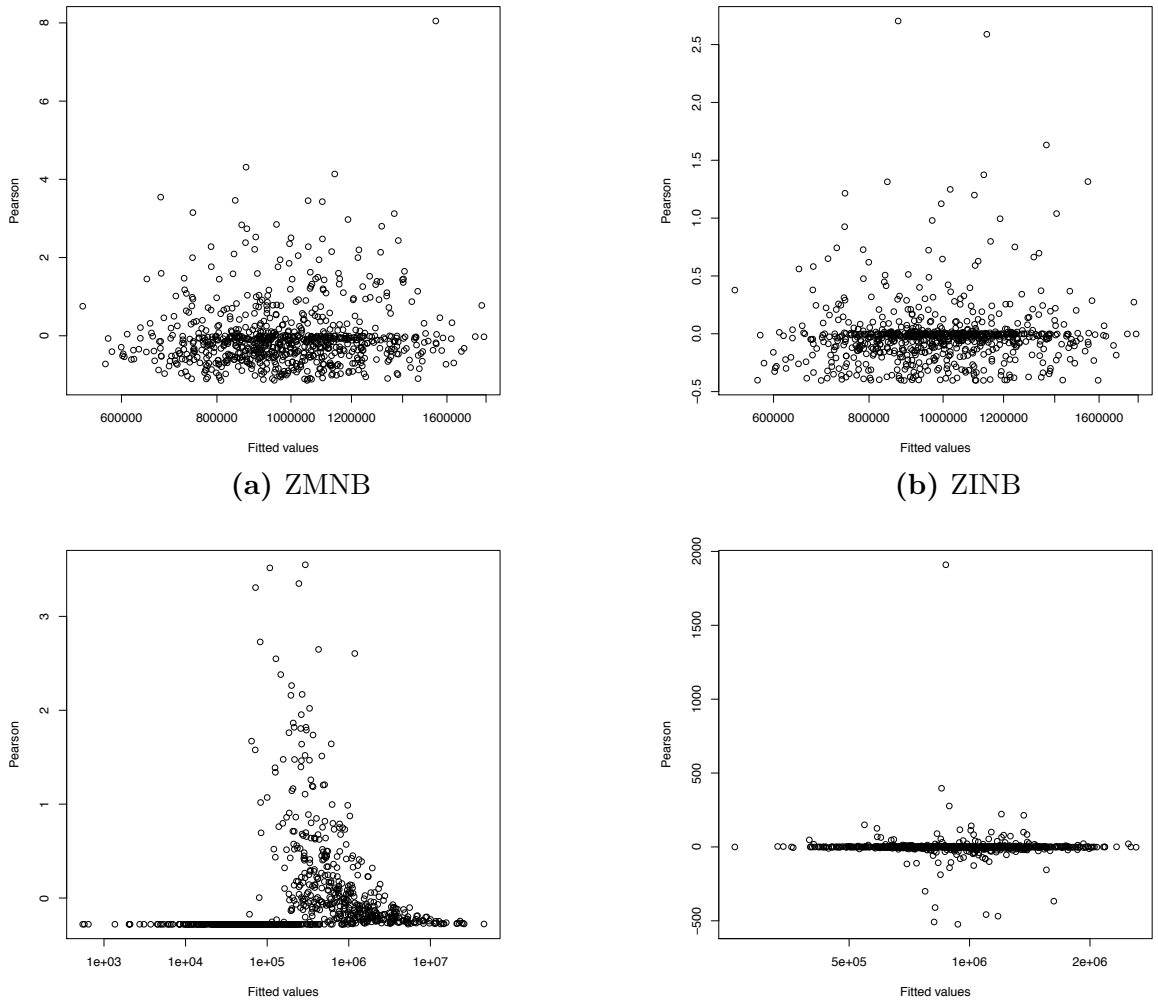
Q-Q plots for RQRs are depicted in Figure 3.9, which shows the relationship between the theoretical quantiles with the sample quantiles. Figures 3.9a and 3.9b present the Q-Q plots of RQRs for the ZMNB and ZINB models, which indicate that both models fit the data well with the points falling along the straight line with a slope of 1. For the NB model and the ZMP model, Q-Q plots have a substantial gap, which indicate that neither is good model for the data. Q-Q plots for Pearson residuals are demonstrated in Figure 3.10, which shows that all the plots exhibit curvature patterns and deviate from the diagonal line even for the true model. Therefore, Pearson residuals cannot differentiate the models.



(a) ZMNB **(b)** ZINB

(c) NB **(d)** ZMP

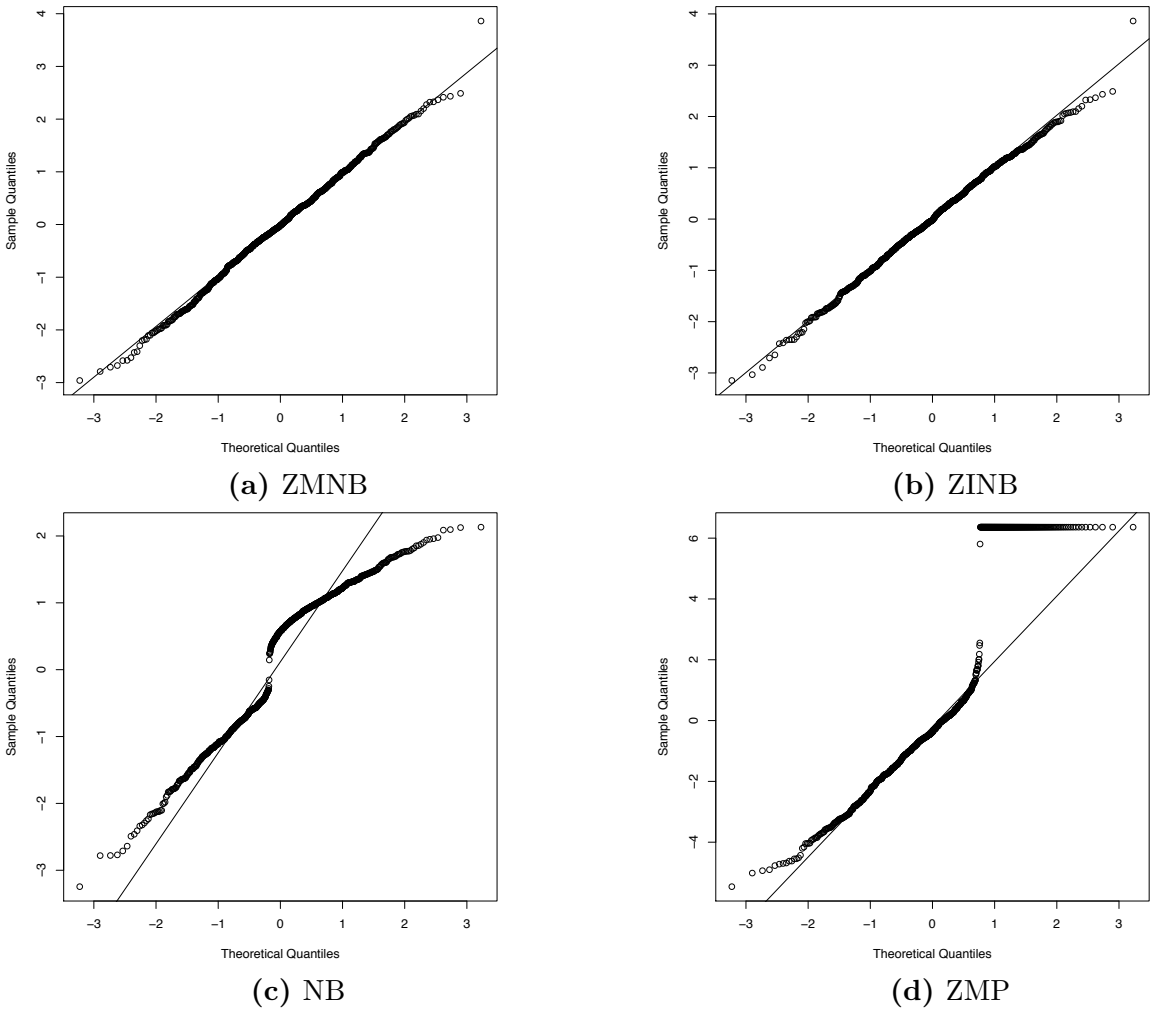
Figure 3.7: RQRs vs. fitted values for ZMNB, NB, ZINB and ZMP models when the dataset is simulated from ZMNB (n=800).



(a) ZMNB **(b)** ZINB

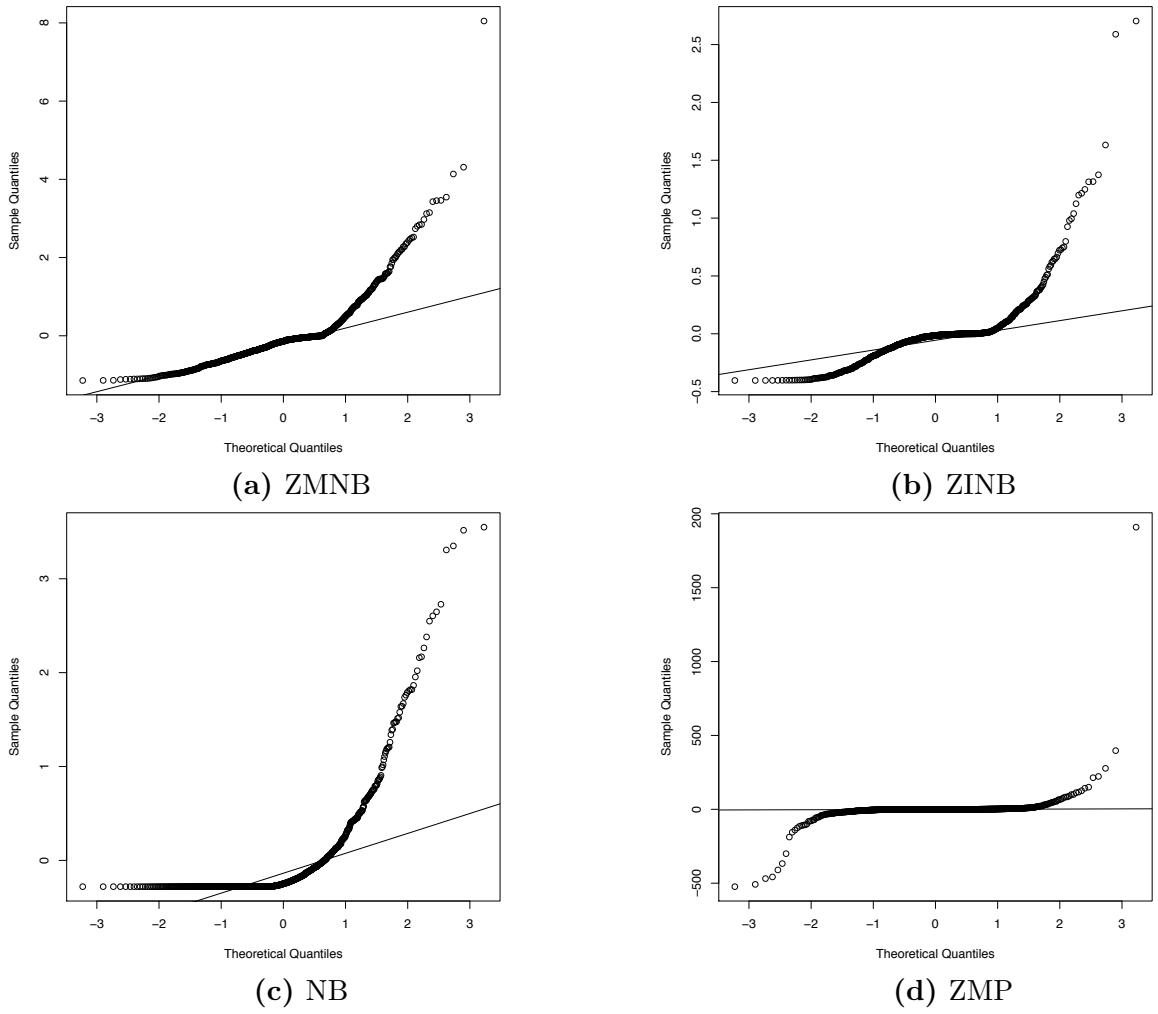
(c) NB **(d)** ZMP

Figure 3.8: Pearson residuals vs. fitted values for ZMNB, NB, ZINB and ZMP models when the dataset is simulated from ZMNB (n=800).



(a) ZMNB **(b)** ZINB **(c)** NB **(d)** ZMP

Figure 3.9: Q-Q plots for RQRs for ZMNB, NB, ZINB and ZMP models when the dataset is simulated from ZMNB (n=800).



(a) ZMNB **(b)** ZINB

(c) NB **(d)** ZMP

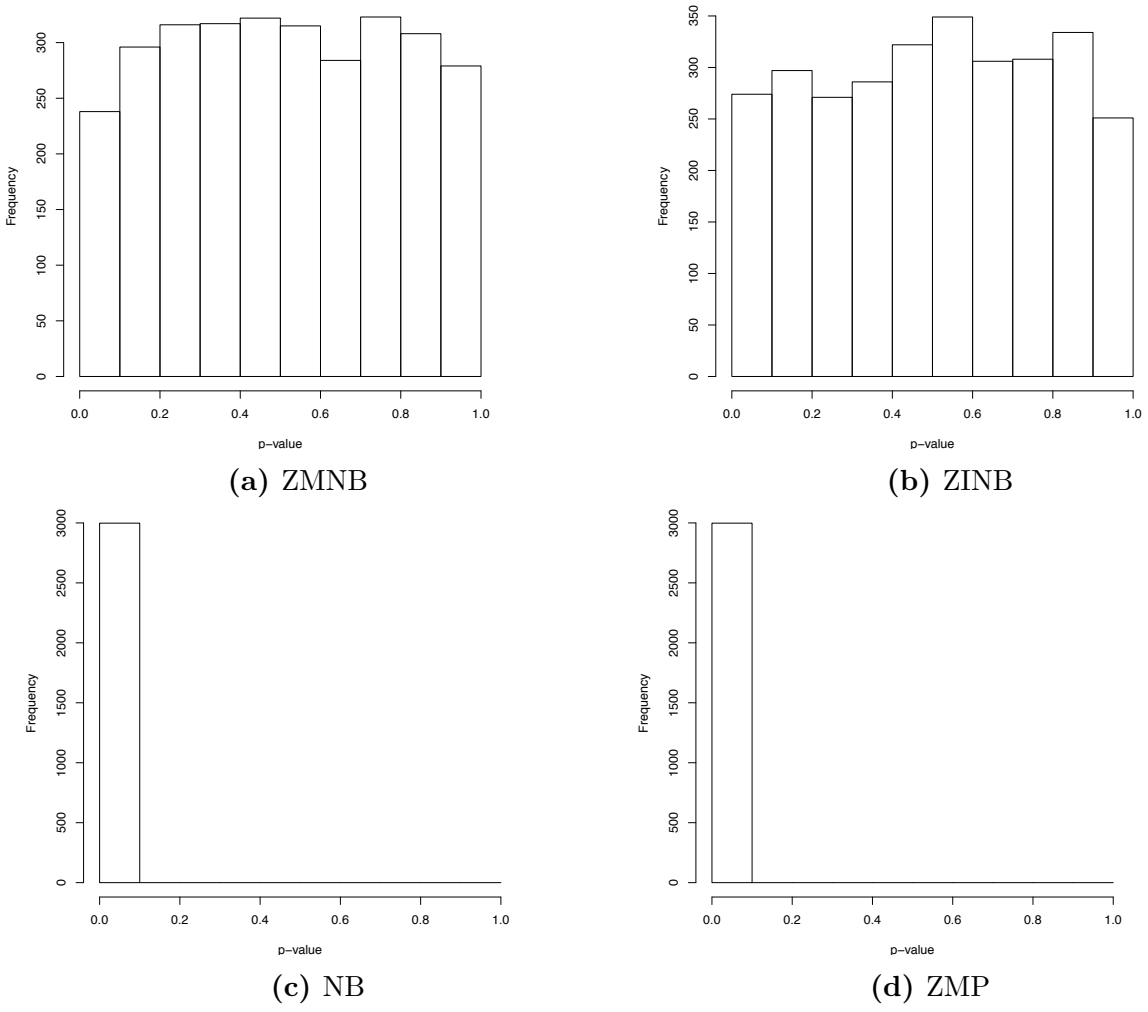
Figure 3.10: Q-Q plots for Pearson residuals for ZMNB, NB, ZINB and ZMP models when the dataset is simulated from ZMNB (n=800).

3.3.2 Assessing Models for High-Dimensional Response Variables

To examine the GOF for all the response variables simultaneously, Shaprio-Wilk normality tests of RQRs are used. We replicate the experiments 3000 times for all response variables simulated from the ZMNB model. Figure 3.11 displays the p-values from the Shapiro-Wilk normality test for the RQRs. Figures 3.11a and 3.11b show that the p-values of the Shaprio-Wilk normality tests of RQRs based on the ZMNB and ZINB models are nearly uniform. Therefore, these two models fit the data well. However, Figures 3.11c and 3.11d show that all the p-values are consistently close to zero and are far from uniform. Both the NB model and the ZMP model do not fit the data well. The results for all response variables are consistent with one response variable. Figure 3.12 displays the p-values from the Shapiro-Wilk normality test for the Pearson residuals, which indicate that all plots are concentrated around zero, so Pearson residuals fail to distinguish the models.

We further examine the power of RQRs and Pearson residuals for diagnosing models at varying sample sizes $n = 200, 400, 800, 1600, 3200$. Table 3.5 indicates that the probabilities of rejecting ZMNB model are 0.067, 0.057, 0.053, 0.047 and 0.040 respectively when the sample size increases from 200 to 3200. Thus, the type I error of RQRs for diagnosing the true model is close to 0.05 as sample size increases. Similarly, the probabilities of rejecting the ZINB model are 0.153, 0.063, 0.049,0.055 and 0.042 as sample size increases, which indicates that the ZINB model also provide adequate fit to the data generated from the ZMNB model. By comparison, the NB model and the ZMP model are undesirable models for the data with extremely high probabilities of rejecting both models based on the RQRs.

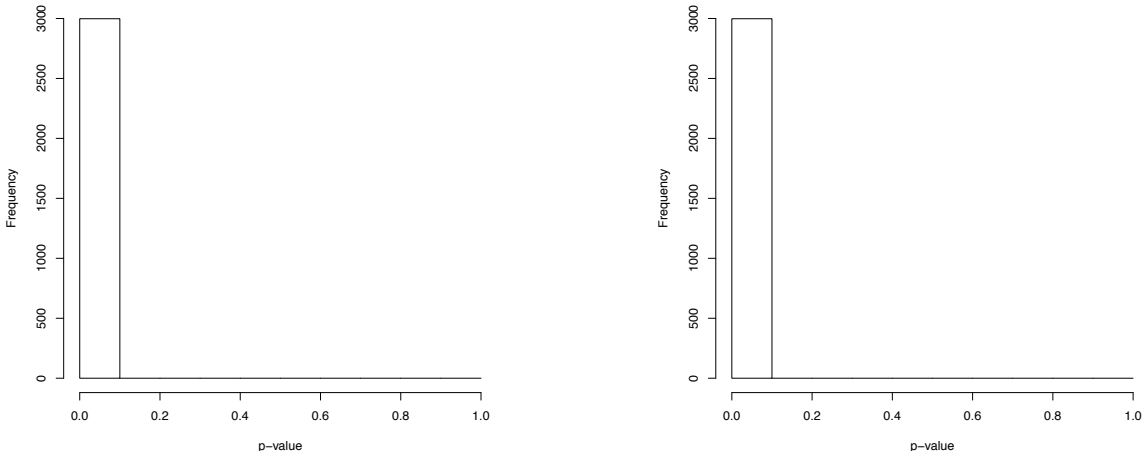
Pearson residuals cannot tell the difference among the compared models. Table 3.6 summarizes the probability of rejecting the models for Pearson residuals when the response is simulated from a ZMNB model at different sample sizes. All models have significantly high probabilities of being rejected. For example, the probabilities of rejecting the ZMNB model are all one at varying sample sizes. Therefore, Pearson residuals are useless compared with RQRs for testing overall GOF of the models.



(a) ZMNB **(b)** ZINB

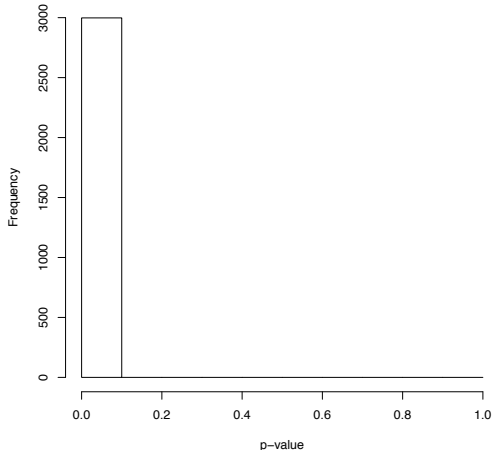
(c) NB **(d)** ZMP

Figure 3.11: Histograms of the p-values for the Shapiro-Wilk test of RQRs for ZMNB, NB, ZINB and ZMP models when the dataset is simulated from ZMNB (m=3000).

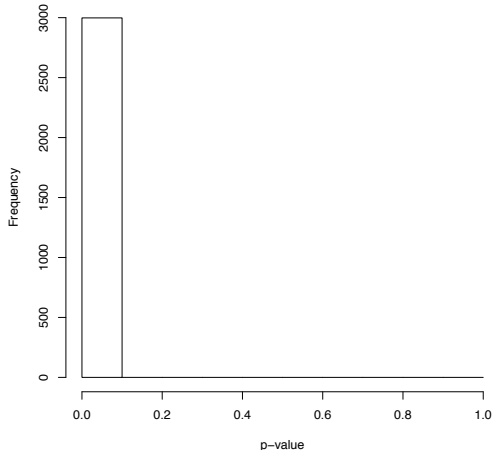


(a) ZMNB

(b) ZINB



(c) NB



(d) ZMP

Figure 3.12: Histograms of the p-values for the Shapiro-Wilk test of the Pearson residuals for ZMNB, NB, ZINB and ZMP models when the dataset is simulated from ZMNB (m=3000).

Table 3.5: Probability of rejecting the model based on RQRs when the dataset is simulated from ZMNB.

Sample size	ZMNB	ZINB	NB	ZMP
200	0.067	0.153	0.957	1.000
400	0.057	0.063	0.883	1.000
800	0.053	0.049	0.759	1.000
1600	0.047	0.055	0.928	1.000
3200	0.040	0.042	1.000	1.000

Table 3.6: Probability of rejecting the model for Pearson residuals when the dataset is simulated from ZMNB.

Sample size	ZMNB	ZINB	NB	ZMP
200	1.000	1.000	1.000	1.000
400	1.000	1.000	1.000	1.000
800	1.000	1.000	1.000	1.000
1600	1.000	1.000	1.000	1.000
3200	1.000	1.000	1.000	1.000

Chapter 4

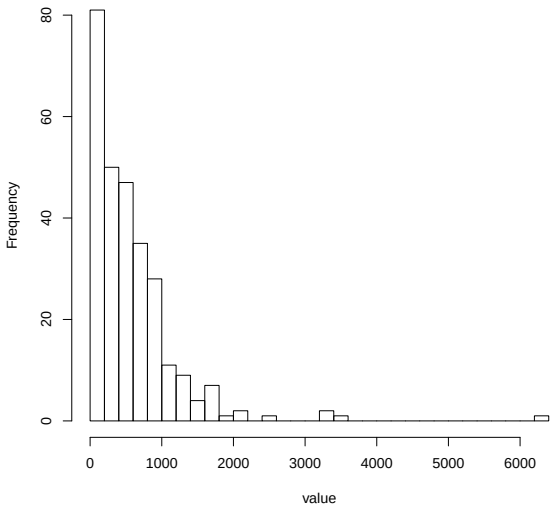
Application to A Real Human Microbiome Dataset

In this chapter, a real human microbiome dataset will be introduced. We apply various models discussed previously to this dataset and use RQRs to test the GOF of all models.

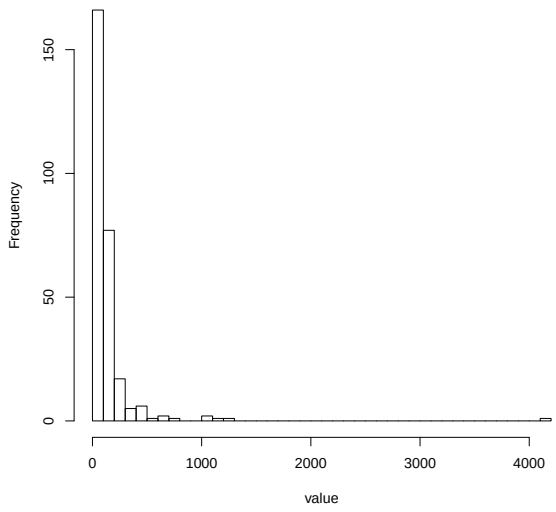
4.1 Data Sources and Descriptions

As a response to the epidemic of worldwide obesity, efforts to identify the relationship between host and environmental factors and energy balance have increased. Comparisons of the distal gut microbiota of genetically obese mice and their lean littermates have revealed that obesity is associated with two dominant bacterial divisions, the Bacteroidetes and the Firmicutes [27]. The human distal gut harbours a vast ensemble of microbes helping to break down otherwise indigestible material. It is often of interest to investigate the relationship between gut microbial ecology and body fat in humans [28]. Each distinct microbe species can be assigned to a diverse taxonomic rank based on shared characteristics, including species, genus, family, order, class, phylum, kingdom, and domain. The OTU data used in our application were generated at the genus level which is the commonly used OTU level for microbiome sequencing analysis and there are 14 different genera in total. Each sample consists of 154 individuals and we characterize individuals into 31 monozygotic (MZ) twin pairs, 23 dizygotic (DZ) twin pairs and 46 mothers. Twins were between 21 and 32 years old and were of European (EA) or African (AA) ancestry respectively. Individuals were classified as obese/overweight if body mass index (BMI) ≥ 25, or lean if BMI < 25. Fecal samples were frozen immediately after they were produced for extracting the DNAs of the bacteria,

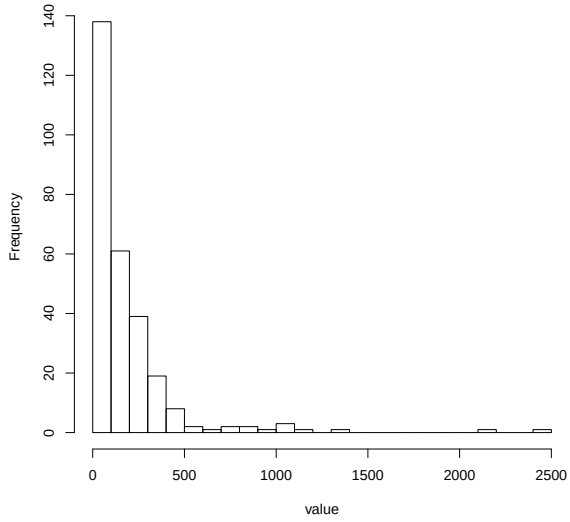
then the 16S rRNA sequencing method was used to group the bacteria into different OTUs with a sequence identity threshold of 97% [6]. Two subjects were dropped from samples for quality control. Among the rest of the152 individuals, 34 of them were measured once and 118 of them were measured twice (timepoint 1 and timepoint 2) for fecal samples. There are 281 OTU measures on the genus level in total. For each measurement, OTU count at each genus level as well as the total number of reads per measure were recorded. Figure 4.1 shows the features of the dataset. We select four genera on behalf of the characteristic of the whole dataset, which have excess zeros and are skewed to large numbers.



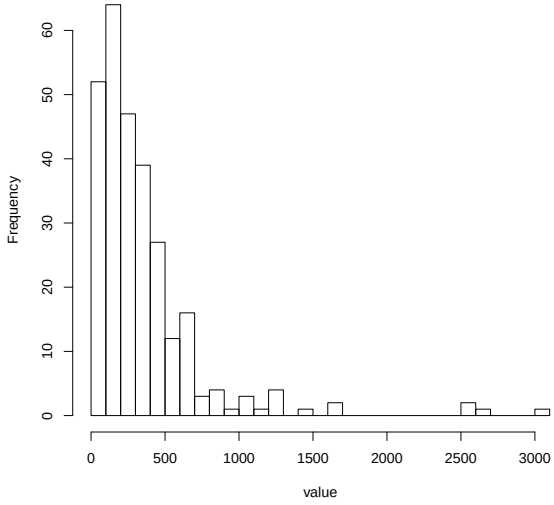
(a) Bacteroides



(b) Lachnospiraceae..g



(c) Roseburia



(d) Faecalibacterium

Figure 4.1: Histogram for some twin study OTUs

4.2 Data Analysis

First, we applied Poisson and NB models to the original dataset, denoted by Poisson1 and NB1, respectively. However, our GOF test based on examining the normality of RQRs showed that these models do not fit the data well (as shown in Figure 4.5), partly because the OTU are clustered by genus; and therefore, each OTU seldom contains zero, even through some of the numbers are very small compared to others. In order to find a desirable model, we set the OTU to be zero when the value of OTU is less than 10 for all genus. However, if the model can not adequately fit the data, we adjust the thresholds for genus until the model is desirable. The threshold for genus Bacteroides and Ruminococcus is set to be 50. The threshold for genus Faecalibacterium is 100 and for genus Lachnospiraceae..g is 150. We choose ancestry and obesity to be host factors while age and family to be sample variables. Then the ZMP model, the ZMNB model, the ZIP model, the ZINB model, the Poisson model and the NB model were applied to fit the dataset. All eight models are listed below:

- **Model 1:**

$$
\begin{aligned}
&y_i \sim \text{ZMP}(\mu_i, \pi_i),\\
&\log(\mu_i) = \log(T_i) + \beta_0 + \beta_{X_i^{(ancestry)}} + \beta_{X_i^{(obesity)}} + u_{Z_i^{(family)}} + u_{Z_i^{(age)}},\\
&\log(\frac{\pi_i}{1-\pi_i}) = \log(T_i) + \tilde{\beta}_0 + \tilde{\beta}_{X_i^{(ancestry)}} + \tilde{\beta}_{X_i^{(obesity)}} + \tilde{u}_{Z_i^{(family)}} + \tilde{u}_{Z_i^{(age)}};
\end{aligned}
$$

- **Model 2:**

$$
\begin{aligned}
&y_i \sim \text{ZMNB}(\mu_i, k, \pi_i),\\
&\log(\mu_i) = \log(T_i) + \beta_0 + \beta_{X_i^{(ancestry)}} + \beta_{X_i^{(obesity)}} + u_{Z_i^{(family)}} + u_{Z_i^{(age)}},\\
&\log(\frac{\pi_i}{1-\pi_i}) = \log(T_i) + \tilde{\beta}_0 + \tilde{\beta}_{X_i^{(ancestry)}} + \tilde{\beta}_{X_i^{(obesity)}} + \tilde{u}_{Z_i^{(family)}} + \tilde{u}_{Z_i^{(age)}};
\end{aligned}
$$

- **Model 3:**

$$
\begin{aligned}
&y_i \sim \text{ZIP}(\mu_i, p_i),\\
&\log(\mu_i) = \log(T_i) + \beta_0 + \beta_{X_i^{(ancestry)}} + \beta_{X_i^{(obesity)}} + u_{Z_i^{(family)}} + u_{Z_i^{(age)}},\\
&\log(\frac{\pi_i}{1-\pi_i}) = \log(T_i) + \tilde{\beta}_0 + \tilde{\beta}_{X_i^{(ancestry)}} + \tilde{\beta}_{X_i^{(obesity)}} + \tilde{u}_{Z_i^{(family)}} + \tilde{u}_{Z_i^{(age)}};
\end{aligned}
$$

- **Model 4:**

$$
\begin{aligned}
&y_i \sim \text{ZINB}(\mu_i, k, p_i),\\
&\log(\mu_i) = \log(T_i) + \beta_0 + \beta_{X_i^{(ancestry)}} + \beta_{X_i^{(obesity)}} + u_{Z_i^{(family)}} + u_{Z_i^{(age)}},\\
&\log(\frac{\pi_i}{1-\pi_i}) = \log(T_i) + \tilde{\beta}_0 + \tilde{\beta}_{X_i^{(ancestry)}} + \tilde{\beta}_{X_i^{(obesity)}} + \tilde{u}_{Z_i^{(family)}} + \tilde{u}_{Z_i^{(age)}};
\end{aligned}
$$

- **Model 5:**

$$
\begin{aligned}
&y_i \sim \text{Poisson}(\mu_i),\\
&\log(\mu_i) = \log(T_i) + \beta_0 + \beta_{X_i^{(ancestry)}} + \beta_{X_i^{(obesity)}} + u_{Z_i^{(family)}} + u_{Z_i^{(age)}};
\end{aligned}
$$

- **Model 6:**

$$
\begin{aligned}
&y_i \sim \text{NB}(\mu_i, k),\\
&\log(\mu_i) = \log(T_i) + \beta_0 + \beta_{X_i^{(ancestry)}} + \beta_{X_i^{(obesity)}} + u_{Z_i^{(family)}} + u_{Z_i^{(age)}};
\end{aligned}
$$

- **Model 7:**

$$
\begin{aligned}
&y_i \sim \text{Poisson1}(\mu_i),\\
&\log(\mu_i) = \log(T_i) + \beta_0 + \beta_{X_i^{(ancestry)}} + \beta_{X_i^{(obesity)}} + u_{Z_i^{(family)}} + u_{Z_i^{(age)}};
\end{aligned}
$$

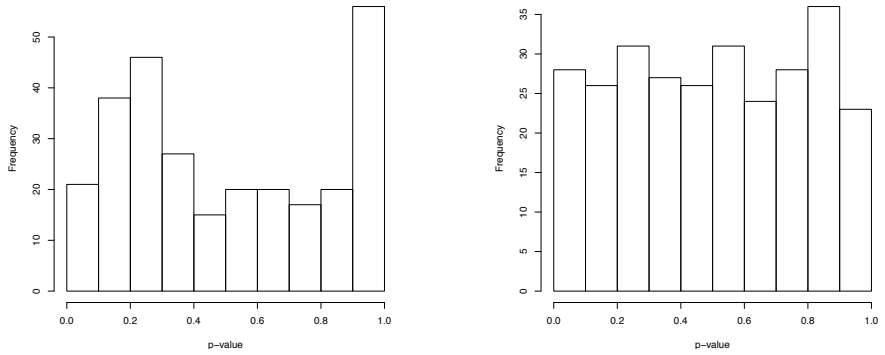
- **Model 8:**

$$
\begin{aligned}
&y_i \sim \text{NB1}(\mu_i, k),\\
&\log(\mu_i) = \log(T_i) + \beta_0 + \beta_{X_i^{(ancestry)}} + \beta_{X_i^{(obesity)}} + u_{Z_i^{(family)}} + u_{Z_i^{(age)}}.
\end{aligned}
$$

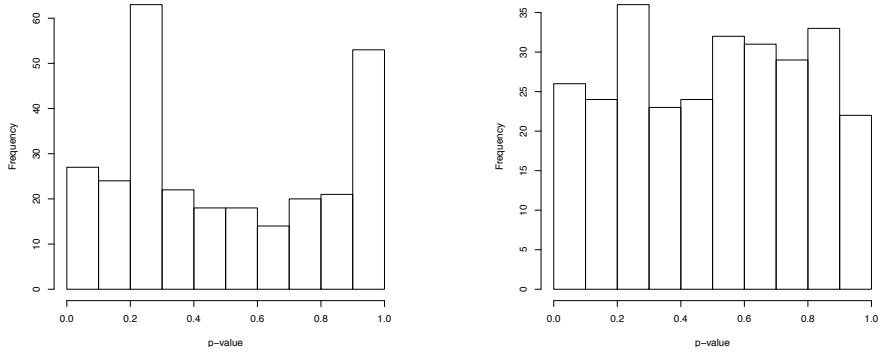
Given different models, we use RQRs to diagnose the models. We first check the GOF for only one OTU and then check the GOF for all of the OTUs.

For only one OTU, if the model fits data well, the histogram of randomized predictive p-values should be uniformly distributed. As shown in Figure 4.2, the histogram for the ZMNB model and the ZINB model are nearly uniform, which suggests these two models are good models. However, the histogram of the ZMP model, the ZIP, the Poisson and the NB models are far from uniform, which indicates those models are undesirable. We also use Q-Q plots for RQRs to verify the results. As shown in Figure 4.4, Q-Q plots for the ZMNB model and the ZINB model fall along a straight line with a slope of 1 and just a few points slightly deviating from the diagonal line, which indicates that RQRs are normally distributed. The Q-Q plots for the other four models exhibit curvature patterns. Therefore, Q-Q plots also verify that only zero-modified NB and ZINB are satisfactory models.

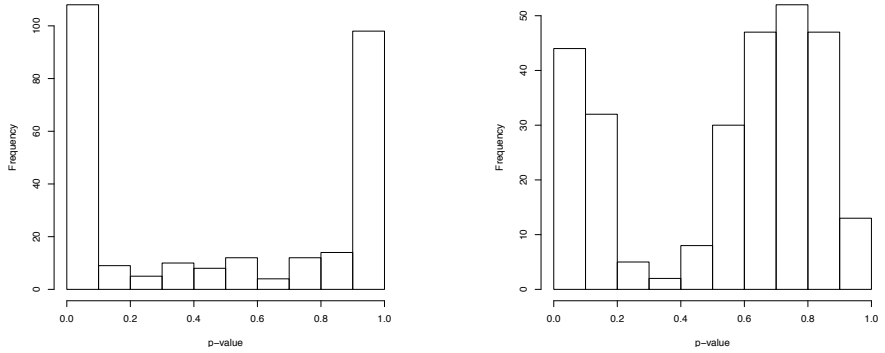
Table 4.1 demonstrates the p-values for the Shapiro-Wilk test of RQRs for all OTUs clustered by genus level. For easy visual inspection of RQRs, we sort p-values for the Shapiro-Wilk test by ZINB model. The first column lists 14 different genus in the twin study OTU data. A good model means the p-value for overall goodness-of-fit should be uniformly distributed between 0 and 1. If the p-value for the Shapiro-Wilk test is less than 0.05, it means that the model may not fit the data well. RQRs contain randomness because the parameter u is randomly chosen from a uniform distribution, therefore, we calculate the mean of the RQR for accuracy by replicating RQR 100 times. From this table, the ZMNB model and the ZINB model are good models with all p-values greater than 0.05. However, the Shapiro-Wilk p-values for the ZMP model, the ZIP model, the NB model, the Poisson model, the NB1 model and the Poisson1 model are mostly less than 0.05, which indicates that these models cannot fit the data well. The ZMNB model and the ZINB model are nearly uniform distributed between 0 and 1. We also draw the histograms of p-values for the Shapiro-Wilk test of RQRs. Figure 4.5 demonstrate that the Shapiro-Wilk p-values of for the ZMNB model and the ZINB model are nearly uniformly distributed while the Shapiro-Wilk p-values for other models are not. We also use Akaike information criterion (AIC) to strengthen the results. The preferred model is the one with the minimum AIC value. Table 4.2 list AIC value for all of the competing models. The ZMNB and ZINB models have smaller AIC value compared to other models, therefore, AIC provides same results with RQR.



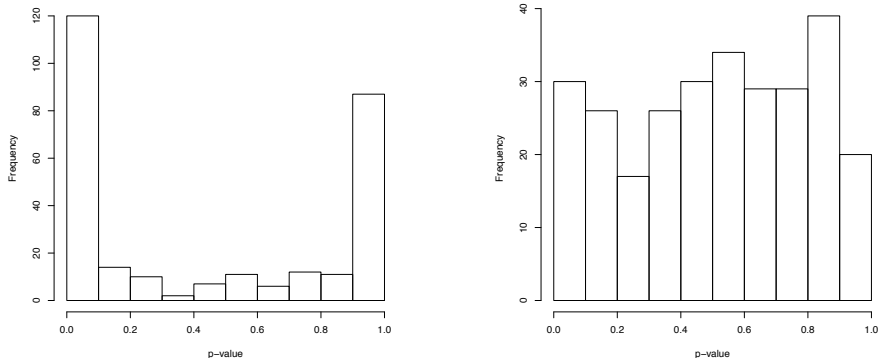
(a) ZMP **(b)** ZMNB



(c) ZIP **(d)** ZINB

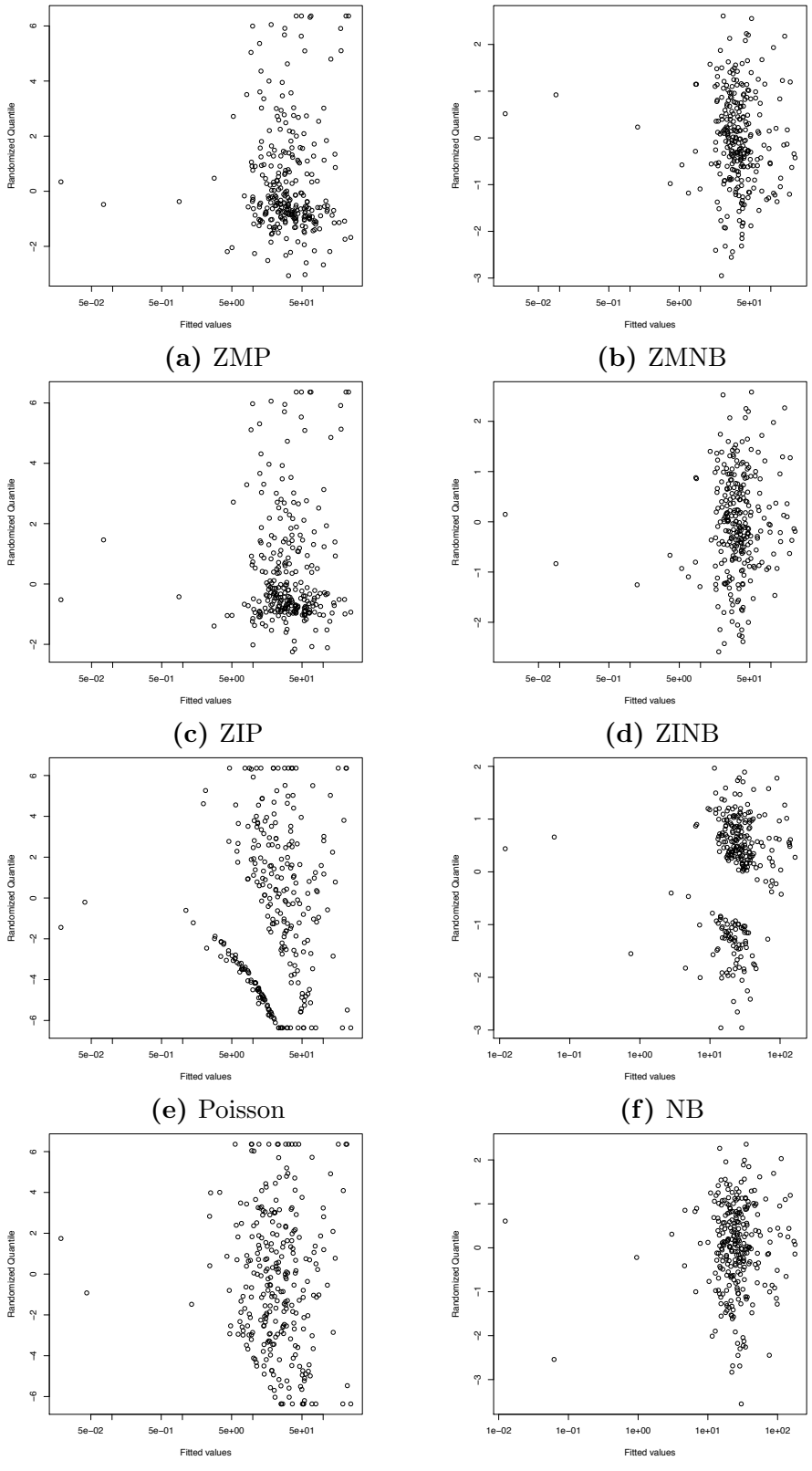


(e) Poisson **(f)** NB



(g) Poisson1 **(h)** NB1

Figure 4.2: Histograms of predictive p-values for ZMP, ZMNB, ZIP, ZINB, Poisson and NB model (genus: Euba).



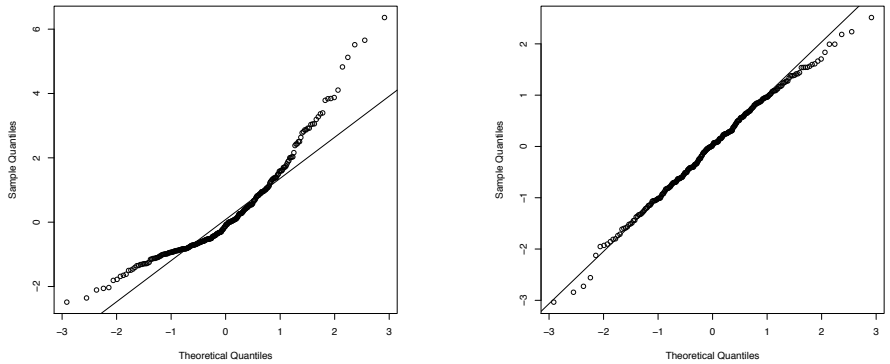
(a) ZMP **(b)** ZMNB

(c) ZIP **(d)** ZINB

(e) Poisson **(f)** NB

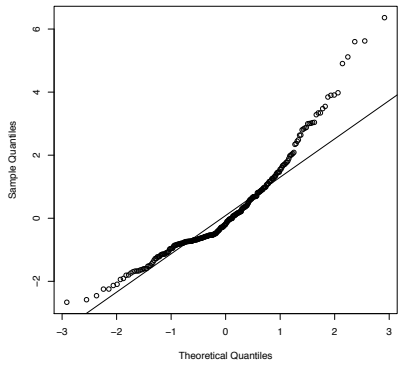
(g) Poisson1 **(h)** NB1

Figure 4.3: RQRs for ZMP, ZMNB, ZIP, ZINB, Poisson and NB models (genus: Euba).

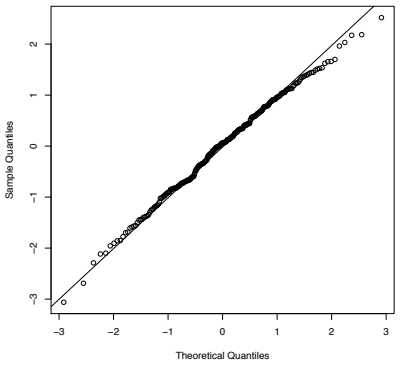


(a) ZMP

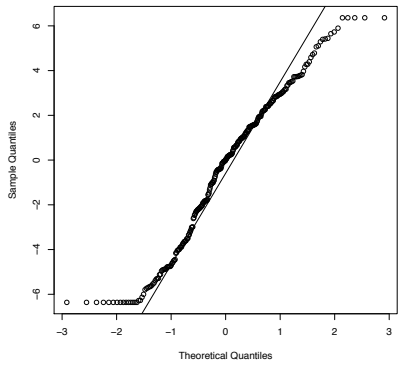
(b) ZMNB



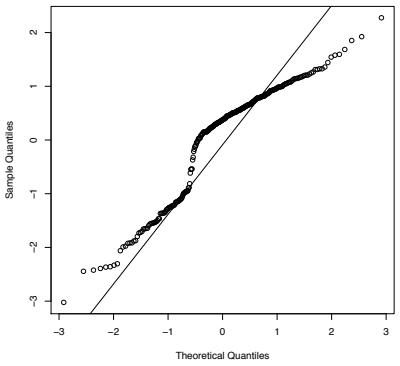
(c) ZIP



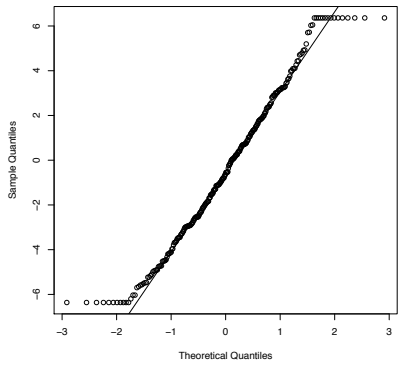
(d) ZINB



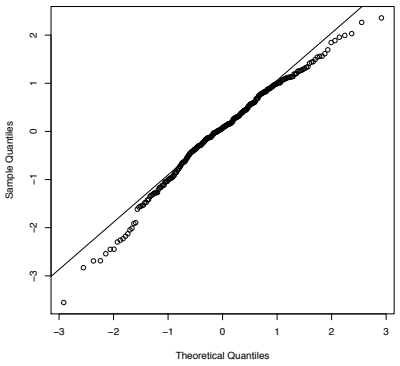
(e) Poisson



(f) NB

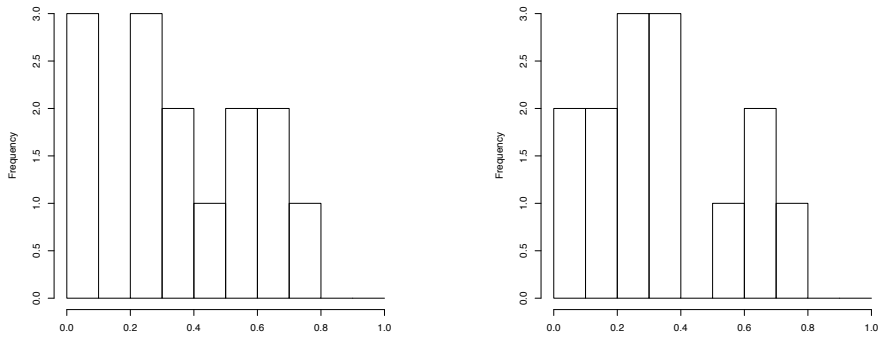


(g) Poisson1



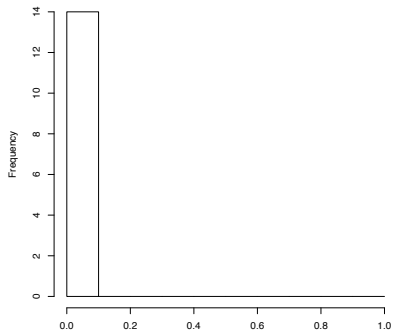
(h) NB1

Figure 4.4: Q-Q plots for RQRs for ZMP, ZMNB, ZIP, ZINB, Poisson and NB models (genus: Euba).

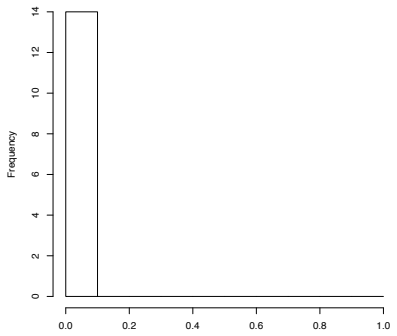


(a) ZMNB

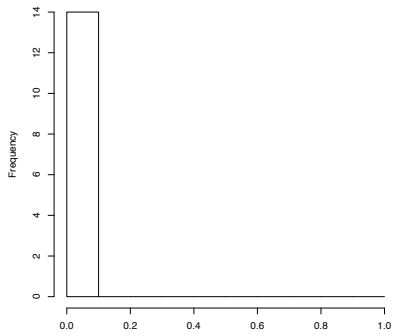
(b) ZINB



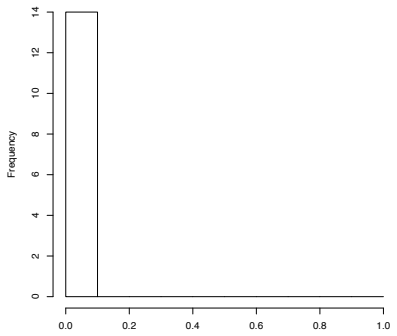
(c) ZMP



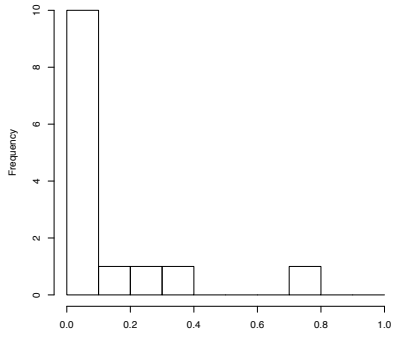
(d) ZIP



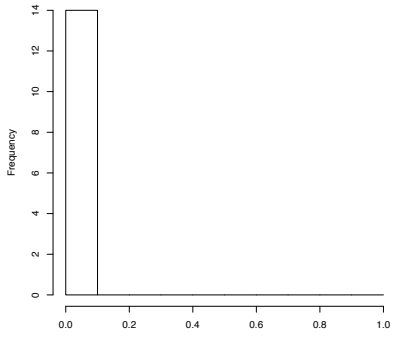
(e) NB



(f) Poisson



(g) NB1



(h) Poisson1

Figure 4.5: Histogram of P-values for the Shapiro-Wilk test of RQRs for twin study OTU data.

Table 4.1: P-values for the Shapiro-Wilk test of RQRs for twin study OTU data sorted by ZMNB model.

Genus	ZMNB	ZINB	ZMP	ZIP	NB	Poisson	NB1	Poisson1
Bact	0.052	0.034	$< 10^{-19}$	$< 10^{-19}$	$< 10^{-16}$	$< 10^{-18}$	$< 10^{-8}$	$< 10^{-17}$
Lach..g	0.072	0.074	$< 10^{-16}$	$< 10^{-15}$	$< 10^{-3}$	$< 10^{-11}$	0.005	$< 10^{-4}$
Faec	0.083	0.107	$< 10^{-17}$	$< 10^{-18}$	$< 10^{-17}$	$< 10^{-15}$	$< 10^{-10}$	$< 10^{-13}$
Rumi	0.232	0.285	$< 10^{-19}$	$< 10^{-19}$	$< 10^{-6}$	$< 10^{-12}$	0.04	$< 10^{-5}$
Rumi.1	0.238	0.366	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-10}$	$< 10^{-11}$	$< 10^{-5}$	$< 10^{-10}$
Blau	0.251	0.104	$< 10^{-10}$	$< 10^{-10}$	0.087	$< 10^{-12}$	0.182	$< 10^{-12}$
Erys	0.344	0.258	$< 10^{-16}$	$< 10^{-17}$	$< 10^{-4}$	$< 10^{-7}$	0.314	$< 10^{-5}$
Alis	0.344	0.352	$< 10^{-16}$	$< 10^{-16}$	$< 10^{-9}$	$< 10^{-7}$	0.003	$< 10^{-6}$
Euba	0.461	0.539	$< 10^{-15}$	$< 10^{-15}$	$< 10^{-10}$	$< 10^{-6}$	0.006	$< 10^{-4}$
Lach	0.521	0.358	$< 10^{-9}$	$< 10^{-10}$	$< 10^{-10}$	$< 10^{-5}$	0.003	0.051
Oscil	0.535	0.606	$< 10^{-15}$	$< 10^{-15}$	$< 10^{-9}$	$< 10^{-5}$	0.006	$< 10^{-4}$
Prev	0.605	0.269	$< 10^{-17}$	$< 10^{-17}$	$< 10^{-4}$	$< 10^{-12}$	0.002	$< 10^{-12}$
Rose	0.627	0.613	$< 10^{-13}$	$< 10^{-14}$	$< 10^{-6}$	$< 10^{-13}$	0.749	$< 10^{-13}$
Copr	0.752	0.721	$< 10^{-13}$	$< 10^{-14}$	$< 10^{-8}$	$< 10^{-6}$	0.245	$< 10^{-6}$

Table 4.2: AIC for the competing models in twin study OTU data

Genus	ZMNB	ZINB	ZMP	ZIP	NB	Poisson	NB1	Poisson1
Bact	3698.20	3689.41	29583.11	30276.44	3954.58	52572.41	3993.04	47727.08
Lach..g	1096.77	1156.08	Inf	5715.61	1317.49	18248.69	2999.31	11174.71
Faec	3328.72	3263.50	13524.32	14132.08	3620.03	29430.38	3677.16	22061.30
Rumi	1597.93	1700.08	4495.07	5007.55	1896.48	13263.94	2800.60	8823.22
Rumi.1	2432.82	2501.87	6993.35	7536.15	2703.78	13199.43	3390.79	13199.43
Blau	3425.68	3401.05	18403.01	18946.44	3396.90	19206.77	3390.79	9344.00
Erys	1530.03	1642.23	4129.93	4585.44	1782.82	9082.96	2850.50	9344.00
Alis	2159.41	2267.34	4768.40	5300.08	2418.27	9055.12	2703.78	13199.43
Euba	2108.20	2123.68	3617.31	4034.78	2292.49	6937.54	2703.78	9344.00
Lach	2089.01	2069.09	2325.09	2702.97	2262.49	5286.85	3390.79	7624.30
Oscil	1941.61	2044.71	3629.43	4104.53	2218.59	7624.30	2703.78	7624.30
Prev	1261.25	1364.03	3757.93	4221.34	1472.21	41117.31	2218.59	7624.30
Rose	3234.63	3272.57	18000.67	18547.76	3340.65	21270.47	2850.50	13199.43
Copr	2848.91	2829.24	6486.43	6949.32	2914.87	8750.86	2218.59	13199.43

Chapter 5

Conclusion and Future Work

In this thesis, we review and extend the randomized quantile residual (RQR) for various mixed-effects models including generalized linear mixed-effects (GLMM) models, zero-inflated, and zero-modified mixed-effects models. We developed generic R functions called `rqr` and `rqrhurdle` for computing RQRs for different models. These functions are written for outputs by the R package `glmmTMB`. In the simulation study, we tested our generic functions using the dataset generated from a ZMP model and a ZMNB model. We made comparisons among different models and showed that RQRs are normally distributed under the true model. We have also replicated the processes and tested the goodness-of-fit with datasets containing high-dimensional response variables. In the GOF tests, the probabilities of rejecting the true model (type 1 error rates) are close to the nominal level 0.05, and the power of rejecting the wrong models are very good. In conclusion, RQR is an excellent tool to compare and diagnose generalized linear mixed-effects models, as well as zero-modified models and zero-inflated mixed-effects models, which can be widely used in many disciplines including human microbiome research.

For further study, we need to correct a problem called optimistic bias in applying RQR. The optimistic bias problem is that actual observations are used twice. The actual observations are not only used to estimate parameters of a corresponding model distribution, but also used to calculate the RQR. The consequence is that even the randomness is applied, the predictive p-values of goodness-of-fit are more concentrated around 0.5 rather than truly uniformly distributed. Leave-one-out cross-validation (LOOCV) is an alternative way to avoid the optimistic bias problem. However, the actual LOOCV is time-consuming because one needs to fit the model once again where an observation is held out as a test case. In Bayesian statistics, there have been a number of computational methods proposed to do model check-

ing with MCMC samples based on the full dataset without actual LOOCV [29]. They can be applied to compute LOOCV RQRs for complex Bayesian methods. RQRs can be applied as an alternative to the widely used posterior predictive checking to check hierarchical Bayesian models in the Bayesian framework [30]. For frequentist statistics, it is worthwhile to find similar tools for approximating LOOCV quantities without actually refitting a model. The deviance residuals for different complex models can be calculated for further study as well.

References

[1] John C Wooley and Yuzhen Ye. Metagenomics: facts and artifacts, and computational challenges. *Journal of computer science and technology*, 25(1):71–81, 2010.

[2] Xinyan Zhang, Himel Mallick, Zaixiang Tang, Lei Zhang, Xiangqin Cui, Andrew K Benson, and Nengjun Yi. Negative binomial mixed models for analyzing microbiome count data. *BMC bioinformatics*, 18(1):4, 2017.

[3] Peter J Turnbaugh, Micah Hamady, Tanya Yatsunenko, Brandi L Cantarel, Alexis Duncan, Ruth E Ley, Mitchell L Sogin, William J Jones, Bruce A Roe, Jason P Affourtit, et al. A core gut microbiome in obese and lean twins. *nature*, 457(7228):480–484, 2009.

[4] Susan M Huse, David Mark Welch, Hilary G Morrison, and Mitchell L Sogin. Ironing out the wrinkles in the rare biosphere through improved otu clustering. *Environmental microbiology*, 12(7):1889–1898, 2010.

[5] Patrick Lorenz and Jürgen Eck. Metagenomics and industrial applications. *Nature Reviews Microbiology*, 3(6):510–516, 2005.

[6] Lizhen Xu, Andrew D Paterson, and Wei Xu. Bayesian latent variable models for hierarchical clustered count outcomes with repeated measures in microbiome studies. *Genetic epidemiology*, 41(3):221–232, 2017.

[7] Alain F Zuur, Elena N Ieno, Neil J Walker, Anatoly A Saveliev, and Graham M Smith. Zero-truncated and zero-inflated models for count data. In *Mixed effects models and extensions in ecology with R*, pages 261–293. Springer, 2009.

[8] Martin Ridout, Clarice GB Demétrio, and John Hinde. Models for count data with many zeros. In *Proceedings of the XIXth international biometric conference*, volume 19, pages 179–192, 1998.

[9] Lizhen Xu, Andrew D Paterson, Williams Turpin, and Wei Xu. Assessment and selection of competing models for zero-inflated microbiome data. *PloS one*, 10(7):e0129606, 2015.

[10] Cindy Feng, Alireza Sadeghpour, and Longhai Li. Randomized quantile residuals: an omnibus model diagnostic tool with unified reference distribution. *arXiv preprint arXiv:1708.08527*, 2017.

[11] Peter K Dunn and Gordon K Smyth. Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, 5(3):236–244, 1996.

[12] Alireza Sadeghpour. Empirical investigation of randomized quantile residuals for diagnosis of non-normal regression models. Master's thesis, University of Saskatchewan, 2016.

[13] Charles E McCulloch. Generalized linear mixed models. In *NSF-CBMS regional conference series in probability and statistics*, pages i–84. JSTOR, 2003.

[14] Ulf Olsson. Generalized linear models. *An applied approach. Studentlitteratur, Lund*, 18, 2002.

[15] Diane Lambert. Zero-inflated poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1):1–14, 1992.

[16] Mei-Chen Hu, Martina Pavlicova, and Edward V Nunes. Zero-inflated and hurdle models of count data with extra zeros: examples from an hiv-risk reduction intervention trial. *The American journal of drug and alcohol abuse*, 37(5):367–375, 2011.

[17] Allen McDowell et al. From the help desk: hurdle models. *The Stata Journal*, 3(2):178–184, 2003.

[18] Ekkehart Dietz and Dankmar Böhning. On estimation of the poisson parameter in zero-modified poisson models. *Computational Statistics & Data Analysis*, 34(4):441–459, 2000.

[19] Ben Bolker. Getting started with the glmmtmb package. 2018.

[20] Mollie E Brooks, Kasper Kristensen, Koen J van Benthem, Arni Magnusson, Casper W Berg, Anders Nielsen, Hans J Skaug, Martin Machler, and Benjamin M Bolker. glmmtmb balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. *The R journal*, 9(2):378–400, 2017.

[21] Mollie E Brooks, Kasper Kristensen, Koen J van Benthem, Arni Magnusson, Casper W Berg, Anders Nielsen, Hans J Skaug, Martin Maechler, and Benjamin M Bolker. Modeling zero-inflated count data with glmmtmb. *bioRxiv*, page 132753, 2017.

[22] Kasper Kristensen, Anders Nielsen, Casper W Berg, Hans Skaug, and Brad Bell. Tmb: automatic differentiation and laplace approximation. *arXiv preprint arXiv:1509.00660*, 2015.

[23] Donald A Pierce and Daniel W Schafer. Residuals in generalized linear models. *Journal of the American Statistical Association*, 81(396):977–986, 1986.

[24] Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 1989.

[25] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[26] Christophe Dutang, Vincent Goulet, Mathieu Pigeon, et al. actuar: An r package for actuarial science. *Journal of Statistical software*, 25(7):1–37, 2008.

[27] Peter J Turnbaugh, Ruth E Ley, Michael A Mahowald, Vincent Magrini, Elaine R Mardis, and Jeffrey I Gordon. An obesity-associated gut microbiome with increased capacity for energy harvest. *nature*, 444(7122):1027, 2006.

[28] Ruth E Ley, Peter J Turnbaugh, Samuel Klein, and Jeffrey I Gordon. Microbial ecology: human gut microbes associated with obesity. *Nature*, 444(7122):1022, 2006.

[29] Longhai Li, Cindy X Feng, and Shi Qiu. Estimating cross-validatory predictive p-values with integrated importance sampling for disease mapping models. *Statistics in medicine*, 36(14):2220–2236, 2017.

[30] Andrew Gelman, Xiao-Li Meng, and Hal Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, pages 733–760, 1996.

Appendix

R Code

A.1 Generic Functions for Computing RQRs and Pearson Residuals

```
###General function "rqr" for calculating RQRs
rqr<-function(object)
{
  family<-family(object)$family
  mu<-predict(object,zitype="conditional")
  size<-sigma(object)
  p<-predict(object,zitype="zprob")
  n<-object$modelInfo$nobs
  y<-object$frame[,1]
  dzpois <- function(x,lambda,p)
  {return((1-p)*dpois(x,lambda)+p*(x==0))}
  pzpois <- function(x,lambda,p)
  {return((1-p)*ppois(x,lambda)+p*(x>=0))}
  dznbinom <- function(x,size,mu,p)
  {return((1-p)*dnbinom(x,size = size, mu = mu)+p*(x==0))}
  pznbinom <- function(x,size,mu,p)
  {return((1-p)*pnbinom(x,size = size, mu = mu)+p*(x>=0))}
  if(1*(object$modelInfo$allForm$ziformula==~0)==1)##no zero-inflation
  {
    if(family[1]=="gaussian")
    {
      pvalue=pnorm(y,mu,size)
    }
    else if(family[1]=="poisson")
    {
      pvalue=ppois (y-1,mu) + dpois (y,mu) * runif(n)
    }
    else if(family[1]=="nbinom2")
    {
      pvalue=pnbinom (y-1,size=size, mu=mu) + dnbinom (y,size = size, mu = mu) * runif(n)
    }
  }
  else if(1*(object$modelInfo$allForm$ziformula==~0)==0)##zero-inflation
  {
    if(family[1]=="poisson")
    {
      pvalue=pzpois(y-1,mu,p) + dzpois (y, mu,p) * runif(n)
    }
    else if(family[1]=="nbinom2")
    {
      pvalue=pznbinom (y-1,size,mu,p) + dznbinom (y,size,mu,p) * runif(n)
    }
```

```
  }
  pvalue<-pmin(pmax(pvalue,10^{-10}),1-10^{-10})
  qnorm=qnorm(pvalue)
  test=shapiro.test(qnorm)$p.value
  list(pvalue=pvalue,qnorm=qnorm,test=test)
}

###General function "rqrhurdle" for calculating RQRs

rqrhurdle<-function(model_count, model_zero, data)
{
  library(actuar)
  name.y <- names(model_count$frame)[[1]]
  y <- data[,name.y]
  m<-model_zero$modelInfo$nobs
  family<-family(model_count)$family
  mu<-predict(model_count,newdata=data,zitype="conditional")
  size<-sigma(model_count)
  ## success rate in negnative binomial
  prob_nb<-size/(size+mu)
  ## probability of 0
  #pi<- predict(model_zero,zitype="zprob")
  pi<- 1-predict(model_zero,newdata = data, type="response")
  if(family[1]=="truncated_poisson")
   {
    pvalue=pzmpois(y-1,mu,pi)+dzmpois(y,mu,pi)* runif(m)
    }
   else if(family[1]=="truncated_nbinom2")
    {
     pvalue=pzmnbinom(y-1,size,prob_nb,pi)+dzmnbinom(y,size,prob_nb,pi)*runif(m)
   }
  pvalue<-pmin(pmax(pvalue,10^{-10}),1-10^{-10})
  qnorm=qnorm(pvalue)
  test=shapiro.test(qnorm)$p.value
  list(pvalue=pvalue,qnorm=qnorm,test=test)
}
##functions for calculating pearson residuals
pearson<-function(object)
{
 family<-family(object)$family
 mu<-predict(object,zitype="conditional")
 size<-sigma(object)
 p0<-predict(object,zitype="zprob")
 y<-object$frame[,1]
 if(1*(object$modelInfo$allForm$ziformula==~0)==1)##no zero-inflation
 {
  pearson<-residuals(object,type="pearson")
 }
 else if(1*(object$modelInfo$allForm$ziformula==~0)==0)##zero-inflation
 {
    if(family[1]=="poisson")
    {
      mu_hat<-mu*(1-p0)
      var_hat<-mu_hat*(1+p0*mu)
```

```
      pearson<-(y-mu_hat)/sqrt(var_hat)
    }
    else if(family[1]=="nbinom2")
    {
      mu_hat<-mu*(1-p0)
      var_hat<-mu_hat*(1+mu/size)+mu^2/(p0^2+p0)
      pearson<-(y-mu_hat)/sqrt(var_hat)
    }
 }
 shapiro<-shapiro.test(pearson)$p.value
 list(pearson=pearson,shapiro=shapiro)
}

pearsonhurdle<-function(model_count, model_zero, data)
{
  name.y <- names(model_count$frame)[[1]]
  y <- data[,name.y]
  m<-model_zero$modelInfo$nobs
  family<-family(model_count)$family
  mu<-predict(model_count,newdata=data,zitype="conditional")
  size<-sigma(model_count)
  p0<-1-predict(model_zero,newdata = data, type="zprob")
  prob_nb<-size/(size+mu)
  if(family[1]=="truncated_poisson")
  {
    mu_hat<-(1-p0)*mu/(1-exp(-mu))
    var_hat<-mu_hat*(1+mu)-mu_hat^2
    pearson<-(y-mu_hat)/sqrt(var_hat)
  }
  else if(family[1]=="truncated_nbinom2")
  {
    mu_hat<-(1-p0)*mu/(1-prob_nb^size)
    var_hat<-mu_hat*(mu+1+mu/size)-mu_hat^2
    pearson<-(y-mu_hat)/sqrt(var_hat)
  }
  shapiro<-shapiro.test(pearson)$p.value
  list(pearson=pearson,shapiro=shapiro)
}
```

A.2 Two Functions for Generating Datasets from ZMP and ZMNB

```
##variables
simulate_x<-function(n,l1,l2,l3,l4,l5,l6,meanT)
{
  factor1<-as.factor(sample(c(1:l1),n,replace=TRUE))
  factor2<-as.factor(sample(c(1:l2),n,replace=TRUE))
  factor3<-as.factor(sample(c(1:l3),n,replace=TRUE))
  factor4<-as.factor(sample(c(1:l4),n,replace=TRUE))
  factor5<-as.factor(sample(c(1:l5),n,replace=TRUE))
  factor6<-as.factor(sample(c(1:l6),n,replace=TRUE))
  logn<- log (rpois(n, meanT))
```

```
  data.frame(logn,factor1,factor2,factor3,factor4,factor5,factor6)
}
variables<-simulate_x(800,5,5,5,10,10,10,300000)
save(variables,file = sprintf("/home/web340/simulations/variables/variables800/variables800.RData"))

##simulated from zero-modified poisson model
library(actuar)
simulate_one_otu_p<-function(n,variables,l1,l2,l3,l4,l5,l6,
                             sig_factor1,sig_factor2,sig_factor3,sig_factor4,sig_factor5,
                             sig_factor6, betafactor0=1, betafactor0s=-0.2)
{
  y <- integer (n)
  maxsim <- 100
  nsim <- 0
  while (mean (y==0) > 0.8 & nsim < maxsim) {
    betafactor1<-rnorm(l1,sd=sig_factor1)
    betafactor2<-rnorm(l2,sd=sig_factor2)
    betafactor3<-rnorm(l3,sd=sig_factor3)
    betafactor4<-rnorm(l4,sd=sig_factor4)
    betafactor5<-rnorm(l5,sd=sig_factor5)
    betafactor6<-rnorm(l6,sd=sig_factor6)
    mu<-exp(variables[,"logn"]+betafactor0+betafactor1[variables[,"factor1"]]+
              betafactor2[variables[,"factor2"]]+betafactor3[variables[,"factor3"]]+
              betafactor4[variables[,"factor4"]]+betafactor5[variables[,"factor5"]]+
              betafactor6[variables[,"factor6"]])
    betafactor1s<-rnorm(l1,sd=sig_factor1*20)
    betafactor2s<-rnorm(l2,sd=sig_factor2*20)
    betafactor3s<-rnorm(l3,sd=sig_factor3*20)
    betafactor4s<-rnorm(l4,sd=sig_factor4*20)
    betafactor5s<-rnorm(l5,sd=sig_factor5*20)
    betafactor6s<-rnorm(l6,sd=sig_factor6*20)
    mu0<-exp(betafactor0s+betafactor1s[variables[,"factor1"]]+
               betafactor2s[variables[,"factor2"]]+betafactor3s[variables[,"factor3"]]+
               betafactor4s[variables[,"factor4"]]+betafactor5s[variables[,"factor5"]]+
               betafactor6s[variables[,"factor6"]])
    p0<-mu0/(1+mu0)
    y<-rzmpois(n,mu,p0)
    nsim <- nsim + 1
  }
  if (nsim == maxsim ){
    warning("maximum number of simulation reached!")
  }
  y
}
##for replication
source("/home/web340/simulations/hurdlep/one_otu_hurdlep.R")
load ("/home/web340/simulations/variables/variables800/variables800.RData")

nsmp <- nrow (variables)
simdata<-data.frame(matrix(data=NA,nrow=nsmp,ncol=3000))
for(i in 1:3000)
{
  simdata[,i]<-simulate_one_otu_p(nsmp,variables,5,5,5,10,10,10,0.1,0.1,0.1,0.1,0.1,0.1)
  print(i)
```

```
}

simdatatotal<-cbind(variables,simdata)
save(simdatatotal, file = sprintf("/home/web340/simulations/hurdlep/hurdlepouts800/simdatap.RData"))
write.csv(simdatatotal, file = sprintf("/home/web340/simulations/hurdlep/hurdlepouts800/simdatap.csv"))

##simulated from zero-modified negative binomial model
library(actuar)
simulate_one_otu<-function(n,variables,l1,l2,l3,l4,l5,l6,
                           sig_factor1,sig_factor2,sig_factor3,sig_factor4,sig_factor5,
                           sig_factor6, betafactor0=1, betafactor0s=0.2)
{
  y <- integer(n)
  maxsim <- 100
  nsim <- 0
  while (mean (y==0) > 0.8 & nsim < maxsim) {
    betafactor1<-rnorm(l1,sd=sig_factor1)
    betafactor2<-rnorm(l2,sd=sig_factor2)
    betafactor3<-rnorm(l3,sd=sig_factor3)
    betafactor4<-rnorm(l4,sd=sig_factor4)
    betafactor5<-rnorm(l5,sd=sig_factor5)
    betafactor6<-rnorm(l6,sd=sig_factor6)
    mu<-exp(variables[,"logn"]+betafactor0+betafactor1[variables[,"factor1"]]+
              betafactor2[variables[,"factor2"]]+betafactor3[variables[,"factor3"]]+
      betafactor4[variables[,"factor4"]]+betafactor5[variables[,"factor5"]]+
        betafactor6[variables[,"factor6"]])
    betafactor1s<-rnorm(l1,sd=sig_factor1*20)
    betafactor2s<-rnorm(l2,sd=sig_factor2*20)
    betafactor3s<-rnorm(l3,sd=sig_factor3*20)
    betafactor4s<-rnorm(l4,sd=sig_factor4*20)
    betafactor5s<-rnorm(l5,sd=sig_factor5*20)
    betafactor6s<-rnorm(l6,sd=sig_factor6*20)
    mu0<-exp(betafactor0s+betafactor1s[variables[,"factor1"]]+
               betafactor2s[variables[,"factor2"]]+betafactor3s[variables[,"factor3"]]+
      betafactor4s[variables[,"factor4"]]+betafactor5s[variables[,"factor5"]]+
        betafactor6s[variables[,"factor6"]])
    size<-1+runif(1)
    p0<-mu0/(1+mu0)
    prob<-size/(size+mu)
    y<-rzmnbinom(n,size,prob,p0)
    nsim <- nsim + 1
  }
  if (nsim == maxsim ){
    warning("maximum number of simulation reached!")
  }
  y
}
##for replication
source("/home/web340/simulations/hurdlenb/one_otu_hurdlenb.R")
load ("/home/web340/simulations/variables/variables800/variables800.RData")

nsmp <- nrow (variables)
simdata<-data.frame(matrix(data=NA,nrow=nsmp,ncol=3000))
for(i in 1:3000)
```

```
{
simdata[,i]<-simulate_one_otu(nsmp,variables,5,5,5,10,10,10,0.1,0.1,0.1,0.1,0.1,0.1)
}
simdatatotal<-cbind(variables,simdata)
save(simdatatotal, file = sprintf("/home/web340/simulations/hurdlenb/hurdlenbouts800/simdata.RData"))
write.csv(simdatatotal, file = sprintf("/home/web340/simulations/hurdlenb/hurdlenbouts800/simdata.csv"))
```

A.3 R Code for Fitting Models and Conducting Experiments with Simulated Datasets

```
##compare different models generated from zero-modified poisson
library(actuar)
library(glmmTMB)
source ("/home/web340/residuals_functions/rqrhurdle/rqrhurdle.R")
source ("/home/web340/residuals_functions/rqr/rqr.R")
source ("/home/web340/residuals_functions/pearsonhurdle/pearsonhurdle.R")
source ("/home/web340/residuals_functions/pearson/pearson.R")
load ("/home/web340/simulations/variables/variables800/variables800.RData")

if (!exists ("ifold")) ifold <- 1
data <-read.csv(file="/home/web340/simulations/hurdlep/hurdlepouts800/simdatap.csv")
hurdlep <- data.frame( variables,y = as.integer(data[,ifold+7]))
########
###true model hurdlep
hurdle1.r<-glmmTMB(y~offset(logn)+factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=subset(hurdlep,y>0),ziformula =~0,family=list(family="truncated_poisson",link="log"))
hurdle2.r<-glmmTMB((y>0)~factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=hurdlep,ziformula =~0,family=binomial)
###1.wrong model poisson
object1<-glmmTMB(y~offset(logn)+factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=hurdlep,ziformula=~0,family=poisson)
###2.wrong model zero inflated poisson
object2<-glmmTMB(y~offset(logn)+factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=hurdlep, ziformula=~factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),family=poisson)
###3.wrong model hurdle negative binomial
hurdle1.w<-glmmTMB(y~offset(logn)+factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=subset(hurdlep,y>0),ziformula =~0,family=list(family="truncated_nbinom2",link="log"))
hurdle2.w<-glmmTMB((y>0)~factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=hurdlep,ziformula =~0,family=binomial)

###fitted value
mu.r<-predict(hurdle1.r,newdata=hurdlep,zitype="conditional")
mu.w1<-predict(object1,zitype="conditional")
mu.w2<-predict(object2,zitype="conditional")
mu.w3<-predict(hurdle1.w,newdata=hurdlep,zitype="conditional")
#rqr output
outputrqr.r<-rqrhurdle(hurdle1.r,hurdle2.r,hurdlep)
outputrqr.w1<-rqr(object1)
outputrqr.w2<-rqr(object2)
outputrqr.w3<-rqrhurdle(hurdle1.w,hurdle2.w,hurdlep)
#pearson output
```

```
outputpearson.r<-pearsonhurdle(hurdle1.r,hurdle2.r,hurdlep)
outputpearson.w1<-pearson(object1)
outputpearson.w2<-pearson(object2)
outputpearson.w3<-pearsonhurdle(hurdle1.w,hurdle2.w,hurdlep)
#pvalue
pvalue.r<-outputrqr.r$pvalue
pvalue.w1<-outputrqr.w1$pvalue
pvalue.w2<-outputrqr.w2$pvalue
pvalue.w3<-outputrqr.w3$pvalue
#rqr
rqr.r<-outputrqr.r$qnorm
rqr.w1<-outputrqr.w1$qnorm
rqr.w2<-outputrqr.w2$qnorm
rqr.w3<-outputrqr.w3$qnorm
#pearson
pearson.r<-outputpearson.r$pearson
pearson.w1<-outputpearson.w1$pearson
pearson.w2<-outputpearson.w2$pearson
pearson.w3<-outputpearson.w3$pearson
#rqr shapiro
shapirorqr.r<-outputrqr.r$test
shapirorqr.w1<-outputrqr.w1$test
shapirorqr.w2<-outputrqr.w2$test
shapirorqr.w3<-outputrqr.w3$test
#pearson shapiro
shapiropearson.r<-outputpearson.r$shapiro
shapiropearson.w1<-outputpearson.w1$shapiro
shapiropearson.w2<-outputpearson.w2$shapiro
shapiropearson.w3<-outputpearson.w3$shapiro

pdf (sprintf("fithurdlepouts800/plot%d.pdf", ifold))
hist (pvalue.r,xlab="p-value",main="Randomized Quantile")
hist (pvalue.w1,xlab="p-value",main="Randomized Quantile")
hist (pvalue.w2,xlab="p-value",main="Randomized Quantile")
hist (pvalue.w3,xlab="p-value",main="Randomized Quantile")

plot (mu.r,rqr.r,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.w1,rqr.w1,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.w2,rqr.w2,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.w3,rqr.w3,log="x",xlab="Fitted values",ylab="Randomized Quantile")

plot(mu.r,pearson.r,log="x",xlab="Fitted values",ylab="Pearson")
plot(mu.w1,pearson.w1,log="x",xlab="Fitted values",ylab="Pearson")
plot(mu.w2,pearson.w2,log="x",xlab="Fitted values",ylab="Pearson")
plot(mu.w3,pearson.w3,log="x",xlab="Fitted values",ylab="Pearson")

qqnorm(rqr.r,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqr.r,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqr.w1,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqr.w1,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqr.w2,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqr.w2,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqr.w3,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqr.w3,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
```

```
qqnorm(pearson.r,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(pearson.r,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(pearson.w1,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(pearson.w1,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(pearson.w2,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(pearson.w2,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(pearson.w3,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(pearson.w3,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")

cat (shapirorqr.r, file = sprintf("fithurdlepouts800/fithurdlepshapirorqr.r%d.txt", ifold))
cat (shapirorqr.w1, file = sprintf("fithurdlepouts800/fithurdlepshapirorqr.1w%d.txt", ifold))
cat (shapirorqr.w2, file = sprintf("fithurdlepouts800/fithurdlepshapirorqr.2w%d.txt", ifold))
cat (shapirorqr.w3, file = sprintf("fithurdlepouts800/fithurdlepshapirorqr.3w%d.txt", ifold))

cat (shapiropearson.r, file = sprintf("fithurdlepouts800/fithurdlepshapiropearson.r%d.txt", ifold))
cat (shapiropearson.w1, file = sprintf("fithurdlepouts800/fithurdlepshapiropearson.1w%d.txt", ifold))
cat (shapiropearson.w2, file = sprintf("fithurdlepouts800/fithurdlepshapiropearson.2w%d.txt", ifold))
cat (shapiropearson.w3, file = sprintf("fithurdlepouts800/fithurdlepshapiropearson.3w%d.txt", ifold))

dev.off()

##cluster for 3000 times

cluster_collect <- function (item, nfold)
{
  system(sprintf("rm %s.txt", item))
  for (i in 1:nfold) {
    system(sprintf("cat %s%d.txt >> %s.txt", item, i, item))
    cat("\n", file = sprintf("%s.txt", item), append = T)
  }
  read.table(sprintf("%s.txt", item), fill = TRUE, header = F)
}

cluster_collect ("fithurdlepouts800/fithurdlepshapirorqr.r", 3000) -> shw_pvaluesrqr.r
cluster_collect ("fithurdlepouts800/fithurdlepshapirorqr.1w", 3000) -> shw_pvaluesrqr.w1
cluster_collect ("fithurdlepouts800/fithurdlepshapirorqr.2w", 3000) -> shw_pvaluesrqr.w2
cluster_collect ("fithurdlepouts800/fithurdlepshapirorqr.3w", 3000) -> shw_pvaluesrqr.w3

cluster_collect ("fithurdlepouts800/fithurdlepshapiropearson.r", 3000) -> shw_pvaluespearson.r
cluster_collect ("fithurdlepouts800/fithurdlepshapiropearson.1w", 3000) -> shw_pvaluespearson.w1
cluster_collect ("fithurdlepouts800/fithurdlepshapiropearson.2w", 3000) -> shw_pvaluespearson.w2
cluster_collect ("fithurdlepouts800/fithurdlepshapiropearson.3w", 3000) -> shw_pvaluespearson.w3

pdf (sprintf("plot800.pdf"))
hist (shw_pvaluesrqr.r[!is.na(shw_pvaluesrqr.r)],main="Randomized Quantile",xlab="p-value")
hist (shw_pvaluesrqr.w1[!is.na(shw_pvaluesrqr.w1)],main="Randomized Quantile",xlab="p-value")
hist (shw_pvaluesrqr.w2[!is.na(shw_pvaluesrqr.w2)],main="Randomized Quantile",xlab="p-value")
hist (shw_pvaluesrqr.w3[!is.na(shw_pvaluesrqr.w3)],main="Randomized Quantile",xlab="p-value")

hist (shw_pvaluespearson.r[!is.na(shw_pvaluesrqr.r)],main="Pearson",xlab="p-value")
hist (shw_pvaluespearson.w1[!is.na(shw_pvaluesrqr.w1)],main="Pearson",xlab="p-value")
hist (shw_pvaluespearson.w2[!is.na(shw_pvaluesrqr.w2)],main="Pearson",xlab="p-value")
hist (shw_pvaluespearson.w3[!is.na(shw_pvaluesrqr.w3)],main="Pearson",xlab="p-value")
```

```
s_rqr.r<-1-(sum((shw_pvaluesrqr.r > 0.05)*1)/nrow(shw_pvaluesrqr.r))
s_rqr.w1<-1-(sum((shw_pvaluesrqr.w1 > 0.05)*1)/nrow(shw_pvaluesrqr.w1))
s_rqr.w2<-1-(sum((shw_pvaluesrqr.w2 > 0.05)*1)/nrow(shw_pvaluesrqr.w2))
s_rqr.w3<-1-(sum((shw_pvaluesrqr.w3 > 0.05)*1)/nrow(shw_pvaluesrqr.w3))

s_pearson.r<-1-(sum((shw_pvaluespearson.r > 0.05)*1)/nrow(shw_pvaluespearson.r))
s_pearson.w1<-1-(sum((shw_pvaluespearson.w1 > 0.05)*1)/nrow(shw_pvaluespearson.w1))
s_pearson.w2<-1-(sum((shw_pvaluespearson.w2 > 0.05)*1)/nrow(shw_pvaluespearson.w2))
s_pearson.w3<-1-(sum((shw_pvaluespearson.w3 > 0.05)*1)/nrow(shw_pvaluespearson.w3))

cat (c(s_rqr.r,s_rqr.w1,s_rqr.w2,s_rqr.w3), file = sprintf("s_rqr800"))
cat (c(s_pearson.r,s_pearson.w1,s_pearson.w2,s_pearson.w3), file = sprintf("s_pearson800"))

dev.off()

##comparing different models generated from zero-modified negative binomial
library(actuar)
library(glmmTMB)
source ("/home/web340/residuals_functions/rqrhurdle/rqrhurdle.R")
source ("/home/web340/residuals_functions/rqr/rqr.R")
source ("/home/web340/residuals_functions/pearsonhurdle/pearsonhurdle.R")
source ("/home/web340/residuals_functions/pearson/pearson.R")
load ("/home/web340/simulations/variables/variables800/variables800.RData")

if (!exists ("ifold")) ifold <- 1

data <-read.csv(file="/home/web340/simulations/hurdlenb/hurdlenbouts800/simdata.csv")
hurdlenb <- data.frame( variables,y = as.integer(data[,ifold+7]))
########
###true model hurdle negative binomial
hurdle1.r<-glmmTMB(y~offset(logn)+factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=subset(hurdlenb,y>0),ziformula =~0,family=list(family="truncated_nbinom2",link="log"))
hurdle2.r<-glmmTMB((y>0)~factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=hurdlenb,ziformula =~0,family=binomial)
###1.wrong model negative binomial
object1<-glmmTMB(y~offset(logn)+factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=hurdlenb,ziformula=~0,family=nbinom2)
###2.wrong model zero-inflated negative binomial
object2<-glmmTMB(y~offset(logn)+factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=hurdlenb, ziformula=~factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),family=nbinom2)
###3.wrong model hurdle poisson
hurdle1.w<-glmmTMB(y~offset(logn)+factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=subset(hurdlenb,y>0),ziformula =~0,family=list(family="truncated_poisson",link="log"))
hurdle2.w<-glmmTMB((y>0)~factor1+factor2+factor3+(1|factor4)+(1|factor5)+(1|factor6),
data=hurdlenb,ziformula =~0,family=binomial)
#########
#fitted value
mu.r<-predict(hurdle1.r,newdata=hurdlenb,zitype="conditional")
mu.w1<-predict(object1,zitype="conditional")
mu.w2<-predict(object2,zitype="conditional")
mu.w3<-predict(hurdle1.w,newdata=hurdlenb,zitype="conditional")
#rqr output
outputrqr.r<-rqrhurdle(hurdle1.r,hurdle2.r,hurdlenb)
```

```
outputrqr.w1<-rqr(object1)
outputrqr.w2<-rqr(object2)
outputrqr.w3<-rqrhurdle(hurdle1.w,hurdle2.w,hurdlenb)
#pearson output
outputpearson.r<-pearsonhurdle(hurdle1.r,hurdle2.r,hurdlenb)
outputpearson.w1<-pearson(object1)
outputpearson.w2<-pearson(object2)
outputpearson.w3<-pearsonhurdle(hurdle1.w,hurdle2.w,hurdlenb)
#pvalue
pvalue.r<-outputrqr.r$pvalue
pvalue.w1<-outputrqr.w1$pvalue
pvalue.w2<-outputrqr.w2$pvalue
pvalue.w3<-outputrqr.w3$pvalue
#rqr
rqr.r<-outputrqr.r$qnorm
rqr.w1<-outputrqr.w1$qnorm
rqr.w2<-outputrqr.w2$qnorm
rqr.w3<-outputrqr.w3$qnorm
#pearson
pearson.r<-outputpearson.r$pearson
pearson.w1<-outputpearson.w1$pearson
pearson.w2<-outputpearson.w2$pearson
pearson.w3<-outputpearson.w3$pearson
#rqr shapiro
shapirorqr.r<-outputrqr.r$test
shapirorqr.w1<-outputrqr.w1$test
shapirorqr.w2<-outputrqr.w2$test
shapirorqr.w3<-outputrqr.w3$test
#pearson shapiro
shapiropearson.r<-outputpearson.r$shapiro
shapiropearson.w1<-outputpearson.w1$shapiro
shapiropearson.w2<-outputpearson.w2$shapiro
shapiropearson.w3<-outputpearson.w3$shapiro

pdf (sprintf("fithurdlenbouts800/plot%d.pdf", ifold))
hist (pvalue.r,xlab="p-value",main="Randomized Quantile")
hist (pvalue.w1,xlab="p-value",main="Randomized Quantile")
hist (pvalue.w2,xlab="p-value",main="Randomized Quantile")
hist (pvalue.w3,xlab="p-value",main="Randomized Quantile")

plot (mu.r,rqr.r,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.w1,rqr.w1,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.w2,rqr.w2,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.w3,rqr.w3,log="x",xlab="Fitted values",ylab="Randomized Quantile")

plot(mu.r,pearson.r,log="x",xlab="Fitted values",ylab="Pearson")
plot(mu.w1,pearson.w1,log="x",xlab="Fitted values",ylab="Pearson")
plot(mu.w2,pearson.w2,log="x",xlab="Fitted values",ylab="Pearson")
plot(mu.w3,pearson.w3,log="x",xlab="Fitted values",ylab="Pearson")

qqnorm(rqr.r,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqr.r,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqr.w1,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqr.w1,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
```

```
qqnorm(rqr.w2,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqr.w2,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqr.w3,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqr.w3,main="Randomized Quantile",ylab="Sample Quantiles",xlab="Theoretical Quantiles")

qqnorm(pearson.r,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(pearson.r,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(pearson.w1,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(pearson.w1,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(pearson.w2,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(pearson.w2,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(pearson.w3,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(pearson.w3,main="Pearson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")

cat (shapirorqr.r, file = sprintf("fithurdlenbouts800/fithurdlenbshapirorqr.r%d.txt", ifold))
cat (shapirorqr.w1, file = sprintf("fithurdlenbouts800/fithurdlenbshapirorqr.1w%d.txt", ifold))
cat (shapirorqr.w2, file = sprintf("fithurdlenbouts800/fithurdlenbshapirorqr.2w%d.txt", ifold))
cat (shapirorqr.w3, file = sprintf("fithurdlenbouts800/fithurdlenbshapirorqr.3w%d.txt", ifold))

cat (shapiropearson.r, file = sprintf("fithurdlenbouts800/fithurdlenbshapiropearson.r%d.txt", ifold))
cat (shapiropearson.w1, file = sprintf("fithurdlenbouts800/fithurdlenbshapiropearson.1w%d.txt", ifold))
cat (shapiropearson.w2, file = sprintf("fithurdlenbouts800/fithurdlenbshapiropearson.2w%d.txt", ifold))
cat (shapiropearson.w3, file = sprintf("fithurdlenbouts800/fithurdlenbshapiropearson.3w%d.txt", ifold))

dev.off()

##cluster
cluster_collect <- function (item, nfold)
{
  system(sprintf("rm %s.txt", item))
  for (i in 1:nfold) {
    system(sprintf("cat %s%d.txt >> %s.txt", item, i, item))
    cat("\n", file = sprintf("%s.txt", item), append = T)
  }
  read.table(sprintf("%s.txt", item), fill = TRUE, header = F)
}

cluster_collect ("fithurdlenbouts800/fithurdlenbshapirorqr.r", 3000) -> shw_pvaluesrqr.r
cluster_collect ("fithurdlenbouts800/fithurdlenbshapirorqr.1w", 3000) -> shw_pvaluesrqr.w1
cluster_collect ("fithurdlenbouts800/fithurdlenbshapirorqr.2w", 3000) -> shw_pvaluesrqr.w2
cluster_collect ("fithurdlenbouts800/fithurdlenbshapirorqr.3w", 3000) -> shw_pvaluesrqr.w3

cluster_collect ("fithurdlenbouts800/fithurdlenbshapiropearson.r", 3000) -> shw_pvaluespearson.r
cluster_collect ("fithurdlenbouts800/fithurdlenbshapiropearson.1w", 3000) -> shw_pvaluespearson.w1
cluster_collect ("fithurdlenbouts800/fithurdlenbshapiropearson.2w", 3000) -> shw_pvaluespearson.w2
cluster_collect ("fithurdlenbouts800/fithurdlenbshapiropearson.3w", 3000) -> shw_pvaluespearson.w3

pdf (sprintf("plot800.pdf"))
hist (shw_pvaluesrqr.r[!is.na(shw_pvaluesrqr.r)],main="Randomized Quantile",xlab="p-value")
hist (shw_pvaluesrqr.w1[!is.na(shw_pvaluesrqr.w1)],main="Randomized Quantile",xlab="p-value")
hist (shw_pvaluesrqr.w2[!is.na(shw_pvaluesrqr.w2)],main="Randomized Quantile",xlab="p-value")
hist (shw_pvaluesrqr.w3[!is.na(shw_pvaluesrqr.w3)],main="Randomized Quantile",xlab="p-value")

hist (shw_pvaluespearson.r[!is.na(shw_pvaluesrqr.r)],main="Pearson",xlab="p-value")
```

```
hist (shw_pvaluespearson.w1[!is.na(shw_pvaluesrqr.w1)],main="Pearson",xlab="p-value")
hist (shw_pvaluespearson.w2[!is.na(shw_pvaluesrqr.w2)],main="Pearson",xlab="p-value")
hist (shw_pvaluespearson.w3[!is.na(shw_pvaluesrqr.w3)],main="Pearson",xlab="p-value")

s_rqr.r<-1-(sum((shw_pvaluesrqr.r > 0.05)*1)/nrow(shw_pvaluesrqr.r))
s_rqr.w1<-1-(sum((shw_pvaluesrqr.w1 > 0.05)*1)/nrow(shw_pvaluesrqr.w1))
s_rqr.w2<-1-(sum((shw_pvaluesrqr.w2 > 0.05)*1)/nrow(shw_pvaluesrqr.w2))
s_rqr.w3<-1-(sum((shw_pvaluesrqr.w3 > 0.05)*1)/nrow(shw_pvaluesrqr.w3))

s_pearson.r<-1-(sum((shw_pvaluespearson.r > 0.05)*1)/nrow(shw_pvaluespearson.r))
s_pearson.w1<-1-(sum((shw_pvaluespearson.w1 > 0.05)*1)/nrow(shw_pvaluespearson.w1))
s_pearson.w2<-1-(sum((shw_pvaluespearson.w2 > 0.05)*1)/nrow(shw_pvaluespearson.w2))
s_pearson.w3<-1-(sum((shw_pvaluespearson.w3 > 0.05)*1)/nrow(shw_pvaluespearson.w3))

cat (c(s_rqr.r,s_rqr.w1,s_rqr.w2,s_rqr.w3), file = sprintf("s_rqr800"))
cat (c(s_pearson.r,s_pearson.w1,s_pearson.w2,s_pearson.w3), file = sprintf("s_pearson800"))

dev.off()
```

A.4 R Code for Analyzing the Twin Study Human Microbiome Data

```
##real data analysis

library(actuar)
library(glmmTMB)
source ("/home/web340/residuals_functions/rqrhurdle/rqrhurdle.R")
source ("/home/web340/residuals_functions/rqr/rqr.R")
if (!exists ("ifold")) ifold <-1

realdata<-read.csv(file="/home/web340/realdata/Twin_study_OTU_genus.csv")
age<-realdata[,"AGE"]
family<-as.factor(realdata[,"FAMILY"])
ancestry<-as.factor(realdata[,"ANCESTRY"])
obesity<-as.factor(realdata[,"OBESITYCAT"])
logn<-log(realdata[,"total_read"])
otu<-realdata[,ifold+9]
pdf (sprintf("outputs1/plot%d.pdf", ifold))
hist (otu,xlab="value",nclass=30,main="OTU")
n<-nrow(realdata)
ncol(realdata)
for(j in 1:n)
  {if(otu[j]<=10)
      otu[j] <- 0}
roundnum<-function(a)
{if(a<0.001)
  paste0("<","10","^","{",ceiling(log(a,base=10)),"}")
  else
  round(a,3)
 }
###hurdlep
hurdlep <- data.frame(age,family,ancestry,obesity,logn,y = otu)
```

```
hurdlep1<-glmmTMB(y~offset(logn)+ancestry+obesity+(1|age)+(1|family),
data=subset(hurdlep,y>0),ziformula =~0,family=list(family="truncated_poisson",link="log"))
hurdlep2<-glmmTMB((y>0)~offset(logn)+ancestry+obesity+(1|age)+(1|family),
data=hurdlep,ziformula =~0,family=binomial)
###hurdlenb
hurdlenb <- data.frame( age,family,ancestry,obesity,logn,y = otu)
hurdlenb1<-glmmTMB(y~offset(logn)+ancestry+obesity+(1|age)+(1|family),
data=subset(hurdlenb,y>0),ziformula =~0,family=list(family="truncated_nbinom2",link="log"))
hurdlenb2<-glmmTMB((y>0)~offset(logn)+ancestry+obesity+(1|age)+(1|family),
data=hurdlenb,ziformula =~0,family=binomial)
###ZIP
object1<-glmmTMB(y~offset(logn)+ancestry+obesity+(1|age)+(1|family),data=hurdlep,
                 ziformula=~offset(logn)+ancestry+obesity+(1|age)+(1|family),family=poisson)
###ZINB
object2<-glmmTMB(y~offset(logn)+ancestry+obesity+(1|age)+(1|family),data=hurdlenb,
                 ziformula=~offset(logn)+ancestry+obesity+(1|age)+(1|family),family=nbinom2)
###Poisson
object3<-glmmTMB(y~offset(logn)+ancestry+obesity+(1|age)+(1|family),
data=hurdlep,ziformula=~0,family=poisson)
###negative binomial
object4<-glmmTMB(y~offset(logn)+ancestry+obesity+(1|age)+(1|family),
data=hurdlenb,ziformula=~0,family=nbinom2)
#fitted value
mu.hurdlep<-predict(hurdlep1,newdata=hurdlep,zitype="conditional")
mu.hurdlenb<-predict(hurdlenb1,newdata=hurdlenb,zitype="conditional")
mu.ZIP<-predict(object1,zitype="conditional")
mu.ZINB<-predict(object2,zitype="conditional")
mu.poisson<-predict(object3,zitype="conditional")
mu.nb<-predict(object4,zitype="conditional")
#rqr output
outhurdlep<-rqrhurdle(hurdlep1,hurdlep2,hurdlep)
outhurdlenb<-rqrhurdle(hurdlenb1,hurdlenb2,hurdlenb)
outZIP<-rqr(object1)
outZINB<-rqr(object2)
outpoisson<-rqr(object3)
outnb<-rqr(object4)
#pvalue
pvaluehurdlep<-outhurdlep$pvalue
pvaluehurdlenb<-outhurdlenb$pvalue
pvalueZIP<-outZIP$pvalue
pvalueZINB<-outZINB$pvalue
pvaluepoisson<-outpoisson$pvalue
pvaluenb<-outnb$pvalue
#rqr
rqrhurdlep<-outhurdlep$qnorm
rqrhurdlenb<-outhurdlenb$qnorm
rqrZIP<-outZIP$qnorm
rqrZINB<-outZINB$qnorm
rqrpoisson<-outpoisson$qnorm
rqrnb<-outnb$qnorm
#rqr shapiro
shapirohurdlep<-roundnum(mean(replicate(100,rqrhurdle(hurdlep1,hurdlep2,hurdlep)$test)))
shapirohurdlenb<-roundnum(mean(replicate(100,rqrhurdle(hurdlenb1,hurdlenb2,hurdlenb)$test)))
shapiroZIP<-roundnum(mean(replicate(100,rqr(object1)$test)))
```

```
shapiroZINB<-roundnum(mean(replicate(100,rqr(object2)$test)))
shapiropoisson<-roundnum(mean(replicate(100,rqr(object3)$test)))
shapironb<-roundnum(mean(replicate(100,rqr(object4)$test)))

pdf (sprintf("outputs/plot%d.pdf", ifold))
hist (pvaluehurdlep,xlab="p-value",main="pvaluehurdlep")
hist (pvaluehurdlenb,xlab="p-value",main="pvaluehurdlenb")
hist (pvalueZIP,xlab="p-value",main="pvalueZIP")
hist (pvalueZINB,xlab="p-value",main="pvalueZINB")
hist (pvaluepoisson,xlab="p-value",main="pvaluepoisson")
hist (pvaluenb,xlab="p-value",main="pvaluenb")

plot (mu.hurdlep,rqrhurdlep,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.hurdlenb,rqrhurdlenb,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.ZIP,rqrZIP,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.ZINB,rqrZINB,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.poisson,rqrpoisson,log="x",xlab="Fitted values",ylab="Randomized Quantile")
plot (mu.nb,rqrnb,log="x",xlab="Fitted values",ylab="Randomized Quantile")

qqnorm(rqrhurdlep,main="rqrhurdlep",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqrhurdlep,main="rqrhurdlep",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqrhurdlenb,main="rqrhurdlenb",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqrhurdlenb,main="rqrhurdlenb",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqrZIP,main="rqrZIP",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqrZIP,main="rqrZIP",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqrZINB,main="rqrZINB",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqrZINB,main="rqrZINB",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqrpoisson,main="rqrpoisson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqrpoisson,main="rqrpoisson",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqnorm(rqrnb,main="rqrnb",ylab="Sample Quantiles",xlab="Theoretical Quantiles")
qqline(rqrnb,main="rqrnb",ylab="Sample Quantiles",xlab="Theoretical Quantiles")


cat (shapirohurdlep, file = sprintf("outputs/shapirohurdlep%d.txt", ifold))
cat (shapirohurdlenb, file = sprintf("outputs/shapirohurdlenb%d.txt", ifold))
cat (shapiroZIP, file = sprintf("outputs/shapiroZIP%d.txt", ifold))
cat (shapiroZINB, file = sprintf("outputs/shapiroZINB%d.txt", ifold))
cat (shapiropoisson, file = sprintf("outputs/shapiropoisson%d.txt", ifold))
cat (shapironb, file = sprintf("outputs/shapironb%d.txt", ifold))

dev.off()

cluster_collect <- function (item, nfold)
{
  system(sprintf("rm %s.txt", item))
  for (i in 1:nfold) {
    system(sprintf("cat %s%d.txt >> %s.txt", item, i, item))
    cat("\n", file = sprintf("%s.txt", item), append = T)
  }
  read.table(sprintf("%s.txt", item), fill = TRUE, header = F)
}
cluster_collect ("outputs/shapiroZINB", 14) -> shw_pvalues
cluster_collect ("outputs/shapirohurdlenb", 14) -> shw_pvalues1
cluster_collect ("outputs/shapiroZIP", 14) -> shw_pvalues2
```

```
cluster_collect ("outputs/shapirohurdlep", 14) -> shw_pvalues3
cluster_collect ("outputs/shapiropoisson", 14) -> shw_pvalues4
cluster_collect ("outputs/shapironb", 14) -> shw_pvalues5
realpvalue<-data.frame(shw_pvalues1,shw_pvalues,shw_pvalues3,shw_pvalues2,shw_pvalues5,shw_pvalues4)
library(xtable)
xtable(realpvalue)
dev.off()
```