Comparison of Models for Count Data With Overdispersion

Chenyu Lin

04/05/2022

Abstract

Overdispersion is frequently encountered in count data analysis that can affect the inference. It could be caused by misspecification or omission of crucial predictors in the model, the omission of random effect in the model, and the addition of either high outliers or excess zeros in the count data. In this report, we discussed four common models to deal with overdispersion, namely, quasi-Poisson regression model, negative binomial regression model, Poisson generalzied linear mixed model (GLMM), and zero-inflated Poisson GLMM. Then we conducted two simulations to compare these models in different scenarios. In the first scenario, when there exists cluster heterogeneity in the dataset, the Poisson GLMM and negative binomial regression model would be preferred. In the second scenario, when the data has excess zeros and heterogeneity, a zero-inflated based model would be preferred.

Keywords: count data, Poisson, overdispersion, zero-inflation, generalized linear mixed model

1 Introduction

Poisson regression model is often used to analyze count data. In Poisson distribution, the variance is assumed to equal the mean (Agresti, 2014). However, in practice, this assumption usually can not be satisfied (Akkol et al., 2016). The situation that variance is greater than mean under Poisson distribution is referred to as overdispersion, and the situation that variance is less than mean under Poisson distribution is referred to as underdispersion, which might be less frequent (Palmer et al., 2013). If the data is overdispersed, and we failed to take it into account in our regression model, the variability in the data would be greater than the predicted. We would incorrectly under-estimate the variance of parameter, thus result in narrower confidence interval and smaller p-value (Payne et al., 2015). Then we could end up with incorrect inference.

There are several causes of overdispersion, including misspecification or omission of crucial predictors in the model, the omission of random effect in the model, and the addition of either high outliers or excess zeros in the count data (Hardin & Hilbe, 2018). The following report will introduce some popular models that people used to address overdispersion in Poisson distribution, including quasi-Poisson regression mdoel, negativebinomial regression model, Poisson generalized linear mixed model (GLMM), and zero-inflated Poisson GLMM. Then we will apply a simulation study to compare these models when there is random effect in the clustered data, and additionally when there are more zeros than expected in the data.

2 Method

2.1 Quasi-Poisson GLM

Assume we have N independent observations $(y_1,\ y_2, ...,\ y_n)$ following Poisson distribution with parameter μ. In the context of quasi-Poisson generalized lienar model (GLM), we suppose $E(Y_i) = \mu_i$ and $var(Y) = \phi\mu_i$ for some constant $\phi > 1$ to capture the overdispersion. Namely, ϕ is the dispersion parameter. The quasi-Poisson generalized linear regression model is assumed to have the form $log(\mu_i) = \boldsymbol{X\beta}$, where X is the covariate matrix.

To understand the estimation process of the parameters $\boldsymbol{\beta}$ and ϕ, we would go through Agresti's text. Poisson distribution belongs to the exponential family. In general, an exponential family with dispersion parameter has the following probability mass density form

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

where ϕ is the dispersion parameter, and θ_i is the natural parameter. The log-likelihood is

$$L(\boldsymbol{\beta}) = \sum_i L_i = \sum log\ f(y_i; \theta_i, \phi) = \sum_i \frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + \sum_i c(y_i,\ \phi)$$

The equation used to compute the maximum likelihood estimates of $\boldsymbol{\beta}$ is

$$\partial L(\boldsymbol{\beta})/\partial\beta_j = \sum_i \partial L_i/\partial\beta_j = \sum_i \frac{(y_i - \mu_i)x_{ij}}{var(Y_i)}\frac{\partial\mu_i}{\partial\eta_i} = 0,$$

where $\eta_i = \sum_j \beta_j x_{ij}$, and $\mu_i = g^{-1}(\sum_j \beta_j x_{ij})$, g is the link function. When Y_i has distribution in the natural exponential family, the mean-variance relationship $var(Y_i) = v(\mu_i)$ characterized the distribution (Jørgensen, 1987). Then the equation

$$\partial L(\boldsymbol{\beta})/\partial\beta_j = \sum_i \frac{(y_i - \mu_i)x_{ij}}{v(\mu_i)}\frac{\partial\mu_i}{\partial\eta_i} = 0 \tag{1}$$

can be used to compute the estimates of parameters, $\hat{\boldsymbol{\beta}}$, applying the iterative weighted least squares (IWLS) algorithm (McCullagh and Neder, 1999). The Fisher information matrix is

$$\boldsymbol{I} = E(-\frac{\partial^2 L(\boldsymbol{\beta})}{\partial\beta_h\partial\beta_j}) = \sum_i E(-\frac{\partial^2 L_i}{\partial\beta_h\partial\beta_j}) = \sum_i \frac{x_{ih}x_{ij}}{var(Y_i)}(\frac{\partial\mu_i}{\partial\eta_i})^2.$$

Then the Fisher information matrix can be simplified as $\boldsymbol{I} = \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}$, where W is the diagonal matrix with main-diagonal elements

$$w_i = \frac{(\partial\mu_i/\partial\eta_i)^2}{var(Y_i)} = \frac{(\partial g^{-1}(\eta_i)/\partial\eta_i)^2}{var(Y_i)} = \frac{(exp(\eta_i))^2}{var(Y_i)} = \frac{\mu_i^2}{var(Y_i)}. \tag{2}$$

Thus the estimated covariance of $\hat{\boldsymbol{\beta}}$ is

$$c\hat{o}v(\hat{\boldsymbol{\beta}}) = \hat{\boldsymbol{I}}^{-1} = (\boldsymbol{X}^T\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}$$

For quasi-Poisson model, recall that we assume mean-variance relationship has the form $var(Y_i) = v(\mu_i) = \phi\mu_i$. If $\phi = 1$, then the Poisson distribution is assumed, and the estimates result is identical to the maximum likelihood estimation for the Poisson distributed response variable. If $\phi > 1$, then $var(Y_i) > \mu_i$ and the overdispersion is taken into account.

After substituting $v(\mu_i) = \phi\mu_i$ into equation (1) to estimate the parameters $\hat{\boldsymbol{\beta}}$, the constant term ϕ_i drops out. The likelihood equations for quasi-Poisson models are the same as the likelihood equations for Poisson models. Thus the parameter estimates are consistent.

Let the diagonal matrix of Poisson model be $\boldsymbol{W}$ with diagonal elements $w_i = \mu_i$, substituting the mean-variance relationship $var(Y_i) = \mu_i$ into equation (2). Then the diagonal elements of $\hat{\boldsymbol{W}}^*$ for the quasi-Poisson model is $w_i^* = \mu_i/\phi = w_i/\phi$, substituting the mean-variance relationship $var(Y_i) = \phi\mu_i$ into equation (2). So the estimated covariance matrix of quasi-Poisson model is

$$c\hat{o}v(\hat{\boldsymbol{\beta}}) = (\boldsymbol{X}^T\hat{\boldsymbol{W}}^*\boldsymbol{X})^{-1} = \phi(\boldsymbol{X}^T\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}.$$

All elements of the covariance matrix of quasi-Poisson model are ϕ times the covariance matrix elements of Poisson regression model.

The Pearson's χ^2 statistic is defined as $\chi^2 = \sum_i \frac{(y_i-\mu_i)^2}{var(Y_i)} = \sum_i \frac{(y_i-\mu_i)^2}{\phi\mu_i}$. The model is satisfactory when the expected value of Pearson statistic equals $N - p$, where N is the number of observations, and p is the number of predictors. The dispersion parameter ϕ can be estimated as $\hat{\phi} = \chi^2/(N - p)$ (Wedderburn, 1974).

As a result, the quasi-Poisson model's parameter estimation is identical to the estimation of the Poisson regression model. The quasi-Poisson model's standard error estimates are $\sqrt{\hat{\phi}} = \sqrt{X^2/(N-p)}$ times the standard error estimates of Poisson regression model, taking the overdispersion into account. The standard error estimates in quasi-Poisson are more plausible to conduct statistical inference.

2.2 Negative Binomial GLM

A mixture model is another often-used model to deal with overdispersion. We assume that given the mean parameter, Y follows a Poisson distribution. The mean parameter follows a gamma distribution. Marginally, the gamma mixture of Poisson distribution is negative binomial distribution of Y (Ver Boveng, 2007).

Suppose that $Y|\lambda \sim Poisson(\lambda)$, with $E(Y|\lambda) = \lambda$ and $var(Y|\lambda) = \lambda$. $Y|\lambda$ has the probability mass function of the form:

$$P(Y = y|\lambda) = e^{-\lambda}\frac{\lambda^y}{y!}\ ,\ y = 0,\ 1,\ 2, ...$$

$\lambda \sim Gamma(k, \mu)$ with $E(\lambda) = \mu$ and $var(\lambda) = \mu^2/k$. λ has the probability density function of the form:

$$f(\lambda) = \frac{(k/\mu)^k}{\Gamma(k)} \lambda^{k-1} e^{-\frac{k\lambda}{\mu}} \ , \ \lambda \geq 0$$

Then the joint probability density function of Y and λ is

$$P(Y = y|\lambda) f(\lambda) = e^{-\lambda} \frac{\lambda^y}{y!} \frac{(k/\mu)^k}{\Gamma(k)} \lambda^{k-1} e^{-\frac{k\lambda}{\mu}}$$

Integrating the parameter λ over its range, we can obtain the probability mass function of the unconditional distribution of Y:

$$\begin{aligned}
P(Y = y) &= \int_0^\infty P(Y = y|\lambda) f(\lambda) d\lambda \\
&= \int_0^\infty e^{-\lambda} \frac{\lambda^y}{y!} \frac{(k/\mu)^k}{\Gamma(k)} \lambda^{k-1} e^{-\frac{k\lambda}{\mu}} d\lambda \\
&= \int_0^\infty \frac{(k/\mu)^k}{y!\Gamma(k)} \lambda^{y+k-1} e^{-(1+k/\mu)\lambda} d\lambda \\
&= \frac{(k/\mu)^k}{y!\Gamma(k)} \frac{\Gamma(y+k)}{(1+k/\mu)^{y+k}} \int_0^\infty \frac{(1+k/\mu)^{y+k}}{\Gamma(y+k)} \lambda^{y+k-1} e^{-(1+k/\mu)\lambda} d\lambda \\
&= \frac{(k/\mu)^k}{y!\Gamma(k)} \frac{\Gamma(y+k)}{(1+k/\mu)^{y+k}} \\
&= \frac{\Gamma(y+k)}{y!\Gamma(k)} (\frac{k}{\mu+k})^k (\frac{\mu}{\mu+k})^y \ , \ y = 0, \ 1, \ 2, ...
\end{aligned}$$

The above probability mass function is that of gamma distribution. Thus, the unconditional variable Y has a gamma distribution. This distribution can be interpreted as the distribution of the number of failures before the kth success in a series of independent Bernoulli trials with a common probability of success $\pi = k/(k + \mu)$ (Hilbe, 2016). Using the law of conditional mean and variance, the mean and variance of Y are

$$E(Y) = E(E(Y|\lambda)) = E(\lambda) = \mu$$

$$var(Y) = var(E(Y|\lambda)) + E(var(Y|\lambda)) = var(\lambda) + E(\lambda) = \mu + \mu^2/k = \mu + \gamma\mu^2$$

where $\gamma = 1/k$ denotes the dispersion parameter. The greater the value of γ, there is more unobserved heterogeneity in the data. As $\gamma \longrightarrow 0$, $var(Y) \longrightarrow \mu$, and the negative binomial distribution converges to Poisson distribution.

We assume $Y_i \sim NB(\mu_i, \gamma)$ for each observation, and let $g(\boldsymbol{\mu}) = \boldsymbol{X\beta}$, where g is the log link function, $\boldsymbol{X}$ is a matrix of covariates, and $\boldsymbol{\beta}$ is a vector of regression coefficients. We can use the equation (1) to estimate the coefficient parameters $\boldsymbol{\beta}$ using the mean-variance relationship $v(\mu_i) = \mu_i + \gamma\mu_i^2$. After substituting the mean-variance relationship of negative binomial distribution into equation (2), we can get the main-diagonal elements $w_i = \mu_i/(1+\gamma\mu_i)$. Then we can use it to calculate the estimated standard error of $\hat{\boldsymbol{\beta}}$ (Hardin and Hilbe, 2018).

2.3 Poisson GLMM

When we have repeated observations for clusters, the within-cluster observations tend to have correlations. The random effect caused by clusters could be incorporated in the generalized linear mixed model (GLMM). Let y_{ij} denotes the observation j in cluster i. Conditional on the random effect u_i, the response variable Y_{ij} follows $Poisson(\lambda_{ij})$. For the fixed effect $\boldsymbol{x_{ij}}$ and random effect u_i, the Poisson GLMM is given by

$$log(E(Y_{ij}|u_i)) = log(\lambda_{ij}) = \boldsymbol{x_{ij}^T \beta} + u_i$$

where $\boldsymbol{\beta}$ is a vector of coefficients, u_i is assumed to follow normal distribution $N(0, \sigma_u^2)$ (Breslow, 1984; Hinde, 1982). Marginally, the mean and variance of Y_{ij} are:

$$E(Y_{ij}) = E(E(Y_{ij}|u_i)) = E(exp(\boldsymbol{x_{ij}^T \beta} + u_i)) = exp(\boldsymbol{x_{ij}^T \beta} + \sigma_u^2/2) = \lambda_{ij}$$

$$var(Y_{ij}) = E(var(Y_{ij}|u_i)) + var(E(Y_{ij}|u_i)) = E(Y_{ij}) + (E(Y_{ij}))^2(exp(\sigma_u^2) - 1) = \lambda_{ij} + \gamma\lambda_{ij}^2$$

where $\gamma = exp(\sigma_u^2) - 1$ is the dispersion parameter. The parameters $\boldsymbol{\beta}$ and σ_u are estimated using pseudo-likelihood (Fitzmaurice, 2009).

2.4 Zero-Inflated Poisson Model with Random Effect

Let y_{ij} denotes the observation j in cluster i. Suppose the response variable Y_{ij} follows a Zero-inflated Poisson (ZIP) distribution, then Y_{ij} follows point-mass distribution at 0 with probability π_{ij}, and Y_{ij} follows $Poisson(\lambda_{ij})$ (including $y_{ij} = 0$) with probability $1 - \pi_{ij}$ (Hall, 2001). Let $f(k; \lambda_{ij}) = \lambda_{ij}^k \frac{e^{-\lambda_{ij}}}{k!}$ denotes the probability mass distribution of $Poisson(\lambda_{ij})$ with the number of occurrence k. Then the probability mass function of Y_{ij} derived by:

$$\begin{aligned}
P(Y_{ij} = y_{ij}) &= \pi_{ij} 1_{y_{ij}=0} + (1 - \pi_{ij}) f(y_{ij}; \lambda_{ij}) \\
&= \begin{cases} \pi_{ij} + (1 - \pi_{ij})\lambda_{ij}^0 \frac{e^{-\lambda_{ij}}}{0!} & \text{if } y_{ij} = 0 \\ (1 - \pi_{ij})\lambda_{ij}^{y_{ij}} \frac{e^{-\lambda_{ij}}}{y_{ij}!} & \text{if } y_{ij} > 0 \end{cases} \\
&= \begin{cases} \pi_{ij} + (1 - \pi_{ij})e^{-\lambda_{ij}} & \text{if } y_{ij} = 0 \\ (1 - \pi_{ij})\lambda_{ij}^{y_{ij}} e^{-\lambda_{ij}}/y_{ij}! & \text{if } y_{ij} > 0 \end{cases}
\end{aligned}$$

The mean and variance of Y_{ij} are:

$$E(Y_{ij}) = (1 - \pi_{ij})\lambda_{ij}$$

$$var(Y_{ij}) = (1 - \pi_{ij})\lambda_{ij}(1 + \pi_{ij}\lambda_{ij})$$

Since $var(Y_{ij}) > E(Y_{ij})$, if we fail to address zero-inflation and incorrectly assume Y_{ij} follows a Poisson distribution, we would encounter the issue of overdispersion. Then we model the parameter λ_{ij} with covariates $\boldsymbol{x_{ij}}$ and univariate random effect u_i using log link; and model the parameter π_{ij} with covariates $\boldsymbol{z_{ij}}$ and univariate random effect v_i using logit link:

$$log(\lambda_{ij}) = \boldsymbol{x_{ij}^T \beta} + u_i$$

$$logit(\pi_{ij}) = \boldsymbol{z}_{ij}^T\boldsymbol{\gamma} + v_i$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are fixed model parameter. The random effects u_i and v_i are assumed to follow normal distribution $N(0, \sigma_u^2)$ and $N(0, \sigma_v^2)$ independently for cluster i. The parameters $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, σ_u^2 and σ_v^2 are estimated using an expectation-maximum (EM) algorithm and associated REML estimate equations (Wang et al., 2002).

3 Simulation Study

We simulated 1000 datasets each with 10 clusters and 1000 random observations in each cluster. Each dataset includes three independent predictors $x1 \sim Uniform(0, 5)$, $x2 \sim Uniform(5, 10)$, $x3 \sim Uniform(10, 15)$. We also assume the response variable $y_{ij} \sim Poisson(\mu_{ij})$. To create overdispersion relative to Poisson, in each cluster, we add a random effect following the normal distribution, $u_i \sim N(0, i/10), i = 1, 2, ..., 10$, where i is the number of the cluster. Thus, μ_{ij} can be modelled as $log(\mu_{ij}) = \beta_0 + \beta_1 x1_{ij} + \beta_2 x2_{ij} + \beta_3 x3_{ij} + u_i$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3) = (0.5, 1, -0.75, 0.5)$.

Then we consider two scenarios. In the first scenario, there is no zero-inflation. We implement Poisson GLM, quasi-Poisson GLM, negative binomial GLM, and Poisson GLMM to fit the overdispersed data. In the second scenario, we incorporate zero-inflation by mixing distributions. In each cluster, we simulate 1000 observations of ones and zeros, z_{ij}, from binomial distribution. We treat $x1$ as the predictor and use logistic regression to model the probabilities of zero, $logit(P(z_{ij} = 0)) = \gamma_0 + \gamma_1 x1$, where $\boldsymbol{\gamma} = (\gamma_0, \gamma_1) = (0.5, 1)$. We apply the Poisson GLM, quasi-Poisson GLM, negative binomial GLM, Poisson GLMM, and Zero-inflated Poisson GLMM to fit the data with zero-inflation and overdispersion.

We evaluate the model performance based on estimated parameters, bias, standard deviation (SD), mean standard error (Mean SE), root mean squared error (RMSE), and coverage probabilities.

3.1 Results

The results from the two scenarios corresponding to without and with zero-inflation are presented in Table 1 and Table 2, respectively.

According to Table 1, not surprisingly, the Poisson GLMM performs the best since it is the correct model to recover the true values. Specifically, the GLMM estimators have no bias, low RMSE, and the coverage probabilities are all over 90%. The "naive" Poisson GLM performs the worst as the coverage probabilities are all only around 5%. Though the Mean SE of Poisson GLM estimators are small, it does not mean Poisson GLM is efficient. The model fails to account for the correlation structure in the data and thus underestimates the variance of parameters, resulting relatively high RMSE. The estimation of quasi-Poisson GLM is consistent with the estimation of Poisson GLM. After adjusting the variance of estimators to capture overdispersion, the quasi-Poisson GLM improves a lot in terms of coverage probabilities of the true parameters. The negative-binomial GLM has a better performance than the quasi-Poisson since the coverage probabilities of β_1, β_2, and β_3 are over 80%, though the coverage probability of β_0 is only 20%. The error of estimation are also small, given the small RMSE. Thus, when there is potential overdispersion in the dataset, we could consider selecting negative-binomial GLM and Poisson GLMM as our candidate model. Further, when there exists reasons that the overdispersion is caused by random effects of clusters or longitudinal study, the Poisson GLMM would be our primary choice to fit the data.

Table 2 gives the results of both overdispersion and zero-inflation in the data. After incorporating random effect for each cluster and mixing the point-mass distribution at zero, the zero-inflated Poisson GLMM (ZIP.GLMM) is the correct model that recover the true values. The performance of the other four models is not ideal. Generally, the standard deviation of the estimator are quite high, especially at $\hat{\beta}_0$, indicating that there could be a wide range of estimation. We can also see that the coverage probabilities of β_1 are all 0.00% for the first four models. This could be explained as, the variable $x1$ is a predictor of both the Poisson log regression and the logistic regression for the probability of zero in the data generating process. Without taking account of excess zero, the log regression model would take less weight on the predictor $x1$ since it is related to the excess zero value of the response variable. As a result, the estimation of β_1 is much less than the true value.

4 Discussion

In this report, we provide a comparison of models for dealing with overdispersion caused by random effect and excess zeros. It turns out that when there is random effect in the dataset, the Poisson GLMM outperforms the other models, and the negative binomial model could also be a candidate to apply. When there exists zero-inflation in the dataset, the models without taking account of the zero outliers do not effectively deal with overdispersion. We would recommend use zero-inflated based models in that situation (Lambert, 1992). It also suggests that it is necessary to check if there are issues of overdispersion and zero-inflation in the dataset to determine which model we would use. Further, other model options such as negative binomial GLMM, zero-inflated negative binomial based models, and GEE methods should be considered when the data is overdispersed (Kong et al, 2015).

TABLE 1: Simulation results for overdispersed data with random effects

	Poisson GLM	Quasi-Poisson GLM	Negative-binomial GLM	Poisson GLMM
Mean of $\hat{\beta}_0$ (SD)	0.697(0.221)	0.697 (0.221)	0.706 (0.096)	0.500(0.009)
Mean of $\hat{\beta}_1$ (SD)	1.001(0.020)	1.001 (0.020)	1.000 (0.007)	1.000(0.001)
Mean of $\hat{\beta}_2$ (SD)	-0.751(0.001)	-0.751 (0.008)	-0.750 (0.005)	-0.750 (0.001)
Mean of $\hat{\beta}_3$ (SD)	0.501(0.001)	0.501(0.007)	0.500(0.005)	0.500 (0.001)
Bias of $\hat{\beta}_0$	0.197	0.197	0.206	0.000
Bias of $\hat{\beta}_1$	0.001	0.001	0.000	0.000
Bias of $\hat{\beta}_2$	-0.001	-0.001	0.000	0.000
Bias of $\hat{\beta}_3$	0.001	0.001	0.000	0.000
Mean SE of $\hat{\beta}_0$	0.008	0.111	0.070	0.008
Mean SE of $\hat{\beta}_1$	0.001	0.010	0.005	0.001
Mean SE of $\hat{\beta}_2$	0.001	0.008	0.005	0.001
Mean SE of $\hat{\beta}_3$	0.001	0.007	0.005	0.001
RMSE of $\hat{\beta}_0$	0.297	0.297	0.227	0.009
RMSE of $\hat{\beta}_1$	0.020	0.020	0.007	0.001
RMSE of $\hat{\beta}_2$	0.019	0.019	0.007	0.001
RMSE of $\hat{\beta}_3$	0.017	0.017	0.007	0.001
Coverage of β_0	4.32%	52.4%	22.4%	94.6%
Coverage of β_1	5.84%	68.8%	84.6%	94.2%
Coverage of β_2	5.63 %	65.3 %	83.4%	94.1%
Coverage of β_3	4.13%	58.4 %	84.1%	94.1%

Abbreviations: GLM, generalized linear model; GLMM, generalized linear mixed model; SD, standard deviation; Mean SE, mean standard error; RMSE, root mean squared error.

TABLE 2: Simulation results for overdispersed data with random effects and excess zeros

	Poisson GLM	Quasi-Poisson GLM	Negative-binomial GLM	Poisson GLMM	ZIP.GLMM
Mean of $\hat{\beta}_0$ (SD)	-0.007 (1.450)	-0.007 (1.450)	-0.029 (0.910)	-0.257 (1.471)	0.501 (0.063)
Mean of $\hat{\beta}_1$ (SD)	0.053 (0.123)	0.053 (0.123)	0.057 (0.076)	0.053 (0.121)	1.000 (0.004)
Mean of $\hat{\beta}_2$ (SD)	-0.742 (0.111)	-0.742 (0.111)	-0.751 (0.063)	-0.743 (0.110)	-0.750 (0.005)
Mean of $\hat{\beta}_3$ (SD)	0.491 (0.111)	0.491 (0.111)	0.498 (0.063)	0.492 (0.110)	0.500 (0.004)
Bias of $\hat{\beta}_0$	-0.507	-0.507	-0.529	-0.757	0.001
Bias of $\hat{\beta}_1$	-0.947	-0.947	-0.943	-0.947	0.000
Bias of $\hat{\beta}_2$	0.008	0.008	-0.001	0.007	0.000
Bias of $\hat{\beta}_3$	-0.009	-0.009	-0.002	-0.008	0.000
Mean SE of $\hat{\beta}_0$	0.059	1.058	0.748	0.059	0.063
Mean SE of $\hat{\beta}_1$	0.003	0.060	0.050	0.003	0.004
Mean SE of $\hat{\beta}_2$	0.004	0.079	0.051	0.004	0.005
Mean SE of $\hat{\beta}_3$	0.004	0.068	0.051	0.004	0.004
RMSE of $\hat{\beta}_0$	1.536	1.536	1.052	1.654	0.063
RMSE of $\hat{\beta}_1$	0.955	0.955	0.946	0.955	0.004
RMSE of $\hat{\beta}_2$	0.111	0.111	0.063	0.110	0.005
RMSE of $\hat{\beta}_3$	0.111	0.111	0.063	0.110	0.004
Coverage of β_0	7.22%	81.78%	84.24%	5.28%	94.63%
Coverage of β_1	0.00%	0.00%	0.00%	0.00%	94.5%
Coverage of β_2	7.13%	83.54%	87.41%	7.48%	94.01%
Coverage of β_3	5.37%	78.17%	88.64%	5.37%	95.51%

Abbreviations: ZIP, Zero-Inflated Poisson; GLM, generalized linear model; GLMM, generalized linear mixed model; SD, standard deviation; Mean SE, mean standard error; RMSE, root mean squared error.

References

Agresti, A. (2014). Categorical Data Analysis. Hoboken: Wiley.

Akkol, Suna & Denizhan, E.. (2016). Analysis of overdispersed count data: An application on acar (Acarina) counts. 69. 1091-1100.

Brooks, Mollie & Kristensen, Kasper & van Benthem, Koen & Magnusson, Arni & Berg, C.W. & Nielsen, Anders & Skaug, Hans & Mächler, Martin & Bolker, Benjamin. (2017). Modeling zero-inflated count data with glmmTMB. 10.1101/132753.

Chen, Xue-dong & Shi, Hong-xing & Wang, Xue-ren. (2016). Hierarchical Mixture Models for Zero-inflated Correlated Count Data. Acta Mathematicae Applicatae Sinica, English Series. 32. 373-384. 10.1007/s10255-016-0564-y.

Fitzmaurice, G. M. (2009). Longitudinal data analysis. Boca Raton: CRC Press.

Hall, Daniel. (2001). Zero-Inflated Poisson and Binomial Regression with Random Effects: A Case Study. Biometrics. 56. 1030-9. 10.1111/j.0006-341X.2000.01030.x.

Hardin, J. W., & Hilbe, J. M. (2018). Generalized linear models and extensions. College Station, TX: Stata Press.

Hilbe, J. (2016). Negative binomial regression. Cambridge: Cambridge University Press.

Jørgensen, B.1987. Exponential dispersion models. J.R.Stat.Soc.B 49: 127-162.

Kong, M., & Xu, S.,& Levy, S. M., & Datta, S. (2015). GEE type inference for clustered zero-inflated negative binomial regression with application to dental caries. Computational statistics & data analysis, 85, 54–66. https://doi.org/10.1016/j.csda.2014.11.014

Lambert, Diane. (1992). Zero-Inflated Poisson Regression, With An Application to Defects in Manufacturing. Technometrics. 34. 1-14. 10.1080/00401706.1992.10485228.

McCullagh, P., & Nelder, J. A. (1999). Generalized linear models. Boca Raton Fla.: Chapman and Hall/CRC.

Palmer, Alfonso & Losilla, J.M. & Vives, Jaume & Jiménez, Rafael & Llorens, N.. (2013). "Overdispersion in the Poisson regression model. A comparative simulation study": Correction.. Methodology European Journal of Research Methods for the Behavioral and Social Sciences. 9. 178. 10.1027/1614-2241/a000069.

Payne, Elizabeth & Hardin, James & Egede, Leonard & Ramakrishnan, Viswanathan & Selassie, Anbesaw & Gebregziabher, Mulugeta. (2015). Approaches for dealing with various sources of overdispersion in modeling count data: Scale adjustment versus modeling. Statistical methods in medical research. 26. 10.1177/0962280215588569.

Ver Hoef, Jay & Boveng, Peter. (2007). Quasi-Poisson vs. negative binomial regression: How should we model overdispersed count data?. Ecology. 88. 2766-72. 10.1890/07-0043.1.

Wang, Kui & Yau, Kelvin & Lee, Andy. (2002). A zero-inflated Poisson mixed model to analyze diagnosis related groups with majority of same-day hospital stays. Computer methods and programs in biomedicine. 68. 195-203. 10.1016/S0169-2607(01)00171-7.

Wedderburn, R.W. (1974). Quasi-likelihood functions, generalized linear models, and the Gauss-Newton method. Biometrika, 61, 439-447.

R Core Team (2021). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Appendix

The appendix provides the R-code for simulation

Simulation for overdispersed data with random effects

```
library(parallel)
library(lme4)
library(pscl)
library(MASS)
```

```
#number of cores
ncores <- max(1, detectCores()-1)
#number of clusters
grp <- 10
#specify sample size in each cluster
n <- 1000
#number of simulation rules
m <- 1000

# specify the regression model parameter
beta0 <- 0.5
beta1 <- 1
beta2 <- -0.75
beta3 <- 0.5

#number of simulations per core
numsim <- floor(m/ncores)
#total simulations given cores
mtot <- ncores * numsim


#define function for multithreading
sim <- function(no) {

    from <- seq(1,mtot,by=numsim)[no]
    to <- seq(numsim,mtot,by=numsim)[no]
```

```
#make empty data structures
beta0_est <- matrix(NA, numsim, 4)
beta1_est <- matrix(NA, numsim, 4)
beta2_est <- matrix(NA, numsim, 4)
beta3_est <- matrix(NA, numsim, 4)

beta0var_est <- matrix(NA, numsim, 4)
beta1var_est <- matrix(NA, numsim, 4)
beta2var_est <- matrix(NA, numsim, 4)
beta3var_est <- matrix(NA, numsim, 4)

beta0_coverage <- matrix(NA, numsim, 4)
beta1_coverage <- matrix(NA, numsim, 4)
beta2_coverage <- matrix(NA, numsim, 4)
beta3_coverage <- matrix(NA, numsim, 4)


#main for-loop to handle each simulation
for (j in 1:numsim) {

    iter <- from + (j - 1) #define current iteration
    set.seed(iter) #set new random seed for each iteration


    # Generate dataset for incorporating random effect:

    grpnum <- c()
    x1 <- c()
    x2 <- c()
    x3 <- c()
    u <- c()
    y <- c()
    for (k in 1:grp) {  # for each cluster
    grp <- rep(k, n)   # the cluster number
    x.1 <- runif(n, 0, 5) # generate first covariate
    x.2 <- runif(n, 5, 10) # generate second covariate
    x.3 <- runif(n, 10, 15) # generate third covariate
    u.0 <- rnorm(n, 0, k/10)  # generate the random effect
    y.0 <- rpois(n, lambda = exp(beta0 + beta1 * x.1 +
                                 beta2 * x.2
                                 + beta3 * x.3 + u.0))
    # generate the outcome for each cluster

    grpnum <- c(grpnum, grp)
    x1 <- c(x1, x.1)
```

```
x2 <- c(x2, x.2)
x3 <- c(x3, x.3)
u <- c(u, u.0)
y <- c(y, y.0)
}
dataset1 <- data.frame(
    "Grpnum" = grpnum,
    "x1" = x1,
    "x2" = x2,
    "x3" = x3,
    "u" = u,
    "y" = y)

 # Poisson GLM:
model1 <- glm(y ~ x1 + x2 + x3,
              data = dataset1, family = poisson)

beta0_est[j,1] <- coef(model1)[1]
beta0var_est[j,1] <- vcov(model1)[1,1]
cil <- beta0_est[j,1] + qnorm(0.025) * sqrt(beta0var_est[j,1])
ciu <- beta0_est[j,1] + qnorm(0.975) * sqrt(beta0var_est[j,1])
beta0_coverage[j,1] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,1] <- coef(model1)[2]
beta1var_est[j,1] <- vcov(model1)[2,2]
cil <- beta1_est[j,1] + qnorm(0.025) * sqrt(beta1var_est[j,1])
ciu <- beta1_est[j,1] + qnorm(0.975) * sqrt(beta1var_est[j,1])
beta1_coverage[j,1] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,1] <- coef(model1)[3]
beta2var_est[j,1] <- vcov(model1)[3,3]
cil <- beta2_est[j,1] + qnorm(0.025) * sqrt(beta2var_est[j,1])
ciu <- beta2_est[j,1] + qnorm(0.975) * sqrt(beta2var_est[j,1])
beta2_coverage[j,1] <- (beta2 >= cil) & (beta2 <= ciu)

beta3_est[j,1] <- coef(model1)[4]
beta3var_est[j,1] <- vcov(model1)[4,4]
cil <- beta3_est[j,1] + qnorm(0.025) * sqrt(beta3var_est[j,1])
ciu <- beta3_est[j,1] + qnorm(0.975) * sqrt(beta3var_est[j,1])
beta3_coverage[j,1] <- (beta3 >= cil) & (beta3 <= ciu)



# Quasi-Poisson GLM
model2 <- glm(y ~ x1 + x2 + x3,
              data = dataset1, family = quasipoisson)

beta0_est[j,2] <- coef(model2)[1]
beta0var_est[j,2] <- vcov(model2)[1,1]
cil <- beta0_est[j,2] + qnorm(0.025) * sqrt(beta0var_est[j,2])
ciu <- beta0_est[j,2] + qnorm(0.975) * sqrt(beta0var_est[j,2])
beta0_coverage[j,2] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,2] <- coef(model2)[2]
beta1var_est[j,2] <- vcov(model2)[2,2]
cil <- beta1_est[j,2] + qnorm(0.025) * sqrt(beta1var_est[j,2])
ciu <- beta1_est[j,2] + qnorm(0.975) * sqrt(beta1var_est[j,2])
beta1_coverage[j,2] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,2] <- coef(model2)[3]
beta2var_est[j,2] <- vcov(model2)[3,3]
cil <- beta2_est[j,2] + qnorm(0.025) * sqrt(beta2var_est[j,2])
ciu <- beta2_est[j,2] + qnorm(0.975) * sqrt(beta2var_est[j,2])
beta2_coverage[j,2] <- (beta2 >= cil) & (beta2 <= ciu)

beta3_est[j,2] <- coef(model2)[4]
beta3var_est[j,2] <- vcov(model2)[4,4]
cil <- beta3_est[j,2] + qnorm(0.025) * sqrt(beta3var_est[j,2])
ciu <- beta3_est[j,2] + qnorm(0.975) * sqrt(beta3var_est[j,2])
beta3_coverage[j,2] <- (beta3 >= cil) & (beta3 <= ciu)


## Negative binomial GLM
model3 <- glm.nb(y ~ x1 + x2 + x3, data = dataset1)

beta0_est[j,3] <- coef(model3)[1]
beta0var_est[j,3] <- vcov(model3)[1,1]
cil <- beta0_est[j,3] + qnorm(0.025) * sqrt(beta0var_est[j,3])
ciu <- beta0_est[j,3] + qnorm(0.975) * sqrt(beta0var_est[j,3])
beta0_coverage[j,3] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,3] <- coef(model3)[2]
beta1var_est[j,3] <- vcov(model3)[2,2]
cil <- beta1_est[j,3] + qnorm(0.025) * sqrt(beta1var_est[j,3])
ciu <- beta1_est[j,3] + qnorm(0.975) * sqrt(beta1var_est[j,3])
beta1_coverage[j,3] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,3] <- coef(model3)[3]
beta2var_est[j,3] <- vcov(model3)[3,3]
cil <- beta2_est[j,3] + qnorm(0.025) * sqrt(beta2var_est[j,3])
```

```
        ciu <- beta2_est[j,3] + qnorm(0.975) * sqrt(beta2var_est[j,3])
        beta2_coverage[j,3] <- (beta2 >= cil) & (beta2 <= ciu)

        beta3_est[j,3] <- coef(model3)[4]
        beta3var_est[j,3] <- vcov(model3)[4,4]
        cil <- beta3_est[j,3] + qnorm(0.025) * sqrt(beta3var_est[j,3])
        ciu <- beta3_est[j,3] + qnorm(0.975) * sqrt(beta3var_est[j,3])
        beta3_coverage[j,3] <- (beta3 >= cil) & (beta3 <= ciu)


        # Poisson GLMM
        model4 <- glmer(y~ x1 + x2 + x3 +(-1 + u|Grpnum),
                        data = dataset1, family= poisson)
        model4_coef <- coef(model4)$Grpnum

        beta0_est[j,4] <- model4_coef$`(Intercept)`[1]
        beta0var_est[j,4] <- vcov(model4)[1,1]
        cil <- beta0_est[j,4] + qnorm(0.025) * sqrt(beta0var_est[j,4])
        ciu <- beta0_est[j,4] + qnorm(0.975) * sqrt(beta0var_est[j,4])
        beta0_coverage[j,4] <- (beta0 >= cil) & (beta0 <= ciu)

        beta1_est[j,4] <- model4_coef$x1[1]
        beta1var_est[j,4] <- vcov(model4)[2,2]
        cil <- beta1_est[j,4] + qnorm(0.025) * sqrt(beta1var_est[j,4])
        ciu <- beta1_est[j,4] + qnorm(0.975) * sqrt(beta1var_est[j,4])
        beta1_coverage[j,4] <- (beta1 >= cil) & (beta1 <= ciu)

        beta2_est[j,4] <- model4_coef$x2[1]
        beta2var_est[j,4] <- vcov(model4)[3,3]
        cil <- beta2_est[j,4] + qnorm(0.025) * sqrt(beta2var_est[j,4])
        ciu <- beta2_est[j,4] + qnorm(0.975) * sqrt(beta2var_est[j,4])
        beta2_coverage[j,4] <- (beta2 >= cil) & (beta2 <= ciu)

        beta3_est[j,4] <- model4_coef$x3[1]
        beta3var_est[j,4] <- vcov(model4)[4,4]
        cil <- beta3_est[j,4] + qnorm(0.025) * sqrt(beta3var_est[j,4])
        ciu <- beta3_est[j,4] + qnorm(0.975) * sqrt(beta3var_est[j,4])
        beta3_coverage[j,4] <- (beta3 >= cil) & (beta3 <= ciu)



}

save(beta0_est, file=file.path(outpath, paste0('beta0_est', no)))
save(beta1_est, file=file.path(outpath, paste0('beta1_est', no)))
```

```
    save(beta2_est, file=file.path(outpath, paste0('beta2_est', no)))
    save(beta3_est, file=file.path(outpath, paste0('beta3_est', no)))

    save(beta0var_est,
         file=file.path(outpath, paste0('beta0var_est', no)))
    save(beta1var_est,
         file=file.path(outpath, paste0('beta1var_est', no)))
    save(beta2var_est,
         file=file.path(outpath, paste0('beta2var_est', no)))
    save(beta3var_est,
         file=file.path(outpath, paste0('beta3var_est', no)))

    save(beta0_coverage,
         file=file.path(outpath, paste0('beta0_coverage', no)))
    save(beta1_coverage,
         file=file.path(outpath, paste0('beta1_coverage', no)))
    save(beta2_coverage,
         file=file.path(outpath, paste0('beta2_coverage', no)))
    save(beta3_coverage,
         file=file.path(outpath, paste0('beta3_coverage', no)))
    return(NULL)
}



#Run the simulation
system.time(
  mclapply(1:ncores, sim, mc.cores=ncores, mc.silent=FALSE))

# Combine the results:
beta0_est_all <- NULL
beta1_est_all <- NULL
beta2_est_all <- NULL
beta3_est_all <- NULL

beta0var_est_all <- NULL
beta1var_est_all <- NULL
beta2var_est_all <- NULL
beta3var_est_all <- NULL

beta0_coverage_all <- NULL
beta1_coverage_all <- NULL
beta2_coverage_all <- NULL
beta3_coverage_all <- NULL
```

```
for (i in 1:ncores) {
    load(file=file.path(outpath, paste0('beta0_est', i)))
    load(file=file.path(outpath, paste0('beta1_est', i)))
    load(file=file.path(outpath, paste0('beta2_est', i)))
    load(file=file.path(outpath, paste0('beta3_est', i)))

    load(file=file.path(outpath, paste0('beta0var_est', i)))
    load(file=file.path(outpath, paste0('beta1var_est', i)))
    load(file=file.path(outpath, paste0('beta2var_est', i)))
    load(file=file.path(outpath, paste0('beta3var_est', i)))

    load(file=file.path(outpath, paste0('beta0_coverage', i)))
    load(file=file.path(outpath, paste0('beta1_coverage', i)))
    load(file=file.path(outpath, paste0('beta2_coverage', i)))
    load(file=file.path(outpath, paste0('beta3_coverage', i)))


    beta0_est_all <- rbind(beta0_est_all, beta0_est)
    beta1_est_all <- rbind(beta1_est_all, beta1_est)
    beta2_est_all <- rbind(beta2_est_all, beta2_est)
    beta3_est_all <- rbind(beta3_est_all, beta3_est)

    beta0var_est_all <- rbind(beta0var_est_all, beta0var_est)
    beta1var_est_all <- rbind(beta1var_est_all, beta1var_est)
    beta2var_est_all <- rbind(beta2var_est_all, beta2var_est)
    beta3var_est_all <- rbind(beta3var_est_all, beta3var_est)

    beta0_coverage_all <- rbind(beta0_coverage_all, beta0_coverage)
    beta1_coverage_all <- rbind(beta1_coverage_all, beta1_coverage)
    beta2_coverage_all <- rbind(beta2_coverage_all, beta2_coverage)
    beta3_coverage_all <- rbind(beta3_coverage_all, beta3_coverage)
}

#make the results table
results <- rbind(colMeans(beta0_est_all), colMeans(beta1_est_all),
                 colMeans(beta2_est_all), colMeans(beta3_est_all),
                 colMeans(beta0_est_all) - beta0,
                 colMeans(beta1_est_all) - beta1,
                 colMeans(beta2_est_all) - beta2,
                 colMeans(beta3_est_all) - beta3,
                 apply(beta0_est_all, 2, sd),
                 apply(beta1_est_all, 2, sd),
                 apply(beta2_est_all, 2, sd),
                 apply(beta3_est_all, 2, sd),
                 sqrt(colMeans(beta0var_est_all)),
                 sqrt(colMeans(beta1var_est_all)),
                 sqrt(colMeans(beta2var_est_all)),
                 sqrt(colMeans(beta3var_est_all)),
                 sqrt(apply(beta0_est_all, 2, var) +
                        (colMeans(beta0_est_all) - beta0)^2),
                 sqrt(apply(beta1_est_all, 2, var) +
                        (colMeans(beta1_est_all) - beta1)^2),
                 sqrt(apply(beta2_est_all, 2, var) +
                        (colMeans(beta2_est_all) - beta2)^2),
                 sqrt(apply(beta3_est_all, 2, var) +
                        (colMeans(beta3_est_all) - beta3)^2),
                 sqrt(colMeans(beta0var_est_all)/m),
                 sqrt(colMeans(beta1var_est_all)/m),
                 sqrt(colMeans(beta2var_est_all)/m),
                 sqrt(colMeans(beta3var_est_all)/m),
                 colMeans(beta0_coverage_all),
                 colMeans(beta1_coverage_all),
                 colMeans(beta2_coverage_all),
                 colMeans(beta3_coverage_all))

colnames(results) <- c('Poisson␣GLM', 'Quasi-Poisson␣GLM',
                       'Negative-Binomial␣GLM', 'Poisson␣GLMM')
rownames(results) <- c('Mean␣of␣beta0', 'Mean␣of␣beta1',
                       'Mean␣of␣beta2', 'Mean␣of␣beta3',
                       'Bias␣of␣beta0', 'Bias␣of␣beta1',
                       'Bias␣of␣beta2', 'Bias␣of␣beta3',
                       'SD␣of␣beta0', 'SD␣of␣beta1',
                       'SD␣of␣beta2', 'SD␣of␣beta3',
                       'Mean␣SE␣of␣beta0', 'Mean␣SE␣of␣beta1',
                       'Mean␣SE␣of␣beta2', 'Mean␣SE␣of␣beta3',
                       'RMSE␣of␣beta0', 'RMSE␣of␣beta1',
                       'RMSE␣of␣beta2', 'RMSE␣of␣beta3',
                       'MCE␣of␣beta0',  'MCE␣of␣beta1',
                       'MCE␣of␣beta2',  'MCE␣of␣beta3',
                       'Coverage␣of␣beta0', 'Coverage␣of␣beta1',
                       'Coverage␣of␣beta2', 'Coverage␣of␣beta3'
                       )
round(results, 5)
```

simulation results for overdispersed data with random effects and excess zeros

```
library(parallel)
```

```r
library(lme4)
library(pscl)
library(MASS)
library(glmmTMB)
```

```r
ncores <- max(1, detectCores()-1)
#number of clusters
grp <- 10
#specify sample size in each cluster
n <- 1000
#specify expected effect size
theta <- 3
#number of simulation rules
m <- 1000
# specify the regression model parameter
beta0 <- 0.5
beta1 <- 1
beta2 <- -0.75
beta3 <- 0.5
gamma0 <- 0.5
gamma1 <- 1

#number of simulations per core
numsim <- floor(m/ncores)
#total simulations given cores
mtot <- ncores * numsim

#define function for multithreading
sim <- function(no) {

    from <- seq(1,mtot,by=numsim)[no]
    to <- seq(numsim,mtot,by=numsim)[no]

    #make empty data structures
    beta0_est <- matrix(NA, numsim, 6)
    beta1_est <- matrix(NA, numsim, 6)
    beta2_est <- matrix(NA, numsim, 6)
    beta3_est <- matrix(NA, numsim, 6)
```

```
beta0var_est <- matrix(NA, numsim, 6)
beta1var_est <- matrix(NA, numsim, 6)
beta2var_est <- matrix(NA, numsim, 6)
beta3var_est <- matrix(NA, numsim, 6)

beta0_coverage <- matrix(NA, numsim, 6)
beta1_coverage <- matrix(NA, numsim, 6)
beta2_coverage <- matrix(NA, numsim, 6)
beta3_coverage <- matrix(NA, numsim, 6)
#main for-loop to handle each simulation
for (j in 1:numsim) {

    iter <- from + (j - 1) #define current iteration
    set.seed(iter) #set new random seed for each iteration


    # Generate dataset for incorporating random effect:
    grpnum <- c()
    x1 <- c()
    x2 <- c()
    x3 <- c()
    u <- c()
    y <- c()
    for (k in 1:grp) {
    grp <- rep(k, n)
    x.1 <- runif(n, 0, 5) # generate first covariate
    x.2 <- runif(n, 5, 10) # generate second covariate
    x.3 <- runif(n, 10, 15) # generate third covariate
    u.0 <- rnorm(n, 0, k/10)  # generate the random effect
    z <- rbinom(n, size = 1,
                prob = (1 -
                1/(1 + exp(-(gamma0 + gamma1 * x.1 )))))
    y.0 <- ifelse(z == 0, 0,
                  rpois(n, lambda =
                  exp(beta0 + beta1 * x.1 +
                  beta2 * x.2 + beta3 * x.3 + u.0) ))
    # generate the outcome for each cluster


    grpnum <- c(grpnum, grp)
    x1 <- c(x1, x.1)
    x2 <- c(x2, x.2)
    x3 <- c(x3, x.3)
    u <- c(u, u.0)
    y <- c(y, y.0)
}
dataset2 <- data.frame(
    "Grpnum" = grpnum,
    "x1" = x1,
    "x2" = x2,
    "x3" = x3,
    "u" = u,
    "y" = y)

# Poisson GLM
model1 <- glm(y ~ x1 + x2 + x3,
              data = dataset2, family = poisson)

beta0_est[j,1] <- coef(model1)[1]
beta0var_est[j,1] <- vcov(model1)[1,1]
cil <- beta0_est[j,1] + qnorm(0.025) * sqrt(beta0var_est[j,1])
ciu <- beta0_est[j,1] + qnorm(0.975) * sqrt(beta0var_est[j,1])
beta0_coverage[j,1] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,1] <- coef(model1)[2]
beta1var_est[j,1] <- vcov(model1)[2,2]
cil <- beta1_est[j,1] + qnorm(0.025) * sqrt(beta1var_est[j,1])
ciu <- beta1_est[j,1] + qnorm(0.975) * sqrt(beta1var_est[j,1])
beta1_coverage[j,1] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,1] <- coef(model1)[3]
beta2var_est[j,1] <- vcov(model1)[3,3]
cil <- beta2_est[j,1] + qnorm(0.025) * sqrt(beta2var_est[j,1])
ciu <- beta2_est[j,1] + qnorm(0.975) * sqrt(beta2var_est[j,1])
beta2_coverage[j,1] <- (beta2 >= cil) & (beta2 <= ciu)

beta3_est[j,1] <- coef(model1)[4]
beta3var_est[j,1] <- vcov(model1)[4,4]
cil <- beta3_est[j,1] + qnorm(0.025) * sqrt(beta3var_est[j,1])
ciu <- beta3_est[j,1] + qnorm(0.975) * sqrt(beta3var_est[j,1])
beta3_coverage[j,1] <- (beta3 >= cil) & (beta3 <= ciu)



# Quasi-Poisson GLM
model2 <- glm(y ~ x1 + x2 + x3,
              data = dataset2, family = quasipoisson)

beta0_est[j,2] <- coef(model2)[1]
beta0var_est[j,2] <- vcov(model2)[1,1]
```

```
cil <- beta0_est[j,2] + qnorm(0.025) * sqrt(beta0var_est[j,2])
ciu <- beta0_est[j,2] + qnorm(0.975) * sqrt(beta0var_est[j,2])
beta0_coverage[j,2] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,2] <- coef(model2)[2]
beta1var_est[j,2] <- vcov(model2)[2,2]
cil <- beta1_est[j,2] + qnorm(0.025) * sqrt(beta1var_est[j,2])
ciu <- beta1_est[j,2] + qnorm(0.975) * sqrt(beta1var_est[j,2])
beta1_coverage[j,2] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,2] <- coef(model2)[3]
beta2var_est[j,2] <- vcov(model2)[3,3]
cil <- beta2_est[j,2] + qnorm(0.025) * sqrt(beta2var_est[j,2])
ciu <- beta2_est[j,2] + qnorm(0.975) * sqrt(beta2var_est[j,2])
beta2_coverage[j,2] <- (beta2 >= cil) & (beta2 <= ciu)

beta3_est[j,2] <- coef(model2)[4]
beta3var_est[j,2] <- vcov(model2)[4,4]
cil <- beta3_est[j,2] + qnorm(0.025) * sqrt(beta3var_est[j,2])
ciu <- beta3_est[j,2] + qnorm(0.975) * sqrt(beta3var_est[j,2])
beta3_coverage[j,2] <- (beta3 >= cil) & (beta3 <= ciu)


# Negative-binomial GLM
model3<- glm.nb(y ~ x1 + x2 + x3, data = dataset2)

beta0_est[j,3] <- coef(model3)[1]
beta0var_est[j,3] <- vcov(model3)[1,1]
cil <- beta0_est[j,3] + qnorm(0.025) * sqrt(beta0var_est[j,3])
ciu <- beta0_est[j,3] + qnorm(0.975) * sqrt(beta0var_est[j,3])
beta0_coverage[j,3] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,3] <- coef(model3)[2]
beta1var_est[j,3] <- vcov(model3)[2,2]
cil <- beta1_est[j,3] + qnorm(0.025) * sqrt(beta1var_est[j,3])
ciu <- beta1_est[j,3] + qnorm(0.975) * sqrt(beta1var_est[j,3])
beta1_coverage[j,3] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,3] <- coef(model3)[3]
beta2var_est[j,3] <- vcov(model3)[3,3]
cil <- beta2_est[j,3] + qnorm(0.025) * sqrt(beta2var_est[j,3])
ciu <- beta2_est[j,3] + qnorm(0.975) * sqrt(beta2var_est[j,3])
beta2_coverage[j,3] <- (beta2 >= cil) & (beta2 <= ciu)

beta3_est[j,3] <- coef(model3)[4]
```

```
beta3var_est[j,3] <- vcov(model3)[4,4]
cil <- beta3_est[j,3] + qnorm(0.025) * sqrt(beta3var_est[j,3])
ciu <- beta3_est[j,3] + qnorm(0.975) * sqrt(beta3var_est[j,3])
beta3_coverage[j,3] <- (beta3 >= cil) & (beta3 <= ciu)



# ZIP GLM
model4 <- zeroinfl(y ~ x1 + x2 + x3 | x1,
                   dist = "poisson", data = dataset2)

beta0_est[j,4] <- coef(model4)[1]
beta0var_est[j,4] <- vcov(model4)[1,1]
cil <- beta0_est[j,4] + qnorm(0.025) * sqrt(beta0var_est[j,4])
ciu <- beta0_est[j,4] + qnorm(0.975) * sqrt(beta0var_est[j,4])
beta0_coverage[j,4] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,4] <- coef(model4)[2]
beta1var_est[j,4] <- vcov(model4)[2,2]
cil <- beta1_est[j,4] + qnorm(0.025) * sqrt(beta1var_est[j,4])
ciu <- beta1_est[j,4] + qnorm(0.975) * sqrt(beta1var_est[j,4])
beta1_coverage[j,4] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,4] <- coef(model4)[3]
beta2var_est[j,4] <- vcov(model4)[3,3]
cil <- beta2_est[j,4] + qnorm(0.025) * sqrt(beta2var_est[j,4])
ciu <- beta2_est[j,4] + qnorm(0.975) * sqrt(beta2var_est[j,4])
beta2_coverage[j,4] <- (beta2 >= cil) & (beta2 <= ciu)

beta3_est[j,4] <- coef(model4)[4]
beta3var_est[j,4] <- vcov(model3)[4,4]
cil <- beta3_est[j,4] + qnorm(0.025) * sqrt(beta3var_est[j,4])
ciu <- beta3_est[j,4] + qnorm(0.975) * sqrt(beta3var_est[j,4])
beta3_coverage[j,4] <- (beta3 >= cil) & (beta3 <= ciu)



# Poisson GLMM
model5<- glmer(y~ x1 + x2 + x3 +(-1 + u|Grpnum),
               data = dataset2, family= poisson)
model5_coef <- coef(model5)$Grpnum

beta0_est[j,5] <- model5_coef$‘(Intercept)‘[1]
beta0var_est[j,5] <- vcov(model5)[1,1]
cil <- beta0_est[j,5] + qnorm(0.025) * sqrt(beta0var_est[j,5])
```

```
ciu <- beta0_est[j,5] + qnorm(0.975) * sqrt(beta0var_est[j,5])
beta0_coverage[j,5] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,5] <- model5_coef$x1[1]
beta1var_est[j,5] <- vcov(model5)[2,2]
cil <- beta1_est[j,5] + qnorm(0.025) * sqrt(beta1var_est[j,5])
ciu <- beta1_est[j,5] + qnorm(0.975) * sqrt(beta1var_est[j,5])
beta1_coverage[j,5] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,5] <- model5_coef$x2[1]
beta2var_est[j,5] <- vcov(model5)[3,3]
cil <- beta2_est[j,5] + qnorm(0.025) * sqrt(beta2var_est[j,5])
ciu <- beta2_est[j,5] + qnorm(0.975) * sqrt(beta2var_est[j,5])
beta2_coverage[j,5] <- (beta2 >= cil) & (beta2 <= ciu)

beta3_est[j,5] <- model5_coef$x3[1]
beta3var_est[j,5] <- vcov(model4)[4,4]
cil <- beta3_est[j,5] + qnorm(0.025) * sqrt(beta3var_est[j,5])
ciu <- beta3_est[j,5] + qnorm(0.975) * sqrt(beta3var_est[j,5])
beta3_coverage[j,5] <- (beta3 >= cil) & (beta3 <= ciu)


# ZIP GLMM
model6<- glmmTMB(y~ x1 + x2 + x3 +(-1 +u|Grpnum),
                 data = dataset2, ziformula = ~x1, family= poisso
model6_coef <- model6$fit$par

beta0_est[j,6] <- model6_coef[[1]]
beta0var_est[j,6] <- vcov(model6)$cond[1]
cil <- beta0_est[j,6] + qnorm(0.025) * sqrt(beta0var_est[j,6])
ciu <- beta0_est[j,6] + qnorm(0.975) * sqrt(beta0var_est[j,6])
beta0_coverage[j,6] <- (beta0 >= cil) & (beta0 <= ciu)

beta1_est[j,6] <- model6_coef[[2]]
beta1var_est[j,6] <- vcov(model6)$cond[6]
cil <- beta1_est[j,6] + qnorm(0.025) * sqrt(beta1var_est[j,6])
ciu <- beta1_est[j,6] + qnorm(0.975) * sqrt(beta1var_est[j,6])
beta1_coverage[j,6] <- (beta1 >= cil) & (beta1 <= ciu)

beta2_est[j,6] <- model6_coef[[3]]
beta2var_est[j,6] <- vcov(model6)$cond[11]
cil <- beta2_est[j,6] + qnorm(0.025) * sqrt(beta2var_est[j,6])
ciu <- beta2_est[j,6] + qnorm(0.975) * sqrt(beta2var_est[j,6])
beta2_coverage[j,6] <- (beta2 >= cil) & (beta2 <= ciu)
```

```
            beta3_est[j,6] <- model6_coef[[4]]
            beta3var_est[j,6] <- vcov(model6)$cond[16]
            cil <- beta3_est[j,6] + qnorm(0.025) * sqrt(beta3var_est[j,6])
            ciu <- beta3_est[j,6] + qnorm(0.975) * sqrt(beta3var_est[j,6])
            beta3_coverage[j,6] <- (beta3 >= cil) & (beta3 <= ciu)

        }

        save(beta0_est, file=file.path(outpath, paste0('beta0_est', no)))
        save(beta1_est, file=file.path(outpath, paste0('beta1_est', no)))
        save(beta2_est, file=file.path(outpath, paste0('beta2_est', no)))
        save(beta3_est, file=file.path(outpath, paste0('beta3_est', no)))

        save(beta0var_est,
             file=file.path(outpath, paste0('beta0var_est', no)))
        save(beta1var_est,
             file=file.path(outpath, paste0('beta1var_est', no)))
        save(beta2var_est,
             file=file.path(outpath, paste0('beta2var_est', no)))
        save(beta3var_est,
             file=file.path(outpath, paste0('beta3var_est', no)))

        save(beta0_coverage,
             file=file.path(outpath, paste0('beta0_coverage', no)))
        save(beta1_coverage,
             file=file.path(outpath, paste0('beta1_coverage', no)))
        save(beta2_coverage,
             file=file.path(outpath, paste0('beta2_coverage', no)))
        save(beta3_coverage,
             file=file.path(outpath, paste0('beta3_coverage', no)))
        return(NULL)
    }

    #Run the simulation
     system.time(
         mclapply(1:ncores, sim, mc.cores=ncores, mc.silent=FALSE)
     )

    # Combine the result
    beta0_est_all <- NULL
    beta1_est_all <- NULL
    beta2_est_all <- NULL
    beta3_est_all <- NULL

    beta0var_est_all <- NULL
```

```
beta1var_est_all <- NULL
beta2var_est_all <- NULL
beta3var_est_all <- NULL

beta0_coverage_all <- NULL
beta1_coverage_all <- NULL
beta2_coverage_all <- NULL
beta3_coverage_all <- NULL


for (i in 1:ncores) {
    load(file=file.path(outpath, paste0('beta0_est', i)))
    load(file=file.path(outpath, paste0('beta1_est', i)))
    load(file=file.path(outpath, paste0('beta2_est', i)))
    load(file=file.path(outpath, paste0('beta3_est', i)))

    load(file=file.path(outpath, paste0('beta0var_est', i)))
    load(file=file.path(outpath, paste0('beta1var_est', i)))
    load(file=file.path(outpath, paste0('beta2var_est', i)))
    load(file=file.path(outpath, paste0('beta3var_est', i)))

    load(file=file.path(outpath, paste0('beta0_coverage', i)))
    load(file=file.path(outpath, paste0('beta1_coverage', i)))
    load(file=file.path(outpath, paste0('beta2_coverage', i)))
    load(file=file.path(outpath, paste0('beta3_coverage', i)))


    beta0_est_all <- rbind(beta0_est_all, beta0_est)
    beta1_est_all <- rbind(beta1_est_all, beta1_est)
    beta2_est_all <- rbind(beta2_est_all, beta2_est)
    beta3_est_all <- rbind(beta3_est_all, beta3_est)

    beta0var_est_all <- rbind(beta0var_est_all, beta0var_est)
    beta1var_est_all <- rbind(beta1var_est_all, beta1var_est)
    beta2var_est_all <- rbind(beta2var_est_all, beta2var_est)
    beta3var_est_all <- rbind(beta3var_est_all, beta3var_est)

    beta0_coverage_all <- rbind(beta0_coverage_all, beta0_coverage)
    beta1_coverage_all <- rbind(beta1_coverage_all, beta1_coverage)
    beta2_coverage_all <- rbind(beta2_coverage_all, beta2_coverage)
    beta3_coverage_all <- rbind(beta3_coverage_all, beta3_coverage)
}

#make the results table
results <- rbind(colMeans(beta0_est_all),
                 colMeans(beta1_est_all),
                 colMeans(beta2_est_all),
                 colMeans(beta3_est_all),
                 colMeans(beta0_est_all) - beta0,
                 colMeans(beta1_est_all) - beta1,
                 colMeans(beta2_est_all) - beta2,
                 colMeans(beta3_est_all) - beta3,
                 apply(beta0_est_all, 2, sd),
                 apply(beta1_est_all, 2, sd),
                 apply(beta2_est_all, 2, sd),
                 apply(beta3_est_all, 2, sd),
                 sqrt(colMeans(beta0var_est_all)),
                 sqrt(colMeans(beta1var_est_all)),
                 sqrt(colMeans(beta2var_est_all)),
                 sqrt(colMeans(beta3var_est_all)),
                 sqrt(apply(beta0_est_all, 2, var)
                      + (colMeans(beta0_est_all) - beta0)^2),
                 sqrt(apply(beta1_est_all, 2, var)
                      + (colMeans(beta1_est_all) - beta1)^2),
                 sqrt(apply(beta2_est_all, 2, var)
                      + (colMeans(beta2_est_all) - beta2)^2),
                 sqrt(apply(beta3_est_all, 2, var)
                      + (colMeans(beta3_est_all) - beta3)^2),
                 colMeans(beta0_coverage_all),
                 colMeans(beta1_coverage_all),
                 colMeans(beta2_coverage_all),
                 colMeans(beta3_coverage_all))

colnames(results) <- c('Poisson␣GLM', 'Quasi-Poisson␣GLM',
                       'Negative-Binomial␣GLM',
                       'Zero-Inflated␣Poisson␣GLM',
                       'Poisson␣GLMM', 'Zero-Inflated␣Poisson␣GLMM')
rownames(results) <- c('Mean␣of␣beta0', 'Mean␣of␣beta1',
                       'Mean␣of␣beta2', 'Mean␣of␣beta3',
                       'Bias␣of␣beta0', 'Bias␣of␣beta1',
                       'Bias␣of␣beta2', 'Bias␣of␣beta3',
                       'SD␣of␣beta0', 'SD␣of␣beta1',
                       'SD␣of␣beta2', 'SD␣of␣beta3',
                       'Mean␣SE␣of␣beta0', 'Mean␣SE␣of␣beta1',
                       'Mean␣SE␣of␣beta2', 'Mean␣SE␣of␣beta3',
                       'RMSE␣of␣beta0', 'RMSE␣of␣beta1',
                       'RMSE␣of␣beta2', 'RMSE␣of␣beta3',
                       'Coverage␣of␣beta0', 'Coverage␣of␣beta1',
                       'Coverage␣of␣beta2', 'Coverage␣of␣beta3'
                       )
```

```
round(results, 5)
```