University of Zurich[UZH]

Author: Anja Fallegger
Matriculation number: 11-926-656

Extension of the Zero-Inflated Hierarchical Models in the EggCounts Package

Master Thesis

Applied Statistics Group at Department of Mathematics
University of Zurich

Mentoring

Supervisor: Prof. Dr. Reinhard Furrer

Submission Date: Sep 21th 2017

Acknowledgement

I would like to express my sincere thanks to my supervisor Prof. Dr. Reinhard Furrer, Department of Mathematics at the University of Zurich, for suggesting this topic to me and for the helpful support whilst writing this thesis. I appreciate the guidance he offered throughout my thesis.
I would also like to thank Craig Wang, Department of Mathematics at the University of Zurich. My thesis is based on his work (Wang et al., 2017), therefore, his advices were very valuable and interesting.
In addition, I would like to express my gratitude to my family and friends who always encouraged me throughout my studies.

Anja Fallegger

Abstract

A gastrointestinal parasitic nematode infection in a livestock can impose huge consequences for the affected animal as well as for its stockman. Therefore such diseases have to be kept under control. This is often done with the help of anthelmintics. But the prevalence of anthelmintic resistance has increased over the last years. A measure to detect such resistances is to observe the reduction in the counts of helminth eggs in the faeces of the animals. These counts are called faecal egg counts. The state-of-the-art statistical method to estimate this reduction is to use zero-inflated Bayesian hierarchical models, which are implemented in the `eggCounts` package in R. Due to the fact that these models are highly sensitive to outliers, in this thesis the existing models are extended regarding outlier detection and the mitigation of its effect on the estimates. To do so the definition of an outlier in the context of faecal egg counts is developed. By extending the existing models in the R package `eggCounts` outliers are taken more into consideration when computing the reduction estimates. With the help of a simulation study it is shown that the extension of the models indeed lower the misclassification rate of datasets in which outliers are present. Hence more accurate results in detecting anthelmintic resistances are obtained.

Contents

1. Introduction

In a livestock of sheep, goats or cattle an infection with gastrointestinal nematodes, a parasitic worm, is quite common. Such infections can have a huge influence on the productivity of these animals. It can induce a performance reduction in milk yield and meat production or it can lead to a smaller body size and less wool growth (van Houtert and Sykes, 1996; Tariq, 2014). Furthermore, the well-being of the animals is in danger (Kaplan, 2004). This can result in great financial damage for the stockman as healthy animals are a very precious renewable resource for humanity (Tariq, 2014; Waller, 1997b). To control these infections, anthelmintics are used. Unfortunately, over the last years the appearance of anthelmintic resistance has increased due to the extensive use of these drugs (Wolstanholme et al., 2004). It is an urgent need to detect these resistances, which exist already almost all-around the world (Kaplan, 2004; Waller, 1997a). There are, for example, anthelmintic resistances for sheep observed in Australia (Roeber et al., 2013), for goats in Northern Italy (Zanzani et al., 2014), for cattle in Central Kenya (Waruiru et al., 2001) and Zimbabwe (Pfukenyi et al., 2007) and for horses in Ontario (Slocombe et al., 2007). To identify these resistances an appropriate definition of a resistance is needed. One possible definition follows from the analysis of the counts of helminth eggs in the faeces of the animals. These counts are called egg counts and they help to analyse the status of parasites in an animal (Tariq, 2014; Blood and Studdert, 2006). A guideline presented by the World Association for the Advancement of Veterinary Parasitology (W.A.A.V.P.) recommends using the following two criteria to identify resistances for sheep and goats:

(i) the percentage reduction in egg count is less than 95%

(ii) the lower limit of the 95% confidence interval is less than 90%.

If both of these conditions are fulfilled a resistance is said to be present. If only one of the two is satisfied a resistance can be suspected (Coles et al., 1992). To get the faecal egg counts (FECs), a modified McMaster counting technique is often used. The general procedure works as follows. A faecal sample of the observed animal is taken and diluted with water. This solution is mixed thoroughly to get a homogeneous distribution of the parasite eggs in the sample. Afterwards, a small sample of the diluted solution is placed onto the chambers of a McMaster slide. Finally, the debris in the solution sinks and the parasite eggs float on the surface such that they can be counted with the help of a microscope. The resulting number is called the raw counts. With the right multiplication factor, called the analytic sensitivity, the number of eggs per gram (EPG) of faeces can be computed out of the raw counts. The analytic sensitivity considers the dilution factor of the sample as well as the volume of the McMaster chamber.

To monitor the two criteria, the faecal egg count reduction test (FECRT) was suggested (Coles et al., 1992). This test compares the pre-treatment and post-treatment FECs. By doing this neither the counting variability resulting from the McMaster counting technique nor the additional information a paired or unpaired design could reveal is taken into account. Therefore, Paul et al. (2014) introduced Bayesian hierarchical models to include this knowledge. In a further step, Wang et al. (2017) extended these models to zero-inflated hierarchical models including unexposed animals in an infected population. In these models the posterior mode for the re-

duction is used as the point estimate. As interval estimate its 95% highest posterior density (HPD) interval is used. These estimates, which are computed with the help of the zero-inflated hierarchical models are strongly influenced by single high values, called outliers. This thesis proposes an extension of these zero-inflated hierarchical models in terms of outlier detection and the handling of the outliers.

The structure of this thesis is as follows. First, a problematic situation concerning the existing zero-inflated Bayesian hierarchical models is demonstrated and the theoretical aspects of these models are explained. Afterwards, some improvements in the model formulation are suggested and a look into the outlier definition are presented. Thereafter, the new model is tested in several simulations and the results are discussed. Finally an outlook on further investigations on this topic is given.

2. Problem Statement

The `eggCounts` package (Wang and Paul, 2017) in R contains hierarchical models to assess anthelmintic efficacy (Paul et al., 2014). Wang et al. (2017) extended these models for zero-inflated faecal egg counts. For the rest of this thesis, these models are called Wang's models. They are explained in detail in Chapter 3. With the help of user feedbacks all around the world, Wang and his team realised that the proposed models do not work well in situations similar to the following. In practise there can be one or more extreme observations. This can occur due to a severely infected animal that experiences a large increase in egg counts while the rest of the cohort experience strong reductions after the treatment. In such a case the overall posterior estimates are strongly influenced by these extreme observations. Consider the following data, which are an example for the raw counts of faecal egg counts resulting from the McMaster counting technique. The animals are measured twice in the same order, once before and once after the treatment. EggsPre stands for the counts before the treatment and eggsPost refers to the counts after the treatment.

$$\text{eggsPre} = (47, 34, 6, 2, 6, 4, 18, 8, 7, 22, 0, 0, 3, 12, 3, 2, 3, 0, 3, 48, 1, 3, 0, 31, 5)$$
$$\text{eggsPost} = (0, 0, 0, 0, 0, 0, 0, 0, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

Considering the FECRT to investigate the reduction, the mean of the reduction and the lower limit of the 95% bootstrap confidence interval (bootCILow95) are computed. Hence the following two values result:

$$\text{mean} = 92.54\%,$$
$$\text{bootCILow95} = 68.25\%.$$

Using the zero-inflated hierarchical models to compute the posterior mode and the lower limit of the 95% HPD interval as estimates for the reduction the following values are obtained:

$$\text{posterior mode} = 92.69\%,$$
$$\text{HPDLow95} = 88.62\%.$$

Making use of the guideline from the W.A.A.P.V. it can be seen that the computed mean as well as the posterior mode are lower than the proposed 95% and also the bootCILow95 and the HPDLow95 are lower than the suggested 90%. With either method both conditions are fulfilled. Therefore a resistance is concluded to be present. By taking a closer look at the egg counts after the treatment (eggsPost) it becomes clear that just one animal does not have a reduction of 100%. This sole animal has a significant influence on the final result, the declared resistance. The purpose of this thesis is to extend the existing models implemented in the `eggCounts` package such that they can handle comparable situations more appropriately. The goal is to define what outlier means in the context of faecal egg counts and to find a way on how to handle these outliers to mitigate their effect on the posterior estimates.

3. Zero-Inflated Bayesian Hierarchical Models

The zero-inflated Bayesian hierarchical models suggested by Wang build the basis of this thesis. Thus this chapter focuses on these models. In order to detect anthelmintic resistance in a flock two different designs can be used. For both designs a zero-inflated Bayesian hierarchical model exists. The models are first discussed in theory and then compared to the models used in practice, integrated in the `eggCounts` package in R.

3.1. Zero-Inflated Bayesian Hierarchical Model for Unpaired Design

First, the model for the unpaired design is considered. In the unpaired design, two distinct groups of animals from the same population are considered. The first group is called the control group with size n_C. This group does not experience any treatment. The second group is the treatment group, consisting of n_T animals. These animals are treated with an anthelmintic. From each animal in both groups a faecal sample is collected and the faecal egg count is determined. The model for the control group animals is

$$\begin{aligned} Y_i^{*C}|Y_i^C &\sim \text{Bin}(Y_i^C, p), \\ Y_i^C|\mu_i^C, \phi &\sim \text{ZIP}(\mu_i^C, \phi), \\ \mu_i^C|\kappa, \mu &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right). \end{aligned} \tag{3.1}$$

For the treatment group, the model is

$$\begin{aligned} Y_i^{*T}|Y_i^T &\sim \text{Bin}(Y_i^T, p), \\ Y_i^T|\mu_i^T, \phi &\sim \text{ZIP}(\delta \cdot \mu_i^T, \phi), \\ \mu_i^T|\kappa, \mu &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right). \end{aligned} \tag{3.2}$$

Referring to the control group, the raw number of eggs in the diluted sample is denoted by Y_i^{*C} and Y_i^C is the true number of eggs per gram of faeces for each animal. For the treatment group, the analogue notations are Y_i^{*T} and Y_i^T. The analytic sensitivity f is supposed to be identical for each animal to make the notation easier to read. Using this parameter, the proportion of the diluted sample placed onto the McMaster slide is determined as $p = 1/f$. The raw number of eggs follows a binomial distribution of size of the true number of EPG and probability p in both the control and the treatment group. Moreover, the true EPG in the control and the treatment group follow a zero-inflated Poisson (ZIP) distribution with zero-inflation parameter ϕ. The mean of the ZIP distribution modelling the egg counts of the control group is denoted by μ_i^C. Further, μ_i^T stands for the mean of the ZIP distribution modelling the treatment group before the usage of the drugs. Due to the treatment, the mean of these counts experience a reduction. The lower mean after the treatment is obtained by multiplying μ_i^T with the factor δ, which represents the proportion of eggs remaining after the treatment. In addition, priors for the parameters μ, κ, ϕ and δ have to be specified.

Using this model described in Eq (3.1) and (3.2), the quantities that are used to identify a declared or suspected resistance are the posterior mode and the corresponding 95% HPD interval. They are used as point and interval estimates.

3.2. Zero-Inflated Bayesian Hierarchical Model for Paired Design

In contrast to the unpaired design, the paired design considers the livestock population as one group. Every animal of the livestock receives the treatment. A faecal sample from each animal is examined twice, meaning the McMaster technique is applied once before the treatment and once after. This yields the following model,

$$\begin{aligned} Y_i^{*C}|Y_i^C &\sim \text{Bin}(Y_i^C, p), \\ Y_i^C|\mu_i^C, \phi &\sim \text{ZIP}(\mu_i^C, \phi), \\ Y_i^{*T}|Y_i^T &\sim \text{Bin}(Y_i^T, p), \\ Y_i^T|\mu_i^C, \phi &\sim \text{ZIP}(\delta \cdot \mu_i^C, \phi), \\ \mu_i^C|\kappa, \mu &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right). \end{aligned} \tag{3.3}$$

Like in the unpaired design the raw egg counts follow a binomial distribution, whereas the true egg counts follow a zero-inflated Poisson distribution. The difference is that in this design the pre-treatment counts Y_i^C and the post-treatment counts Y_i^T depend on the same Poisson mean μ_i^C. This indicates that the counts belong to the same animals. The parameters μ, κ, ϕ and δ have the same meanings and conditions as in the unpaired design.

Analogously to the unpaired design, the posterior mode and the corresponding 95% HPD interval of the reduction parameter serve as point and interval estimate.

3.3. Models in Stan

In the `eggCounts` package the hierarchical models are implemented using the programming language Stan (for more information about Stan check Carpenter et al., 2017 and Stan Development Team, 2017). These models do not exactly correspond to the theoretical ones. One important difference is that they have one level less modelled. Instead of looking at the true number of eggs per gram as well as the raw counts, only the raw counts are considered. This is legitimate as the true number of eggs can easily be computed from the raw counts. With this simplification the computational costs are reduced. Another difference arises from the fact that in Stan not only the design but also the presence of zero-inflation implies different model formulations. As a result in Stan four different models are implemented.

First, the model for the unpaired design without zero-inflation is considered,

$$
\begin{aligned}
Y^*_{b,i} &\sim \text{Poisson}(\lambda_{b,i}),\\
Y^*_{a,i} &\sim \text{Poisson}(\lambda_{a,i}),\\
\lambda_{b,i} &= \frac{\mu_{b,i}}{f_{b,i}},\\
\lambda_{a,i} &= \delta \cdot \frac{\mu_{a,i}}{f_{a,i}},\\
\mu_{b,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right),\\
\mu_{a,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right).
\end{aligned}
\tag{3.4}
$$

This model corresponds to the model formulations Eq (3.1) and (3.2) where the zero-inflation parameter is left out. The McMaster count before the treatment is denoted by $Y^*_{b,i}$. It corresponds to the raw count before the treatment for each animal. In the unpaired design this corresponds to the control group and is denoted by Y^{*C}_i. The McMaster count after the treatment for each animal, denoted by $Y^*_{a,i}$, corresponds to the raw count of the treatment group Y^{*T}_i. They both follow a Poisson distribution. In the theoretical formulation the true EPG are modelled by a Poisson distribution while the raw counts follow a binomial distribution. As the true EPG are not modelled in Stan and the raw counts represent a sample of the true EPG, they follow as well a Poisson distribution. Further, $\lambda_{a,i}$ and $\lambda_{b,i}$ are the means of the Poisson distributions modelling these raw count. They are computed using $\mu_{b,i}$ and $\mu_{a,i}$, the means of the Poisson distributions modelling the true egg count. These means are divided by the individual analytical sensitivity. With this division the transformation from the true egg counts to the raw egg counts occurs. In the `eggCounts` package, priors for μ, κ and δ are defined but they can also be specified by the user.

Using the same notation, the second case, the unpaired design with zero-inflation, is regarded. Equally this model corresponds to the model with Eq (3.1) and (3.2), but this time the zero-inflation parameter ϕ is preserved. The model for this situation is

$$
\begin{aligned}
Y^*_{b,i} &\sim \begin{cases} \phi + (1-\phi)\cdot \text{Poisson}(\lambda_{b,i}), & Y^*_{b,i} = 0,\\ (1-\phi)\cdot \text{Poisson}(\lambda_{b,i}), & \text{else,}\end{cases}\\
Y^*_{a,i} &\sim \begin{cases} \phi + (1-\phi)\cdot \text{Poisson}(\lambda_{a,i}), & Y^*_{a,i} = 0,\\ (1-\phi)\cdot \text{Poisson}(\lambda_{a,i}), & \text{else,}\end{cases}\\
\lambda_{b,i} &= \frac{\mu_{b,i}}{f_{b,i}},\\
\lambda_{a,i} &= \delta \cdot \frac{\mu_{a,i}}{f_{a,i}},\\
\mu_{b,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right),\\
\mu_{a,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right).
\end{aligned}
\tag{3.5}
$$

In contrast to the model without zero-inflation, a case differentiation for the raw counts before the treatment and the raw counts after the treatment is needed. This differentiation has to be done for the zero counts. To do so a parameter ϕ is added, where ϕ is the probability of drawing a zero. Hence $(1-\phi)$ is the probability of drawing from a Poisson(λ_i). Again, priors for μ, κ, δ and also for ϕ are defined but they can also be specified by the user.

Further the paired design is inspected. The following two models, one without and one with zero-inflation, arise from the model definition Eq (3.3). The case without zero-inflation has the following model formulation,

$$\begin{aligned} Y^*_{b,i} &\sim \text{Poisson}(\lambda_{b,i}), \\ Y^*_{a,i} &\sim \text{Poisson}(\lambda_{a,i}), \\ \lambda_{b,i} &= \frac{\mu_{b,i}}{f_{b,i}}, \\ \lambda_{a,i} &= \delta \cdot \frac{\mu_{b,i}}{f_{a,i}}, \\ \mu_{b,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right). \end{aligned} \tag{3.6}$$

The only difference to the unpaired design (Eq (3.4)) is that $\mu_{a,i}$ is equal to $\mu_{b,i}$. This follows from the fact that each animal is considered twice. This dissimilarity between the models for the two designs has as well been observed in the theoretical model formulations (Eq (3.1), (3.2) and Eq (3.3)).

If the zero-inflation is taken into consideration in the paired design, the model is set as follows,

$$\begin{aligned} Y^*_{b,i} &\sim \begin{cases} \phi + (1-\phi) \cdot \text{Poisson}(\lambda_{b,i}), & Y^*_{b,i} = 0, \\ (1-\phi) \cdot \text{Poisson}(\lambda_{b,i}), & \text{else}, \end{cases} \\ Y^*_{a,i} &\sim \begin{cases} \phi + (1-\phi) \cdot \text{Poisson}(\lambda_{a,i}), & Y^*_{a,i} = 0, \\ (1-\phi) \cdot \text{Poisson}(\lambda_{a,i}), & \text{else}, \end{cases} \\ \lambda_{b,i} &= \frac{\mu_{b,i}}{f_{b,i}}, \\ \lambda_{a,i} &= \delta \cdot \frac{\mu_{b,i}}{f_{a,i}}, \\ \mu_{b,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right). \end{aligned} \tag{3.7}$$

This formulation resembles the one from the unpaired design (Eq (3.5)). In fact the only difference is again that $\mu_{a,i}$ is equal to $\mu_{b,i}$.

4. New Models considering Outliers

This chapter deals with the suggestion to improve the existing model respective the handling with outliers. First the basic idea is introduced, followed by the definition of the term outlier. Finally the new model is presented.

4.1. Basic Idea

The basic idea of the model change is to consider the post EPG not as a single zero-inflated Poisson distribution as assumed up to now but to suggest a sum of two weighted Poisson distributions. So in general, instead of

$$Y^*_{a,i} \sim \text{ZIP}(\lambda_{a,i}), \tag{4.1}$$

the proposed distribution is

$$Y^*_{a,i} \sim w_i \cdot \text{ZIP}(\lambda_{a,i}) + (1 - w_i) \cdot \text{Poisson}(\lambda_{a,i,\text{outlier}}). \tag{4.2}$$

The first Poisson distribution is the same zero-inflated Poisson distribution used before, a zero-inflated Poisson distribution with mean λ_a. The second Poisson distribution models the outliers. As an outlier in this setting is not of value zero, this Poisson distribution is not zero-inflated. To determine which value belongs with which weight to which distribution, an accurate definition of outliers is needed. In other studies where outliers appear they have just been deleted, for example in McMahon et al. (2013).

4.2. Outlier

To get a model that is not as sensitive to outliers as Wang's zero-inflated hierarchical model, a rule to identify observations to be outliers is needed. "An outlier is an observation that lies an abnormal distance from other values in a random sample from a population" (Nist/Sematech, 2003). This is a clear and understandable definition of an outlier but it is not explicit at all. The word 'abnormal' has to be described more precisely. The following two sections deal with this challenge and each of them presents one possible way to determine outliers.

4.2.1. Outlier Definition I

The first definition of an outlier that is chosen is based on the egg counts after the treatment. Only these values are considered and depending on their distribution outliers are declared in the following way.

1. The points that are outliers according to the boxplot definition of outliers (Panaretos, 2016) are marked. This concerns all points that are higher or equal than $Q3 + 1.5 \cdot \text{IQR}$, where

$$\begin{aligned} Q1 &= 25\text{th percentiles}, \\ Q3 &= 75\text{th percentiles}, \\ \text{IQR} &= Q3 - Q1. \end{aligned}$$

2. The marked points are left away and the mean of the remaining values is computed.

3. The 95% quantile of a Poisson distribution around the mean computed in the step before is calculated. All points that are higher than this limit are called outlier.

After finding the outliers, all observations are given a weight w. This weight is needed to determine whether an observation belongs rather to the first or the second Poisson distribution (compare Section 4.1). All non-outliers are weighted with $w = 1$, which means that they only belong to the first Poisson distribution. For the outliers the weight is chosen in the following manner. The larger the distance of an outlier to the 95% quantile from step three in the definition above, the smaller the weight. The highest outlier is weighted with 0.01 the others follow proportionally.

4.2.2. Outlier Definition II

The outlier Definition I depends only on the post-treatment values. In the paired design the change from pre-treatment to post-treatment can also be used to define an outlier. For this, the ratio of the egg counts after the treatment (eggsPost) and the egg counts before the treatment (eggsPre) is considered. An observation, in the paired design a specific animal, is declared as outlier if this ratio is ≥ 1. In short, an outlier in the paired design satisfies

$$\frac{\text{eggsPost}}{\text{eggsPre}} \geq 1. \tag{4.3}$$

This occurs if the number of EPG after the treatment are higher or equal than the counts before the treatment. As this definition is dealing with a ratio, cases where the denominator is zero have to be taken into account. In case of $0/0 = \text{NaN}$ the animal does not have any parasite eggs before and after the treatment, so this is defined as not being an outlier. For the situation in which the number of EPG before the treatment is 0 but after treatment is higher, leading to $n/0 = \infty$, the observation is declared to be an outlier as the number of eggs increased after the treatment.

Concerning the weights, again, the non-outliers are weighted with $w = 1$. The outliers in this definition are weighted with

$$w = \left(\frac{\text{eggsPost}}{\text{eggsPre}}\right)^{-1}.$$

To all points with eggsPost/eggsPre $= \infty$ the weight 0.01 is assigned.

4.2.3. Usability of Outlier Definitions

In the previous sections two different definitions of outliers are presented. To use them correctly, their scope of application has to be analysed. The first outlier definition is based on the post-treatment values, which correspond to the treatment group in the unpaired design. Hence this definition can be used in a paired as well as in an unpaired design. In contrast to the first definition, Definition II depends on the relationship between the pre- and post-treatment values. As this connection does not exist for the unpaired design, outlier Definition II can only be applied to datasets with a paired design. Further, the zero-inflation does not lead to any restriction in the usability of the two outlier definitions.

4.3. New models

The extended distribution of the raw counts after the treatment as described in Eq (4.2) leads to changes not only in the theoretical model formulations but also in the models written in Stan.

4.3.1. Adjustment in Theoretical Model Formulation

The important change in the new model formulation happens in modelling the egg counts in the treatment group (unpaired design) and in the post-treatment measurement (paired design) respectively. Instead of using one Poisson distribution the egg counts are modelled as two weighted Poisson distributions (see Section 4.1). This change has to be adapted for the two available designs.

Considering the unpaired case, the following model is suggested. The model for the control group stays unmodified as the change affects only the post-treatment values. Therefore, the model formulation Eq (4.4) is equal to Eq (3.1), namely

$$\begin{aligned} Y_i^{*C}|Y_i^C &\sim \text{Bin}(Y_i^C, p), \\ Y_i^C|\mu_i^C, \phi &\sim \text{ZIP}(\mu_i^C, \phi), \\ \mu_i^C|\kappa, \mu &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right). \end{aligned} \tag{4.4}$$

For the treatment group, the following model is used,

$$\begin{aligned} Y_i^{*T}|Y_i^T &\sim \text{Bin}(Y_i^T, p), \\ Y_i^T|\mu_i^T, \phi &\sim w_i \cdot \text{ZIP}(\delta \cdot \mu_i^T, \phi) + (1 - w_i) \cdot \text{Poisson}(\delta \cdot \mu_i^T \cdot \alpha), \\ \mu_i^T|\kappa, \mu &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right). \end{aligned} \tag{4.5}$$

As a result, not only priors for the parameters μ, κ, ϕ and δ have to be specified but also for the scaling factor α.

The model for the paired design is

$$\begin{aligned}
Y_i^{*C}|Y_i^C &\sim \text{Bin}(Y_i^C, p),\\
Y_i^C|\mu_i^C, \phi &\sim \text{ZIP}(\mu_i^C, \phi),\\
Y_i^{*T}|Y_i^T &\sim \text{Bin}(Y_i^T, p),\\
Y_i^T|\mu_i^C, \phi &\sim w_i \cdot \text{ZIP}(\delta \cdot \mu_i^C, \phi) + (1 - w_i) \cdot \text{Poisson}(\delta \cdot \mu_i^C \cdot \alpha),\\
\mu_i^C|\kappa, \mu &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right).
\end{aligned} \tag{4.6}$$

The true EPG after the treatment are distributed as the sum of two weighted Poisson distributions, one of them with a zero-inflation parameter. Similar to the unpaired design, the priors for μ, κ, δ, ϕ and α need to be specified.

4.3.2. Adjustment in Stan Model Formulation

The model formulation in Stan that is used in the `eggCounts` package has to be adapted to the changes done in the theoretical model formulation. Transferring these changes, the following models arise. In contrast to the two theoretical formulations, in Stan again four models have to be specified, as they differ in situations with or without zero-inflation.

Considering the unpaired design without zero-inflation the counts after the treatment are distributed as follows,

$$\begin{aligned}
Y_{b,i}^* &\sim \text{Poisson}(\lambda_{b,i}),\\
Y_{a,i}^* &\sim w_i \cdot \text{Poisson}(\lambda_{a,i}) + (1 - w_i) \cdot \text{Poisson}(\lambda_{a,i} \cdot \alpha),\\
\lambda_{b,i} &= \frac{\mu_{b,i}}{f_{b,i}},\\
\lambda_{a,i} &= \delta \cdot \frac{\mu_{a,i}}{f_{a,i}},\\
\mu_{b,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right),\\
\mu_{a,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right).
\end{aligned} \tag{4.7}$$

In an unpaired case with zero-inflation the model is determined by the following structure. For the zero counts after the treatment the model stays the same as these counts are never identified as outliers. The modification affects just the non-zero counts. This yields the following model,

$$\begin{aligned}
Y_{b,i}^* &\sim \begin{cases} \phi + (1 - \phi) \cdot \text{Poisson}(\lambda_{b,i}), & Y_{b,i}^* = 0,\\ (1 - \phi) \cdot \text{Poisson}(\lambda_{b,i}), & \text{else,} \end{cases}\\
Y_{a,i}^* &\sim \begin{cases} \phi + (1 - \phi) \cdot \text{Poisson}(\lambda_{a,i}), & Y_{a,i}^* = 0,\\ (1 - \phi) \cdot (w_i \cdot \text{Poisson}(\lambda_{a,i}) + (1 - w_i) \cdot \text{Poisson}(\lambda_{a,i} \cdot \alpha)), & \text{else,} \end{cases}\\
\lambda_{b,i} &= \frac{\mu_{b,i}}{f_{b,i}},\\
\lambda_{a,i} &= \delta \cdot \frac{\mu_{a,i}}{f_{a,i}},\\
\mu_{b,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right),\\
\mu_{a,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right).
\end{aligned} \tag{4.8}$$

In the paired design these changes follow analogously. In the case without zero-inflation the counts after the treatment are similarly distributed as the counts of the treatment group in the unpaired case. The model is

$$\begin{aligned}
Y^*_{b,i} &\sim \text{Poisson}(\lambda_{b,i}),\\
Y^*_{a,i} &\sim w_i \cdot \text{Poisson}(\lambda_{a,i}) + (1 - w_i) \cdot \text{Poisson}(\lambda_{a,i} * \alpha),\\
\lambda_{b,i} &= \frac{\mu_{b,i}}{f_{b,i}},\\
\lambda_{a,i} &= \delta \cdot \frac{\mu_{b,i}}{f_{a,i}},\\
\mu_{b,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right).
\end{aligned} \tag{4.9}$$

Considering the paired design with zero-inflation, similar to the unpaired design, the change affects only the non-zero counts. This yields the following model,

$$\begin{aligned}
Y^*_{b,i} &\sim \begin{cases} \phi + (1-\phi) \cdot \text{Poisson}(\lambda_{b,i}), & Y^*_{b,i} = 0,\\ (1-\phi) \cdot \text{Poisson}(\lambda_{b,i}), & \text{else}, \end{cases}\\
Y^*_{a,i} &\sim \begin{cases} \phi + (1-\phi) \cdot \text{Poisson}(\lambda_{a,i}), & Y^*_{a,i} = 0,\\ (1-\phi) \cdot (w_i \cdot \text{Poisson}(\lambda_{a,i}) + (1-w_i) \cdot \text{Poisson}(\lambda_{a,i} \cdot \alpha)), & \text{else}, \end{cases}\\
\lambda_{b,i} &= \frac{\mu_{b,i}}{f_{b,i}},\\
\lambda_{a,i} &= \delta \cdot \frac{\mu_{b,i}}{f_{a,i}},\\
\mu_{b,i} &\sim \text{Gamma}\left(\kappa, \frac{\kappa}{\mu}\right).
\end{aligned} \tag{4.10}$$

For these four cases the parameters μ, κ, δ, α and sometimes ϕ need priors. Their determination is topic of the next section.

4.3.3. Priors

For the new models in Section 4.3.2, priors for μ, κ, δ, ϕ and α are needed. Although users of these models are allowed to specify the priors for these parameters, default priors need to be defined. Considering the parameters μ, κ, δ and ϕ, a default prior is already given in Wang's model. For the new models, these priors can be inherited. Hence, the following priors are used:

$$\begin{aligned}
\mu &\sim \text{Gamma}(1, 0.001),\\
\kappa &\sim \text{Gamma}(1, 0.7),\\
\delta &\sim \text{Beta}(1, 1),\\
\phi &\sim \text{Beta}(1, 1).
\end{aligned} \tag{4.11}$$

For the scaling factor α, a prior has to be determined. The one chosen in this thesis is

$$\alpha \sim \text{Normal}\left(\frac{wmo}{postmean}, 100\right)\ I(\alpha \geq 1). \tag{4.12}$$

The default prior for α is a truncated normal distribution with a mean equal to $wmo/postmean$ and the variance equal to 100. For the mean, the weighted mean of the outliers (wmo) and the mean of the post-treatment values ($postmean$) are needed. This eventually used prior has been

developed as follows. Considering Eq (4.2) in the new model formulation, the second Poisson distribution with mean $\lambda_{a,i} \cdot \alpha$ models the outliers. Therefore this mean should be around the weighted mean of the outliers (wmo), indicating $\lambda_{a,i} \cdot \alpha \sim wmo$. From this $\alpha \sim wmo/\lambda_{a,i}$ can be derived. This implies a different α for each i. As only one parameter α is desired, a quantity that represents the different $\lambda_{a,i}$ is needed. It was decided to use the mean of these values. This explains why $wmo/postmean$ was taken to be the mean for the prior of α. The variance of 100 developed while doing the simulations. The truncation is a left truncation at one. This results from the fact that the outliers are high values, so on a number line the outliers are the most right values. Due to this fact, the mean of the second Poisson distribution, which is modelling the outliers, has to be higher than the mean of the first Poisson distribution. This leads to the knowledge that the scaling factor $\alpha \geq 1$, as the mean of the second Poisson distribution is a multiple of the mean of the first Poisson distribution.

5. Simulation

In order to analyse the performance of the new model in terms of the faecal egg count reduction estimates, several simulations have been carried out. The results obtained using the new models are compared to the results achieved with Wang's zero-inflated hierarchical models. At the beginning of this chapter the achievements concerning the variation in δ for the different designs with and without zero-inflation are presented. Further, the influence of the change in the parameter ϕ is demonstrated. In addition, the impact of a change in the mean of the pre-treatment count μ is analysed, followed by the effect of a changing overdispersion parameter κ. Last but not least, the consequences of a change in the prior chosen for α is illustrated. The simulations for ϕ, μ, κ and for the α-prior are based only on the paired design with zero-inflation while considering the first outlier definition. The model contains additional parameters, such as the sample size n. For this thesis a fixed parameter $n = 25$ is used, which seems so be a good reflection of the sample sizes in practice. Analysing the effect of varying the sample size is beyond the scope of this work. Moreover, the parameter f is not analysed further, as in practice generally an analytic sensitivity of 50 is used. The priors for the parameters μ, κ, δ and ϕ are not investigated any further in this work as its choice is explained in Wang et al. (2017).

The datasets used in this chapter are generated by the function `simData2s()`, which belongs to the `eggCounts` package in R. For each dataset this function produces two samples, each containing n observations. In a paired design these samples correspond to the FECs before and after the treatment. For the unpaired design one sample represents the treatment group and the other one the control group. Thanks to the options that are available for `simData2s()`, the different cases that exist can be specified. Datasets with the paired as well as with the unpaired design can be generated. The percentage of zero-inflation can be specified via the parameter ϕ. Further, δ can be determined, where $1 - \delta$ is the reduction in mean after the treatment. Finally, the overdispersion parameter κ can be defined. In the simulations in this chapter, for every situation needed, 100 datasets using $n = 25$ are drawn. All simulations are done using `set.seed(246)`.

5.1. Simulation with Varying δ

In this section the performance of the new model is compared to Wang's model for different values of δ. The parameter δ indicates the percentage of eggs remaining after the treatment compared to the counts before the treatment. In the first part of this section, datasets with an extra outlier are considered while in the second part datasets without an extra outlier are used.

The following situations need to be considered: the paired and the unpaired case, either one with and without zero-inflation. For the paired case simulations are once run based on the outlier Definition I and once on the outlier Definition II. In the unpaired case only simulations based on the outlier Definition I can be done, as the second outlier definition can not be applied to

the unpaired design. For these six cases datasets with $\delta = \{0.02,\ 0.04,\ 0.06,\ 0.1,\ 0.15,\ 0.2\}$ are generated. For the situations in which zero-inflation is present, $\phi = 0.9$ is used. This means that 10% of the animals in the population are not infected. Further, as explained, the datasets contain samples with $n = 25$ observations. For all the other parameters the following default values are used: $f = 50$, $\kappa = 0.5$ and $\mu = 500$.

5.1.1. Simulation with Extra Outlier

The goal of this thesis is to improve the zero-inflated hierarchical models in situations, where outliers are present, therefore an obvious outlier is added to all the generated datasets. Then the faecal egg count reduction estimates are computed for all these datasets. Applying the definition of resistance, for each dataset three possible results exist: a declared resistance, a suspected resistance or no resistance is present. For the datasets the δ used for its generation is known, so the obtained result can be compared to the result a perfect model would achieve. If $\delta = 0.02$ or $\delta = 0.04$ then a perfect model should result in no resistances. And if $\delta = \{0.06,\ 0.1,\ 0.15,\ 0.2\}$ a resistance should be declared for each case. The better the results fit the perfect model, the better the model is. To visualize the results for every situation (paired design without zero-inflation, paired design with zero-inflation, unpaired design without zero-inflation, unpaired design with zero-inflation) and for every δ the number of datasets that are misclassified are presented. A perfect model would have value 0 for each δ. The solid line shows the number of misclassified datasets considering a suspected resistance as not being a resistance. In contrast, the dotted line shows the counts of misclassifications in case suspected resistances are considered as resistances too.

First, the paired design without zero-inflation is considered. For this case both outlier definitions can be applied. The results using Definition I are shown in Figure 5.1, the results using Definition II can be seen in Figure 5.2. With either definition the extended model misclassifies less datasets than Wang's model for $\delta = 0.02$ and $\delta = 0.04$. Following, the meaning of this is explained in more details. FECs with a reduction of 98% or 96% from before to after the treatment are considered. To the FECs after the treatment an outlier is added. Considering these datasets the new model identifies less declared and suspected resistances than Wang's model. For $\delta = 0.02$ in case of outlier Definition I Wang's model declares just one dataset out of 100 to have no resistance whereas the new model detects 65 datasets to be resistance-free. For both models, considering $\delta = 0.04$, the number of misclassified datasets is higher than for $\delta = 0.02$. The simulation for outlier Definition II (Figure 5.2) shows a similar pattern. But the improvement achieved applying the new model using outlier Definition II compared to Wang's model is smaller than using outlier Definition I. The performance of the new model for higher δ depends on the chosen outlier definition. Considering outlier Definition I, for $\delta = 0.06$ and $\delta = 0.1$ the new model identifies 27 suspected resistances and 10 datasets without a resistance, respectively 1 suspected resistances and no dataset without a resistance. In contrast, Wang's model performs as a perfect model should, meaning that all datasets are identified as having a declared resistance. For $\delta = 0.15$ and $\delta = 0.2$, both models perform as desired. Considering outlier Definition II, for $\delta = 0.06$ just one dataset is misclassified using the new model. For the considered $\delta > 0.06$ the new model corresponds exactly to Wang's model.

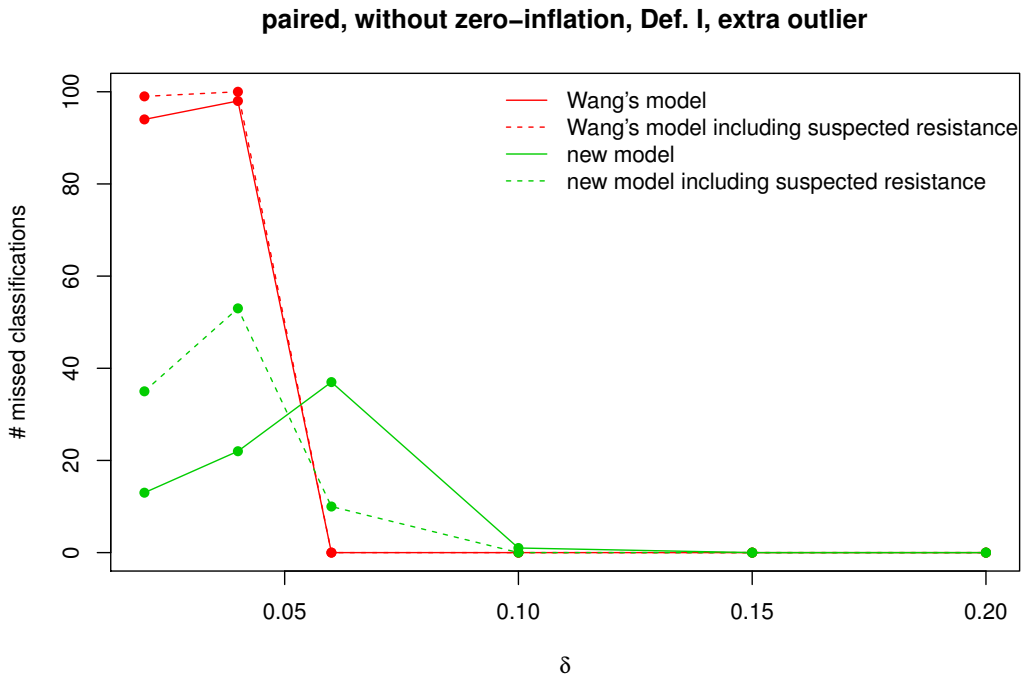

Figure 5.1.: The number of misclassified datasets using datasets where an obvious outlier is added. The data is paired, has no zero-inflation and for outlier detection Definition I is used.

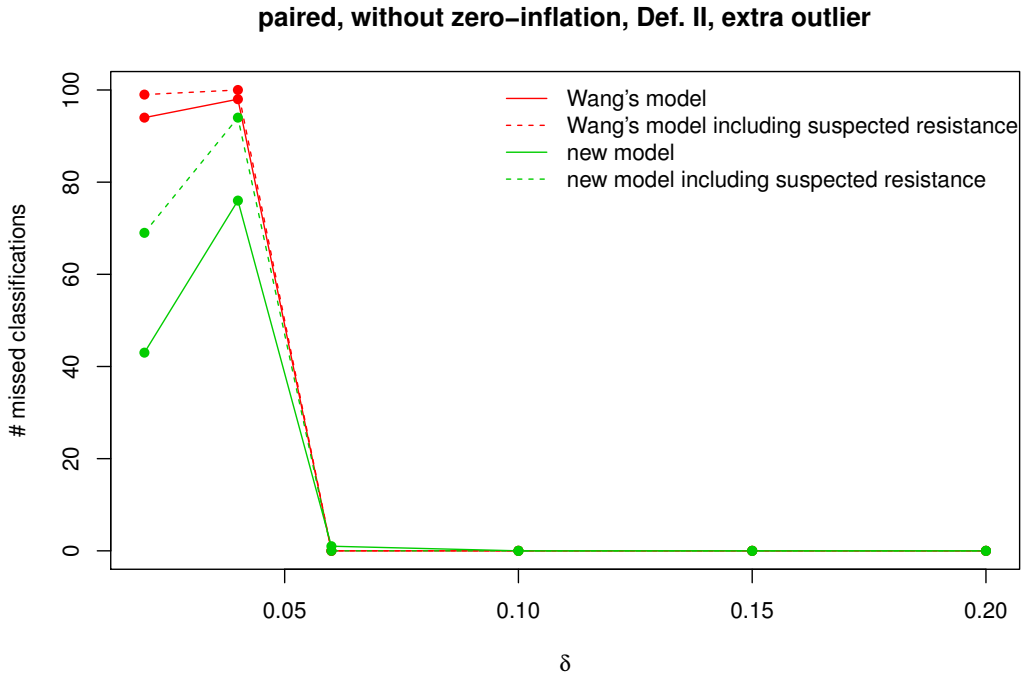

Figure 5.2.: The number of misclassified datasets using datasets where an obvious outlier is added. The data is paired, has no zero-inflation and for outlier detection Definition II is used.

In a next step a simulation with the same setting as before but with a zero-inflation of 10% is shown in Figures 5.3 and 5.4. Similar to the discussed cases, the new model performs better than Wang's model for $\delta = 0.02$ and $\delta = 0.04$. For $\delta = 0.02$, applying outlier Definition I, the new model marks 68 samples out of 100 as having no resistance whereas Wang's model identifies only 11 datasets without a resistance. For $\delta = 0.04$, 52 datasets using the new model respectively 8 datasets using Wang's model are correctly marked. Similar as for the case without zero-inflation, outlier Definition I leads to better results for the smallest two δ compared to outlier Definition II. For the higher δ, especially with outlier Definition I, Wang's model performs better than the new model. Using outlier Definition I, for $\delta = 0.06$ Wang's model identifies 98 declared and 2 suspected resistances out of 100 datasets, the new model only finds 71 declared and 15 suspected resistances. The more δ increases the more the new model approaches Wang's model that performs perfectly for $\delta \geq 0.1$. Checking the results for $\delta \geq 0.06$ using outlier Definition II, the difference between the two models is smaller and the new model approaches Wang's model faster compared to the simulation where outlier Definition I is used.

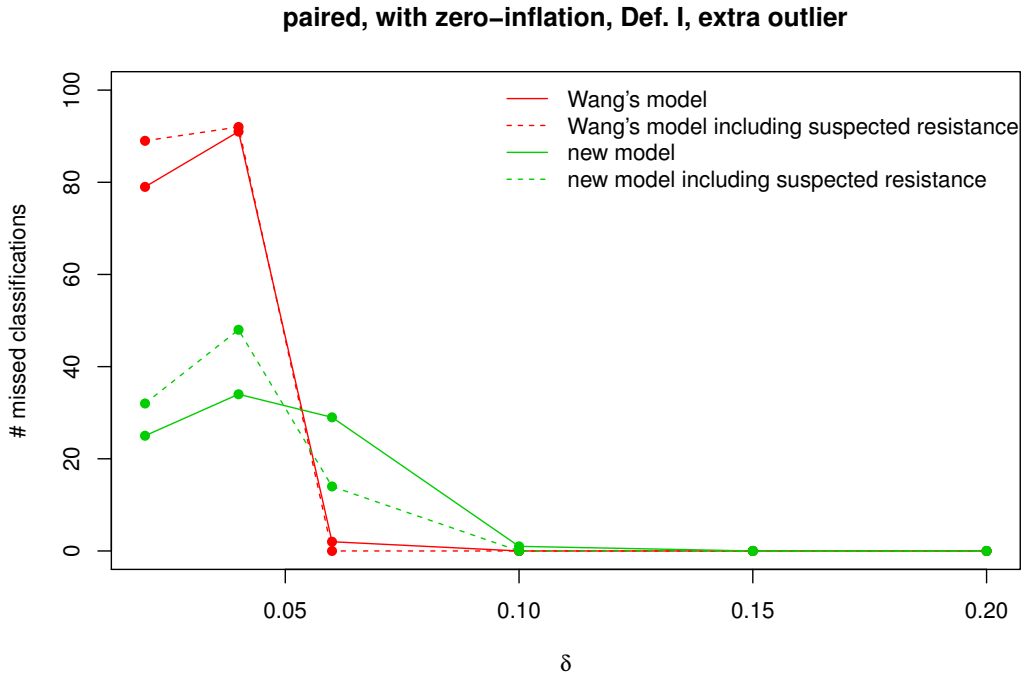

Figure 5.3.: The number of misclassified datasets using datasets where an obvious outlier is added. The data is paired, has zero-inflation and for outlier detection Definition I is used.

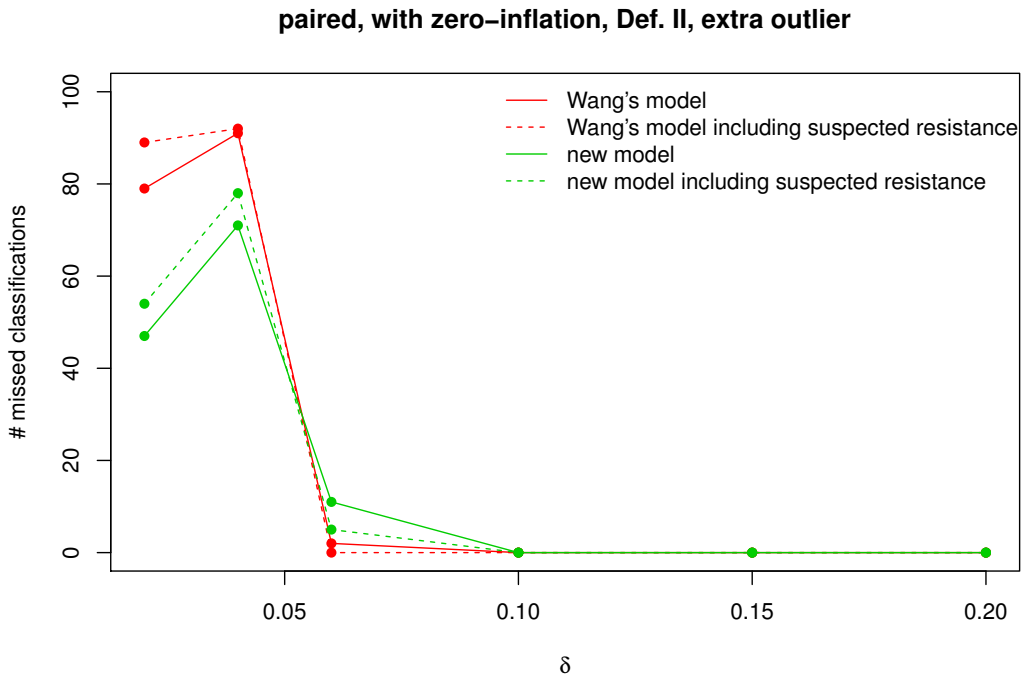

Figure 5.4.: The number of misclassified datasets using datasets where an obvious outlier is added. The data is paired, has zero-inflation and for outlier detection Definition II is used.

Next the unpaired design is analysed (Figures 5.5 and 5.6). Outlier Definition II can not be used in this design, therefore the simulations are based on outlier Definition I. In both simulations, with and without zero-inflation, the proposed model leads to less missed classifications for $\delta = 0.02$ and $\delta = 0.04$ compared to Wang's model. In the case without zero-inflation for $\delta = 0.02$, 96 declared and 4 suspected resistances are identified with Wang's model. With the new model no dataset is identified to have a declared resistance and 38 suspected resistances are determined. The extended model does a better job as its results are closer to the results of a perfect model. The improvement achieved is bigger for $\delta = 0.02$ than for $\delta = 0.04$. In this case the new model identifies 28 declared and 57 suspected resistances, whereas Wang's model identifies all datasets as having a declared resistance. Checking the case with zero-inflation the results show a similar pattern as in the case without zero-inflation. For $\delta = 0.02$ the new model identifies 6 datasets with a declared and 62 datasets with a suspected resistance. Wang's model determines 97 datasets to have a declared resistance and 3 datasets to have a suspected resistance. This means that Wang's model attributes one form of resistance to each dataset when in practice none of them is resistant. For the values of δ not discussed yet, the zero-inflation has just a small impact. In the case without zero-inflation 15 and in the situation with zero-inflation 17 datasets are identified as having a suspected resistance. For all other datasets a declared resistance is present. Independent of the zero-inflation, for $\delta = 0.06$ Wang's model is slightly better than the new model. Considering $\delta = 0.1$, $\delta = 0.15$ and $\delta = 0.2$ both models provide the requested results in the situation with and without zero-inflation. This means that for each dataset a resistance is declared.

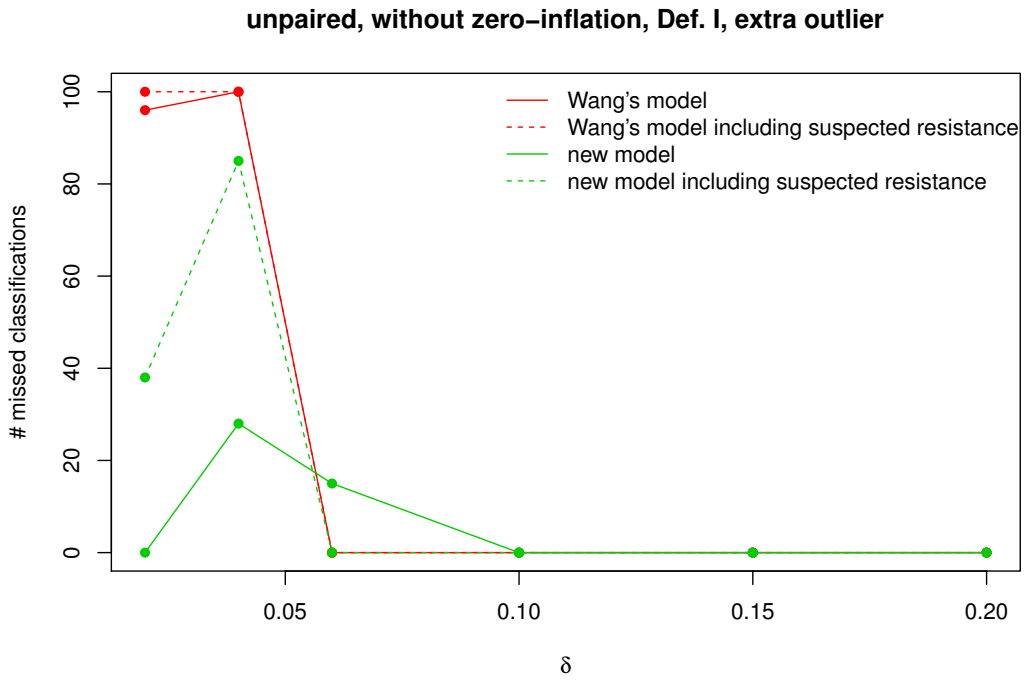

Figure 5.5.: The number of misclassified datasets using datasets where an obvious outlier is added. The data is unpaired and has no zero-inflation.

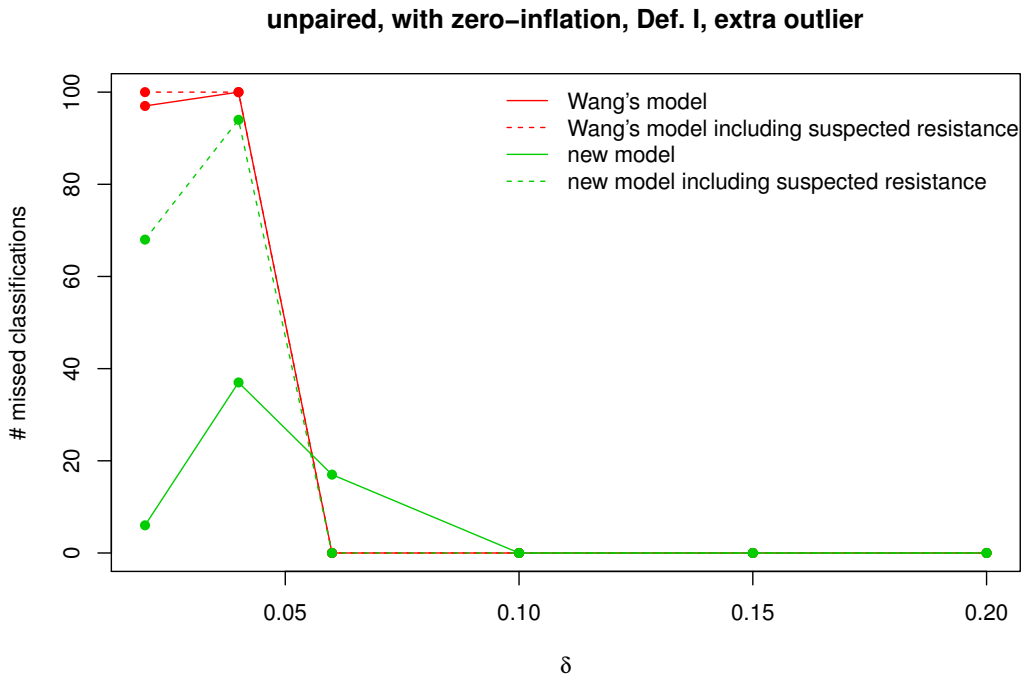

Figure 5.6.: The number of misclassified datasets using datasets where an obvious outlier is added. The data is unpaired and has zero-inflation.

5.1.2. Simulation without Extra Outlier

To compare the two models a further simulation is done. It works similar to the one before with the difference that no outlier is manually added to the datasets. This change has an influence on the marking of values to be outliers. Outlier Definition I not only identifies obvious outliers, but marks also some moderate values to be outliers. In this simulation no obvious outlier is added, so more moderate values are identified to be outliers. Without an obvious outlier, not the same values are considered as outliers and therefore the results will not be identical with the results from Section 5.1.1. The cases with outlier Definition II are skipped in this section, due to the fact that `simData2s()` does not generate this kind of outlier. Therefore the results using the new model are identical to the results using Wang's model.

The results obtained from simulations where no obvious outlier is added and just the datasets generated by `simData2s()` are used differ a lot from the results obtained by the simulation with an extra outlier. This holds for both the paired and for the unpaired design.

First the paired design is discussed (Figures 5.7 and 5.8). For $\delta = 0.02$ and $\delta = 0.04$ a perfect model is achieved with the new model. This means that there are no missed classifications. For these δ there are no wrongly declared resistances. This result is valid for both cases, with and without zero-inflation. In contrast, Wang's model reaches the result that for $\delta = 0.04$ in the case without zero-inflation for 2 datasets a resistance is declared and 5 resistances are suspected out of a total of 100 datasets. If a zero-inflation exists, 11 datasets are identified as having a declared resistance and 9 datasets as having a suspected resistance. For $\delta = 0.02$ Wang's model achieves perfect results, independent of any zero-inflation. In contradiction to these good results for small δ the new model performs very badly for $\delta = 0.06$ and $\delta = 0.1$. In these situations for all 100 datasets a resistance should be declared. First, the situation without zero-inflation is analysed. For $\delta = 0.06$ the new model identifies only 4 datasets to have a declared resistance and another 11 to have a suspected resistance. This leads to 85 completely misclassified datasets. In comparison, Wang's model just fails 41 times, meaning in 41 datasets neither a declared nor a suspected resistance is identified. For $\delta = 0.1$ the difference is still noteworthy. With the new model, for 1/5 of all datasets no kind of resistance is declared, where Wang's model has an error rate of 1/25. Regarding the situation with zero-inflation a similar pattern can be observed. While for $\delta = 0.6$, 88 datasets are misclassified using the new model, Wang's model just fails completely in 24 datasets. For $\delta = 0.1$ there are less failures in both models. The new model determines 32 datasets without any kind of resistance and Wang's model just 2. Using the last two deltas, $\delta = 0.15$ and $\delta = 0.2$, for datasets with and without zero-inflation, both models show the requested performance.

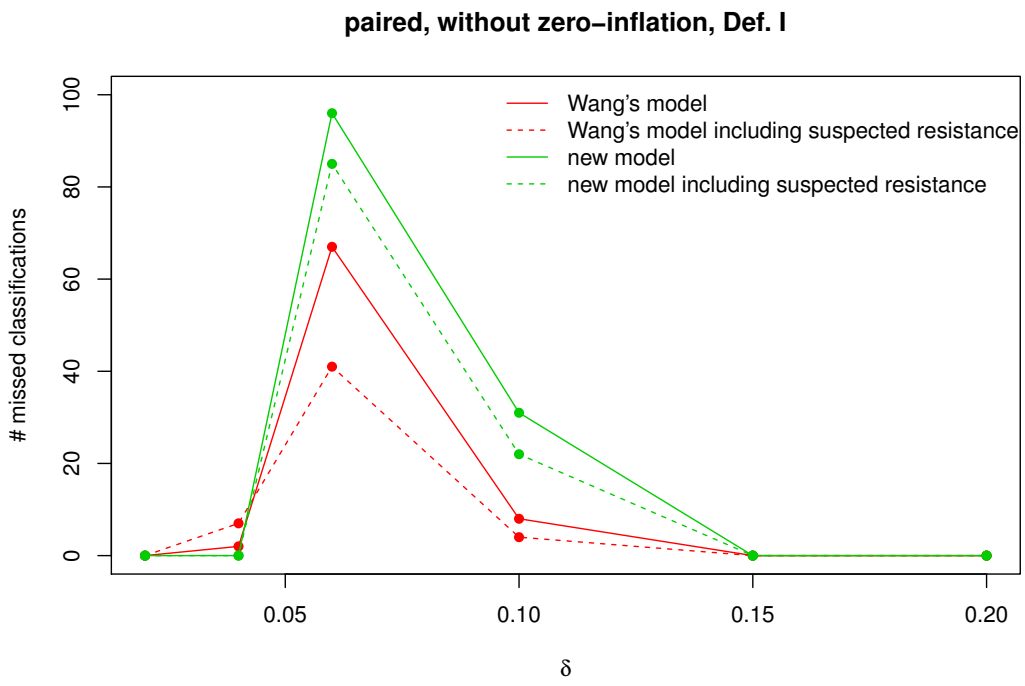

Figure 5.7.: The number of misclassified datasets using datasets generated with `simData2s()`. The data is paired and has no zero-inflation.

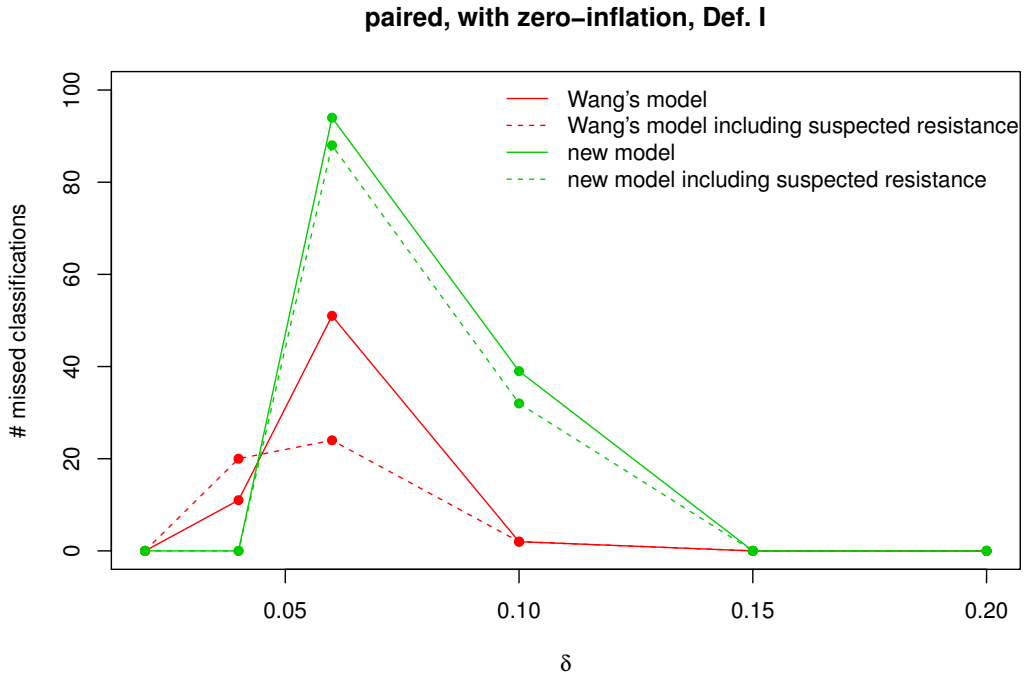

Figure 5.8.: The number of misclassified datasets using datasets generated with `simData2s()`. The data is paired and has zero-inflation.

In the unpaired case in which no extra outlier is added to the datasets, again the case without zero-inflation (Figure 5.9) and the case with zero-inflation (Figure 5.10) show a similar pattern. In both situations the new model achieves better results for $\delta = 0.02$ and $\delta = 0.04$ than Wang's model. But for the other δ Wang's model is more convincing. Not even for $\delta = 0.2$ a perfect result is obtained with the new model. In more detail, in the case without zero-inflation and $\delta = 0.02$ where no kind of resistance should be found, the new model does not misidentify any datasets. Using Wang's model, no resistance is declared but in 12 datasets a resistance is suspected. For $\delta = 0.4$ the new model is superior as it identifies 2 declared and 6 suspected resistances, whereas Wang's model obtains 14 declared and 44 suspected resistances. For $\delta = 0.06$, $\delta = 0.1$, $\delta = 0.15$ and $\delta = 0.2$ the new model totally misidentifies 63, 27, 3 and 1 datasets while Wang's model only misidentifies 17, 4, 1 and 0 datasets. In the situation with zero-inflation (Figure 5.10) almost identical results are observed.

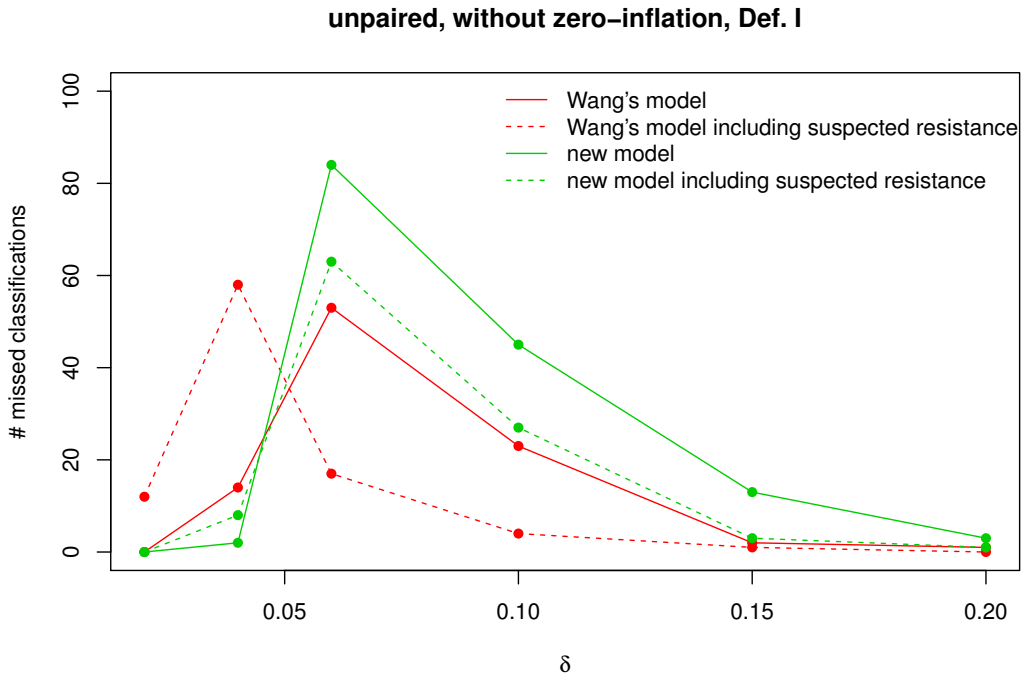

Figure 5.9.: The number of the misclassified datasets using datasets generated with `simData2s()`. The data is unpaired and has no zero-inflation.

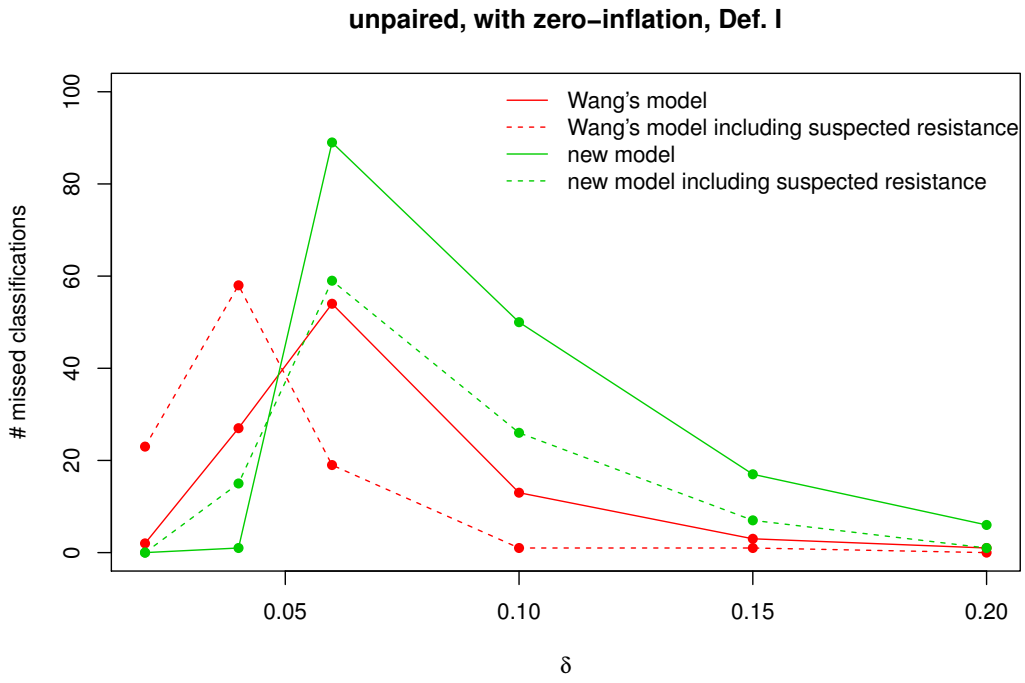

Figure 5.10.: The number of the misclassified datasets using datasets generated with `simData2s()`. The data is unpaired and has zero-inflation.

The expected results for the two models considering the different kind of datasets can be summarised as follows. For a dataset without an extra outlier, in general the performance of the new model will be somewhat worse than the performance of Wang's model. In contrast, the results obtained for datasets with an added outlier are supposed to be better with the new model than with Wang's model. This behaviour can be observed in the simulations above (Figures 5.1–5.10). These unsimilar results arise from unequal posterior densities for the faecal egg count reduction (fecr) for the two models based on the same datasets. For a good model, another condition has to be fulfilled. Comparing the posterior density of the fecr for a simulation without an extra outlier using Wang's model to a simulation with an extra outlier but analysed with the new model, should lead to similar plots. As an illustration, the unpaired design with zero-inflation using $\delta = 0.02$, $\mu = 500$, $\kappa = 0.5$ and $f = 50$ is observed. The densities for the mentioned situations are shown in Figure 5.11. Indeed, quite similar density plots for the fecr are obtained.

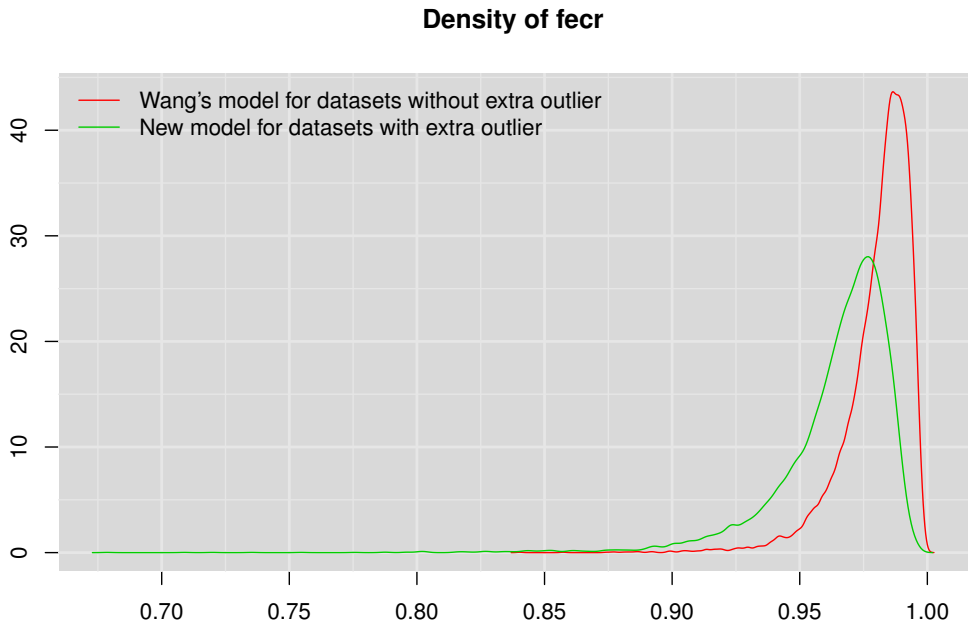

Figure 5.11.: The posterior density for the faecal egg count reduction using Wang's model to datasets without extra outlier and the new model to datasets with an extra outlier.

5.2. Simulation with Varying ϕ

To analyse how the new model performs for a varying zero-inflation parameter ϕ, the following simulations are executed. The basic model applied is the model for the paired design with zero-inflation, compare Eq (4.8). To all datasets an extra outlier is added. For the pre-treatment mean μ and the overdispersion parameter κ, the default values are applied. To identify outliers Definition I is considered. The simulations are done once for $\delta = 0.02$ and once for $\delta = 0.06$. For ϕ the values 0.1, 0.5 and 0.9 are checked.

By looking at Figure 5.12 and Figure 5.13 the attention is drawn to the constant results obtained by Wang's model. For all different ϕ the number of misclassified datasets does not change relevantly. This means that Wang's model is robust to changes in ϕ. Considering the new model under the condition of $\delta = 0.02$ a decrease in the number of missed classifications can be observed for increasing ϕ. This leads to the conclusion that through the extensions done in the new model, the robustness of Wang's model concerning ϕ is affected. At least for $\delta = 0.06$ the new model shows constant results.

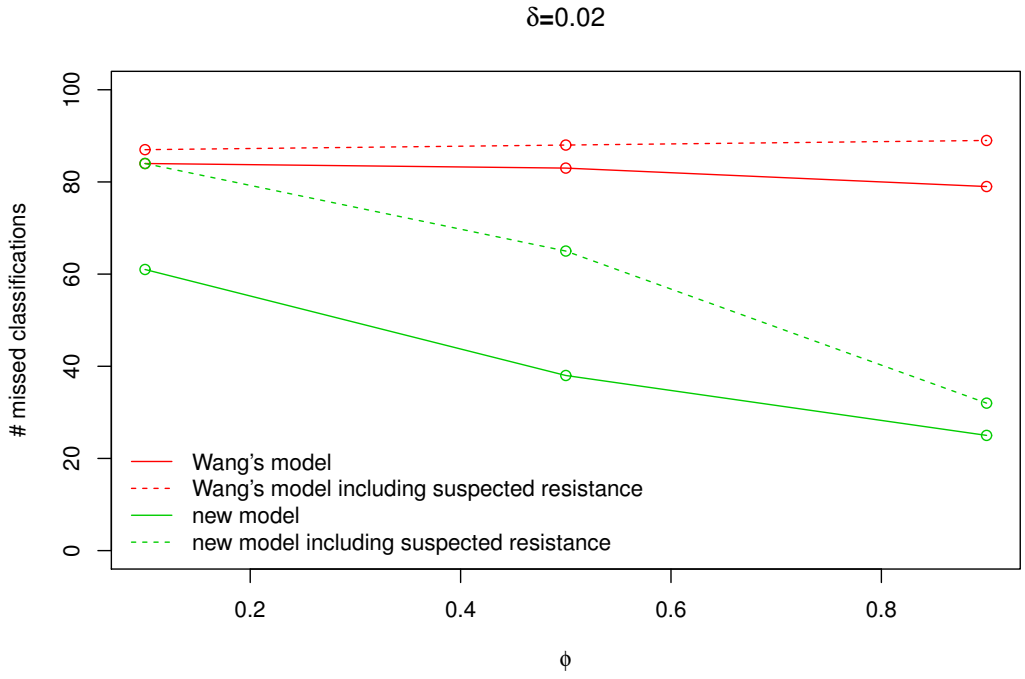

Figure 5.12.: The number of misclassified datasets for different values of ϕ, where $\delta = 0.02$.

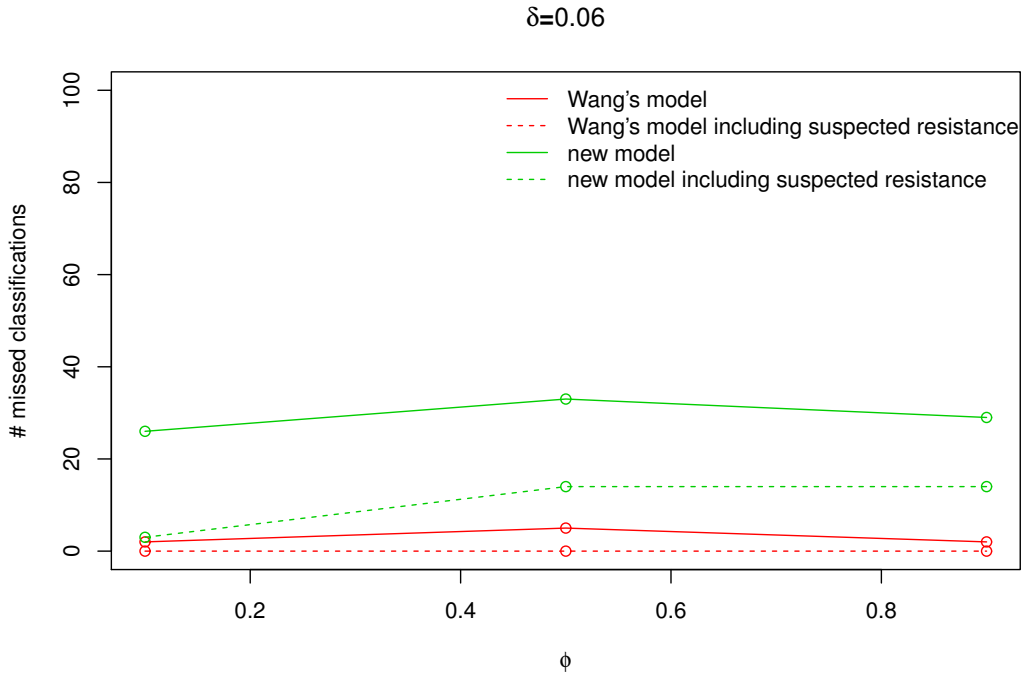

Figure 5.13.: The number of misclassified datasets for different values of ϕ, where $\delta = 0.06$.

5.3. Simulation with Varying μ

To analyse the new model's performance for different values for the pre-treatment mean μ, the following simulations are executed. The reference model is the paired design for datasets with a zero-inflation of 10% and an extra outlier. For κ, the default value of 0.5 is used. The simulations compare the number of misclassified datasets for $\mu = 100$, $\mu = 500$ and $\mu = 900$. In the first case $\delta = 0.02$ and in the second $\delta = 0.06$ is used. In order to be able to compare the results with Wang's model, these results are shown as well.

First, the illustration for $\delta = 0.02$ is discussed (Figure 5.14). For every μ that is analysed the new model obtains better results than Wang's model. This shows that independent of the pre-treatment mean the new model provides more accurate results. It is interesting that the improvement of the results using the new model compared to using Wang's model increases, the higher the pre-treatment mean is. Although Wang's model also shows a slightly better performance for higher μ, the change using the new model is more serious.

Next, the situation for $\delta = 0.06$ is investigated (Figure 5.15). For all μ that are analysed, Wang's model is superior. Although for both models, the number of misclassified datasets increases for higher μ. The deterioration is stronger for the new model.

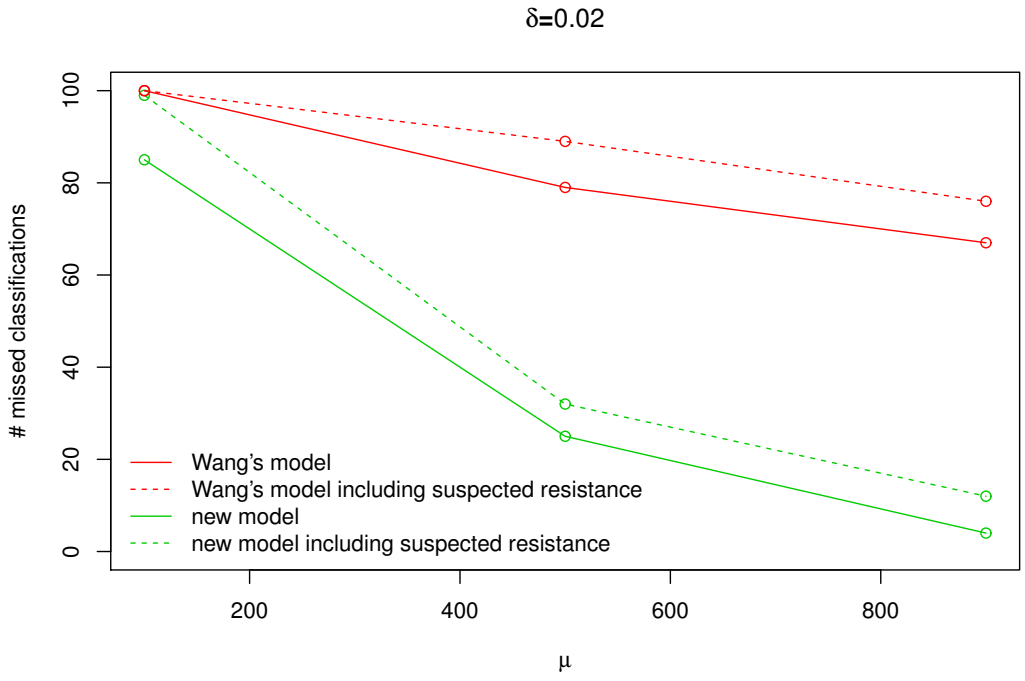

Figure 5.14.: The number of misclassified datasets for different values of μ, where $\delta = 0.02$.

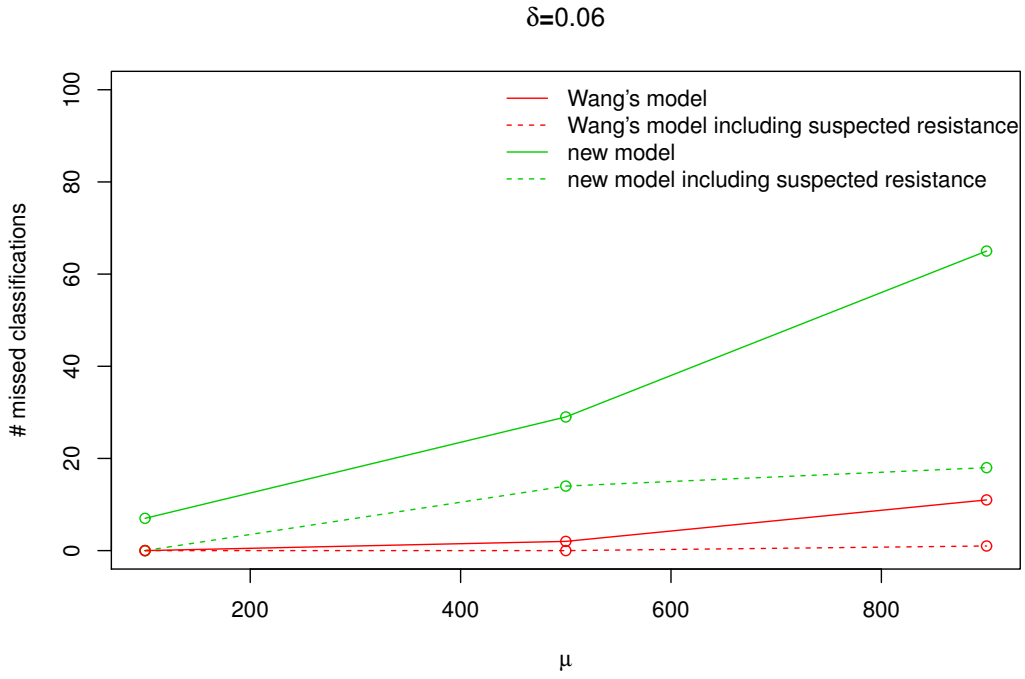

Figure 5.15.: The number of misclassified datasets for different values of μ, where $\delta = 0.06$.

To give a better understanding of this increase and decrease, further analysis can be done by looking at the output of the MCMC sampling on the hierarchical models. For $\delta = 0.02$ Figure 5.16 shows the posterior densities for the faecal egg count reduction. The higher μ is, the narrower the posterior density gets and the values for the fecr increase. This leads in practice to more datasets that are identified as not having a resistance. For $\delta = 0.02$ the perfect model does not identify any resistances. The improvement of the new model for higher μ can also be seen in these density plots.

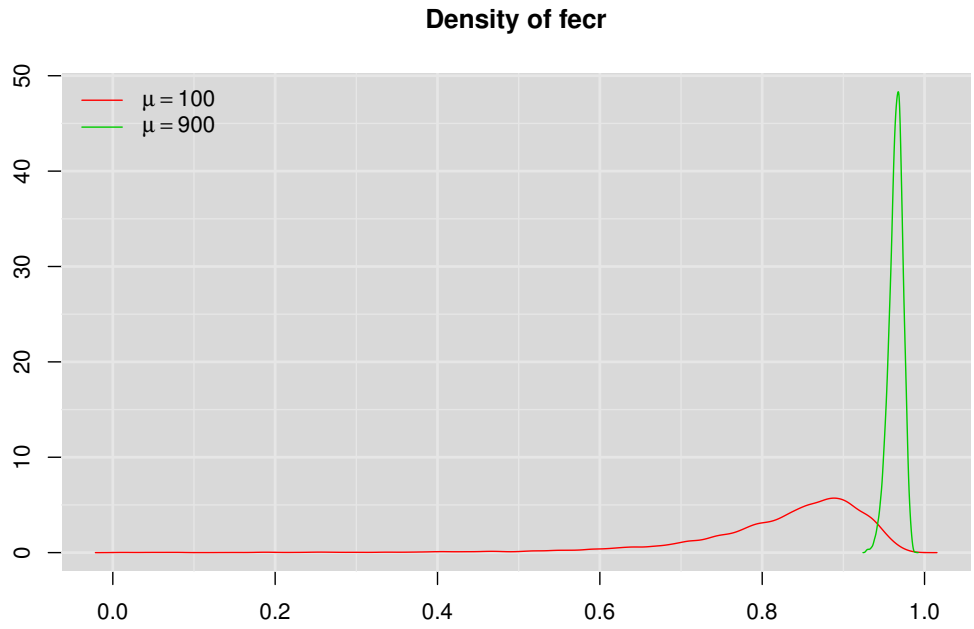

Figure 5.16.: The posterior density for the faecal egg count reduction using $\mu = 100$ and $\mu = 900$.

5.4. Simulation with Varying κ

In this section the performance of the new model for different values of the overdispersion parameter κ is shown. To interpret the results, they are compared to the results obtained with Wang's model. The paired design with zero-inflation serves as underlying model. Again an extra outlier is added to each dataset. This model is analysed for $\kappa = 0.05$, $\kappa = 2$ and $\kappa = 6$. The simulation is execute using $\delta = 0.02$ and $\delta = 0.04$. For the other parameters, the default values are used: $n = 25$, $f = 50$, $\phi = 0.09$ and $\mu = 500$.

In a first step, the achievements for $\delta = 0.02$ are analysed (Figure 5.17). In this situation a perfect model does not identify any datasets to have a resistances. For all κ analysed, the new model performs better. In addition, the improvement of the results obtained with the new model compared to the results of Wang's model seem to increase for higher κ. The number of wrongly identified declared resistance for $\delta = 0.02$ using the new model are 25, 32 and 13 for $\kappa = \{0.5, 2, 6\}$ whereas using Wang's model, the corresponding results are higher. For 79, 97 and 100 datasets a resistance is declared. Wang's model leads to an increase of the number of resistance for increasing κ, whereas the results using the new model at first increase a bit but then decrease.

For $\delta = 0.06$ (Figure 5.18) Wang's model performs better than the new model for all κ. The results using Wang's model are quite as good as the results obtained with a perfect model. Compared to the situation with $\delta = 0.02$, the difference between the two models is smaller for $\delta = 0.06$. For increasing κ, the results obtained using the new model approach the results achieved with Wang's model.

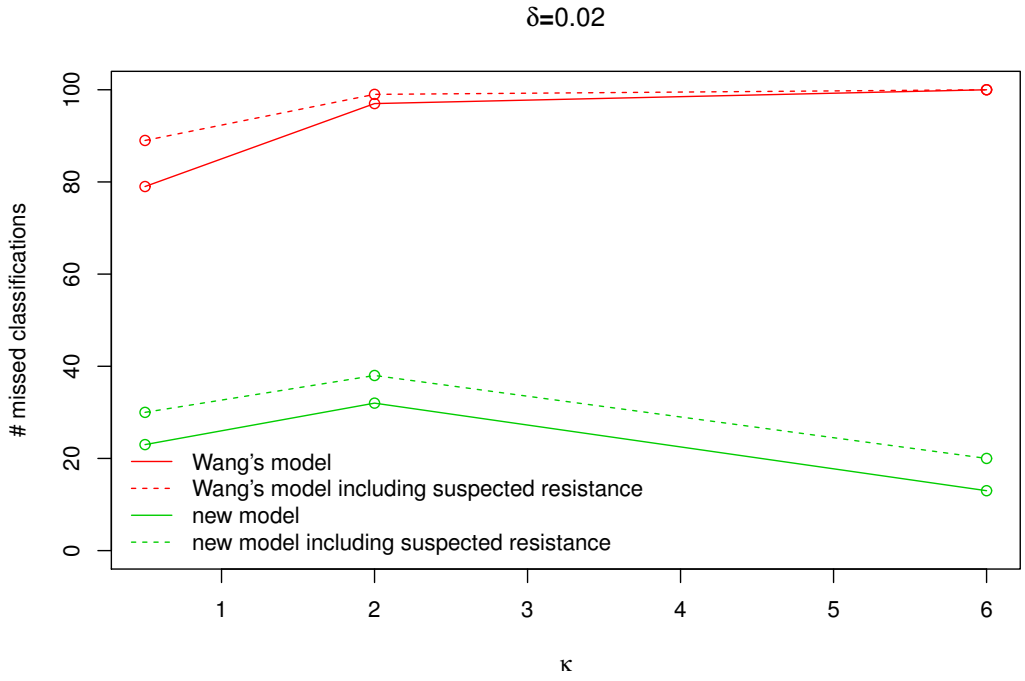

Figure 5.17.: The number of misclassified datasets for different values of κ, where $\delta = 0.02$.

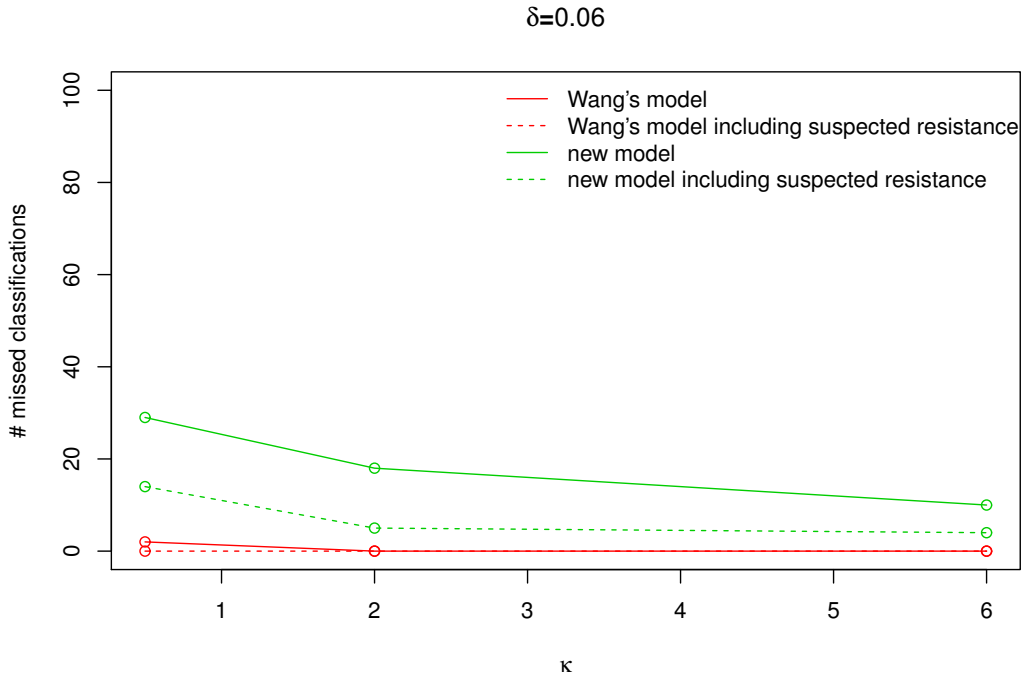

Figure 5.18.: The number of misclassified datasets for different values of κ, where $\delta = 0.06$.

5.5. Simulation with Varying the Prior for α

The goal of this section is to find out, how robust the new model is concerning the prior for α. The prior used in the new model for the scaling factor α is defined in Eq (4.12). The simulation is again done for the paired design with zero-inflation and an extra outlier for each dataset. In addition, outlier Definition I is considered. The following default values for the other required parameters are used: $\phi = 0.9$, $\mu = 500$ and $\kappa = 0.5$. To to the simulation, three different priors for α are taken into account, namely

$$\alpha \sim \text{Normal}\left(\frac{wmo}{postmean}, 1\right) \; I(\alpha \geq 1), \tag{5.1}$$

$$\alpha \sim \text{Normal}(10, 25) \; I(\alpha \geq 1), \tag{5.2}$$

$$\alpha \sim \text{Normal}(50, 900) \; I(\alpha \geq 1). \tag{5.3}$$

Eq (5.1) is a prior that is quite similar to the one used in the extended model. The only difference is the smaller variance. For Eq (5.2) and (5.3) uninformative priors are chosen. The performance of these priors is analysed for $\delta = \{0.02,\ 0.04,\ 0.06,\ 0.1,\ 0.15,\ 0.2\}$ (Figure 5.19).

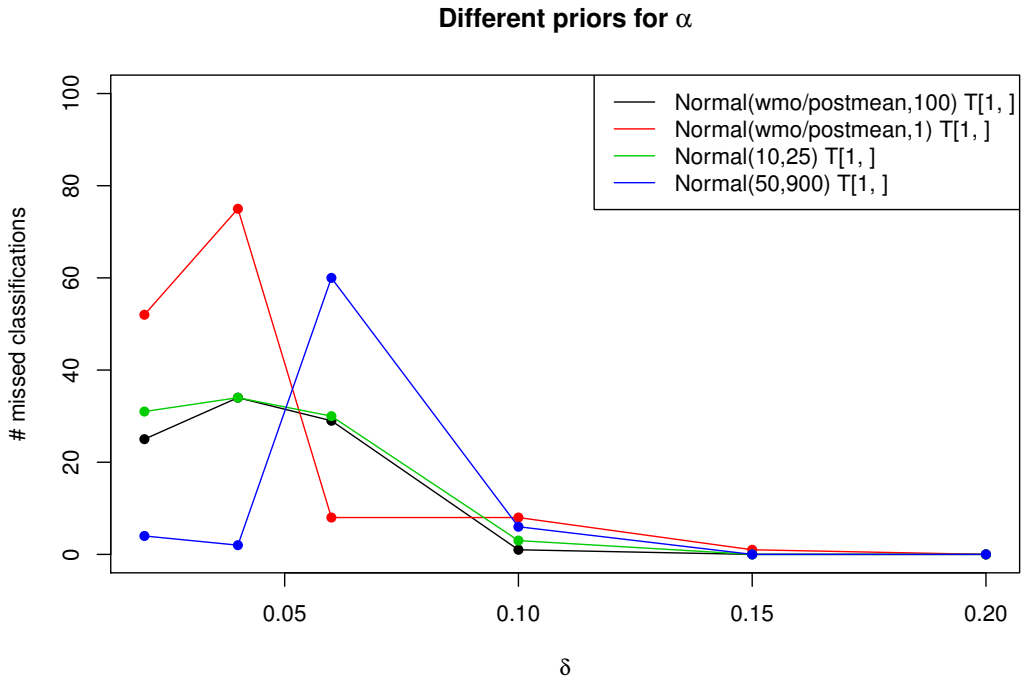

Figure 5.19.: The number of misclassified datasets for different priors for α. The paired design with zero-inflation and an added outlier is considered. Only the declared resistances are considered, the suspected resistances are supposed to be no resistances.

First, $\alpha \sim \text{Normal}(wmo/postmean, 1) \; I(\alpha \geq 1)$ is compared to the one used in the new model (Eq (4.12)). Regarding the smallest two δ, the smaller variance leads to a worse result. This means when using $\alpha \sim \text{Normal}(wmo/postmean, 1) \; I(\alpha \geq 1)$ more datasets are identified as having a declared resistance. In contrast for $\delta = 0.06$, using a variance equals 1 results in less missed classifications. For the higher δ, the small variance for the prior of α leads again to a worse result. For the more general priors, different results can be observed. Using the prior

$\alpha \sim \text{Normal}(10, 25)\ I(\alpha \geq 1)$, the model leads to very similar results as with the prior defined in Eq (4.12). For $\alpha \sim \text{Normal}(50, 900)\ I(\alpha \geq 1)$ almost no missed classification appear for small δ. However, for $\delta = 0.06$ a lot of datasets are wrongly identified. For the three highest δ, the results do not vary considerably.

5.6. Summary Simulation Results

In this section the results of the simulations in Sections 5.1–5.5 are summed up. In a first step, the new model was tested for different δ in a paired as well as in an unpaired design, each with and without zero-inflation. In Section 5.1.1, where an outlier is added to the datasets generated by `simData2s()`, also the differences caused by different definitions of outliers are considered. Independent of the design and the fact whether a zero-inflation is present or not, for datasets with a reduction in mean of 98% and 96% and an added outlier, the new model leads to less wrongly identified resistances. This holds for both outlier definitions. In these settings the new model performs better and mitigates the effect of a single outlier compared to the model suggested by Wang. If the reduction in mean from pre- to post-treatment is 94% and an outlier is added, the new model misclassifies more datasets than Wang's model, especially when the outlier Definition I is used. If the reduction in the mean is just 90% and less, the new model as well as Wang's model achieve the requested results. Regarding the situation where no extra outlier is added, only outlier Definition I can be used. Again, for a high reduction in mean, the new model is superior to Wang's model. If the mean reduction is lower, the new model performs worse than Wang's model.

Additionally, the influence of a variation in the zero-inflation parameter ϕ is discussed. Considering the new model for $\delta = 0.02$, the number of misclassified datasets reduces for increasing ϕ. For $\delta = 0.06$ the number of missed classifications stays constant. Wang's model shows a non-varying number of failures for both situations. These simulations show that the new model is less robust to varying ϕ values than Wang's model.

Further, the performance of the new model for changing the pre-treatment mean were simulated and compared to Wang's model. While for $\delta = 0.02$, the new model achieves better results, for $\delta = 0.06$ Wang's model is superior. The higher the pre-treatment mean is, the more the difference between the models increases. The new model obtains much better result than Wang's model for small δ. On the other hand the new model achieves worse results for $\delta = 0.06$ compared to Wang's model if μ increases. The pre-treatment mean does not influence the fact which model is better but the intensity of how much better or worse the new model is.

Moreover, the influence of the overdispersion parameter κ was investigated. The results of the new model improve slightly for increasing κ compared to the result obtained with Wang's model. In general it is concluded that the new model is not sensitive to the overdispersion parameter.

The change in the results due to a different prior for α is crucial. Some priors lead to requested results for small δ but produce lots of misclassifications for higher δ. Other priors result in exactly the inverse performance. Further, a prior is found that leads to results, which are comparable to the results achieved using the prior specified for the new models. But this particular prior does not depend on the data. This simulation shows that the new model is highly sensitive to the choice of the prior for the scaling factor α.

6. Discussion

In recent years the appearance of anthelmintic resistance in flocks of cattle, sheep and goats has increased. As a result, methods to detect these resistances have become more important. One of these methods using zero-inflated hierarchical models is introduced by Wang as an extension of the studies done by Paul et al. (2014). By its usage, Wang's models show some weaknesses relating to outliers in datasets. In this thesis the zero-inflated hierarchical models are therefore extended in a way that outliers in faecal egg counts are detected and their effect on the result is reduced.

To build the examinations on a solid base, the term outlier needed to be concretised. That is when the first difficulties arose whilst doing this thesis. No established definition could be detected, so an own definition had to be formulated. Two outlier definitions were presented in Section 4.2. It turned out that both have their advantages and disadvantages. First, outlier Definition I is considered. Looking at the simulations in Section 5.1.2, where datasets without an extra outlier are used, the bad performance of the new model for $\delta = 0.06$ and $\delta = 0.1$ attracts the attention (Figures 5.7–5.10). This problem has its origin in the definition of the outliers. In these simulations no extra outlier is introduced to the datasets, therefore many of the moderate values are identified to be outliers. Their influence on the posterior estimates is mitigated, which leads to lots of misclassified datasets. Another weak point of the outlier Definition I arises when datasets with zero-inflation are considered. To detect all the relevant outliers, the zeros indicating the non-infected animals, should be removed. Since by reason of the definition only the post-treatment values are considered, it is not known which zeros belong to animals with a reduction of 100% in the egg counts and which belong to the non-infected animals. Due to this fact, the first step in the process of identifying values as outliers is biased. Therefore, the dividing point between outlier and non-outlier regarding the boxplot definition of an outlier is lower than in a setting where cleaned-up data is used. This leads to more values that are identified as being outliers. On the other hand, the advantage of this definition is its scope of application. It can be applied for all versions of models, to the paired and unpaired design and as well to situations with and without zero-inflation. This leads directly to the discussion about the disadvantage of the second outlier definition. Outlier Definition II can only be applied in the paired design. But for this setting, the number of misclassified datasets for the problematic δ mentioned above are lower than using outlier Definition I. An other plus of outlier Definition II is the handling of zero-inflated datasets, where extra zeros do not have an impact on the outlier definition.

After defining the outliers, the most time-intensive and delicate task was to determine a prior for the scaling parameter α. As seen in Section 5.5, the choice of this prior leads to very different results. One of the goals persued in this thesis was to find an uninformative prior for α. Something of similar appearance to the priors for μ, κ, δ, and ϕ was desired. Such a prior could not be found, as in each case datasets were detected for which the prior leads to very bad results. Therefore more informative priors, i.e. priors depending on the data, had to be taken into consideration. Regarding this process it can be seen that the potential of the prior of α has possibly not fully been enveloped yet. Throughout the investigations it was seen that also the value of an outlier leads to problems for models using an uninformative prior for α.

Summing up the simulations, an important message can be derived. Looking at the plots in Chapter 5, a trade-off between the new models and Wang's models shows up. Using Wang's model, lots of datasets with a strong reduction in mean but which contain a single outlier are wrongly identified as having a resistance. On the other hand, using the new models, datasets for which a resistance is obviously present are marked as not having a resistance. The user should be aware of this problem.

In conclusion it can be said, that this thesis helps to solve the problem described at the beginning, although it is not a terminal solution. In a next step, investigations can be done on the outlier definitions. The outlier Definition I could be explored in more detail. For example the impact of the 95% quantile chosen in the 3rd step of the outlier Definition I has to be analysed further. It should be investigated whether another quantile results in a better performance. Another aspect that needs to be addressed is the presence of more than one extra outlier. The datasets used in this thesis included at most one single extra outlier. The performance of the new models for several high outliers or for high and moderate outliers is not analysed yet. In addition, an issue considering the influence of the absolute value of the included outlier arose. Especially the problem of finding an uninformative prior for α is affected by this fact. Inquiries on finding a range for reasonable outliers depending on μ can lead to further insights. Moreover, the findings of this thesis indicate that the prior for α plays an important role in determining a good model. On this topic further investigations are suggested.

With the proposed model an improvement for the practical use is achieved. Scientists, veterinary doctors and also farmers can use both models to obtain more accurate results for the detection of anthelmintic resistances.

Bibliography

Blood, D. and Studdert, V. (2006). *Saunders Comprehensive Veterinary Dictionary (Hard Cover)*. Saunders Ltd., 3rd edition. 1

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1):1–32. 6

Coles, G., Bauer, C., Borgsteede, F., Geerts, S., Klei, T., Taylor, M., and Waaller, P. (1992). World Association for the Advancement of Veterinary Parasitology (W.A.A.V.P.) methods for the detection of anthelmintic resistance in nematodes of veterinary importance. *Veterinary Parasitology*, 44:35–44. 1

Kaplan, R. M. (2004). Drug resistance in nematodes of veterinary importance: a status report. *Trends in Parasitology*, 20(10):477–481. 1

McMahon, C., Bartley, D., Edgar, H., S.E.Ellison, Barley, J., Malone, F., Hanna, R., Brennan, G., and Fairweather, I. (2013). Anthelmintic resistance in Northern Ireland (I): Prevalence of resistance in ovine gastointestinal nematodes, as determined through faecal egg count reduction testing. *Veterinary Parasitology*, 195:122–130. 9

Nist/Sematech (2003). e-handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/. [Online; accessed 3-August-2017]. 9

Panaretos, V. M. (2016). *Statistics for Mathematicians: A Rigorous First Course*. Birkhäuser. 10

Paul, M., Torgerson, P. R., Höglund, J., and Furrer, R. (2014). Hierarchical modelling of faecal egg counts to asses anthelmintic efficacy. *arXiv:1401.2642 (stat.AP)*. 1, 3, 33

Perry, B. and Randolph, T. (1999). Improving the assessment of the economic impact of parasitic diseases and of their control in production animals. *Veterinary Parasitology*, 84:145–168.

Pfukenyi, D., Mukaratirwa, S., Willingham, A., and Monrad, J. (2007). Epidemiological studies of parasitic gastrointestinal nematodes, cestodes and coccidia infections in cattle in the highveld and lowveld communal grazing areas of Zimbabwe. *Onderstepoort Journal of Veterinary Research*, 74:129–142. 1

Roeber, F., Jex, A. R., and Gasser, R. B. (2013). Impact of gastrointestinal parasitic nematodes of sheep, and the role of advanced molecular tools for exploring epidemiology and drug resistance - an Australian perspective. *Parasites & Vectors*, 6. 1

Slocombe, J. O. D., Coté, J. F., and de Gannes, R. V. (2007). The persistence of benzimidazole-resistant cyathostomes on horse farms in Ontario over 10 years and the effectiveness of

ivermectin and moxidectin against these resistant strains. *Canadian Veterinary Journal*, 48:56–60. 1

Stan Development Team (2017). Stan Modeling Language: User's Guide and Reference Manual. http://mc-stan.org. Version 2.15.0. 6

Tariq, K. A. (2014). A Review of the Epidemiology and Control of Gastrointestinal Nematode Infections of Small Ruminants. *Proceedings of the National Academy of Science, India Secion B: Biological Science*, 85(2):693–703. 1

van Houtert, M. F. J. and Sykes, A. R. (1996). Implications of Nutrition for the Ability of Ruminants to Wihstand Gastrointestinal Nematode Infections. *International Journal for Parasitology*, 26(11):1151–1168. 1

Waller, P. J. (1997a). Anthelmintic resistance. *Veterinary Parasitology*, 72:391–412. 1

Waller, P. J. (1997b). Sustainable helminth control of ruminants in developing countries. *Veterinary Parasitology*, 71:195–207. 1

Wang, C. and Paul, M. (2017). *eggCounts: Hierarchical Modelling of Faecal Egg Counts*. R package version 1.4. 3

Wang, C., Torgerson, P. R., Höglund, J., and Furrer, R. (2017). Zero-inflated hierarchical models for faecal egg counts to assess anthelmintic efficacy. *Veterinary Parasitology*, 235:20–28. i, 1, 3, 15

Waruiru, R., Thamsborg, S., Nansen, P., Kyvsgaard, N., Bogh, H., Munyua, W., and Gathuma, J. (2001). The Epidemiology of Gastrointestinal Nematodes of Dairy Cattle in Central Kenya. *Tropical Animal Health and Production*, 33:173–187. 1

Wolstanholme, A. J., Fairweather, I., Prichard, R., von Samson-Himmelstjerna, G., and Sangster, N. C. (2004). Drug resistance in veterinary helminths. *Trends in Parasitology*, 20(10):469–476. 1

Zanzani, S. A., Gazzonis, A. L., Cerbo, A. D., Varady, M., and Manfredi, M. T. (2014). Gastrointestinal nematodes of dairy goats, anthelmintic resistance and practices of parasite control in Northern Italy. *BMC Veterinary Research*, 10(1):1–10. 1

Appendices

In this chapter the Stan codes of the four models are printed. Further, different codes in R are shown. This includes the code for `fecr_stan_outlier()` as well as for some auxiliary functions, which are needed inside the main function. In the last section, the R Documentation for `fecr_stan_outlier()` is given.

A. Stan Codes

This appendix contains the formulation of the Bayesian hierarchical models for the different cases in Stan language. The models for all four cases, meaning the paired and unpaired design, each with and without zero-inflation, are presented.

A.1. Paired Model without Zero-Inflation with Outliers

```
# paired without zero-inflation with outliers

data{
  int J; // number of animals
  int ystararaw[J]; // after treatment McMaster count
  int ystarbraw[J]; // before treatment McMaster count
  int fpre[J];
  int fpost[J];
  real w[J];
  real wmo;
  real postmean;
}
parameters{
  real<lower=0> kappa;
  real<lower=0> mu;
  real<lower=0,upper=1> delta;
  real<lower=0> mub[J];
  real <lower=1>alpha;
}
transformed parameters{
  real lambdaa[J];
  real lambdab[J];
  real kappamu;
  real lambdaaAlpha[J];
  for (i in 1:J){
    lambdab[i] = mub[i]/fpre[i];
  }
  for (i in 1:J){
    lambdaa[i] = delta*mub[i]/fpost[i];
  }
  kappamu = kappa/mu;
  for (i in 1:J){
    lambdaaAlpha[i] = alpha*lambdaa[i];
  }
}
model{
  mu ~ gamma(1,0.001);    // prior
  kappa ~ gamma(1,0.7);
  delta ~ beta(1,1);
  alpha ~ normal(wmo/postmean,10) T[1,];
  target += gamma_lpdf(mub | kappa,kappamu);   // likelihoods
  ystarbraw ~ poisson(lambdab);
  for (n in 1:J){
    target += w[n]*poisson_lpmf(ystararaw[n] | lambdaa[n]) +
              (1-w[n])*poisson_lpmf(ystararaw[n] | lambdaaAlpha[n] );
  }
}
```

A.2. Unpaired Model without Zero-Inflation with Outliers

```
# unpaired without zero-inflation with outliers

data{
  int Ja; // number of animals
  int Jb;
  int ystararaw[Ja]; // after treatment McMaster count
  int ystarbraw[Jb]; // before treatment McMaster count
  int fpre[Ja];
  int fpost[Jb];
  real w[Ja];
  real wmo;
  real postmean;
}
parameters{
  real<lower=0> kappa;
  real<lower=0> mu;
  real<lower=0, upper = 1> delta;
  real<lower=0> mub[Jb]; # true epg before treatment
  real<lower=0> mua[Ja]; # true epg after treatment
  real <lower=1>alpha;
}
transformed parameters{
  real lambdaa[Ja];
  real lambdab[Jb];
  real kappamu;
  real lambdaaAlpha[Ja];
  for (i in 1:Jb){
    lambdab[i] = mub[i]/fpre[i];
  }
  for (i in 1:Ja){
    lambdaa[i] = delta*mua[i]/fpost[i];
  }
  kappamu = kappa/mu;
  for (i in 1:Ja){
    lambdaaAlpha[i] = alpha*lambdaa[i];
  }
}
model{
  mu ~ gamma(1,0.001);    // prior
  kappa ~ gamma(1,0.7);
  delta ~ beta(1,1);
  alpha ~ normal(wmo/postmean,10) T[1,];
  target += gamma_lpdf(mub | kappa,kappamu)+gamma_lpdf(mua | kappa,kappamu);   // likelihoods
  ystarbraw ~ poisson(lambdab);
  for (n in 1:Ja){
    target += w[n]*poisson_lpmf(ystararaw[n] | lambdaa[n]) +
              (1-w[n])*poisson_lpmf(ystararaw[n] | lambdaaAlpha[n] );
  }
}
```

A.3. Paired Model with Zero-Inflation with Outliers

```
# paired with zero-inflation with outliers

data{
  int J; // number of animals
  int ystararaw[J]; // after treatment McMaster count
  int ystarbraw[J]; // before treatment McMaster count
  int fpre[J];
  int fpost[J];
  real w[J];
  real wmo;
  real postmean;
}
parameters{
  real<lower=0> kappa;
  real<lower=0> mu;
  real<lower=0,upper=1> delta;
  real<lower=0> mub[J];
  real<lower=0,upper=1> phi;
  real <lower=1>alpha;
}
transformed parameters{
  real lambdaa[J];
  real lambdab[J];
  real kappamu;
  real lambdaaAlpha[J];
  for (i in 1:J){
    lambdab[i] = mub[i]/fpre[i];
    lambdaa[i] = delta*mub[i]/fpost[i];
  }
  kappamu = kappa/mu;
  for (i in 1:J){
    lambdaaAlpha[i] = alpha*lambdaa[i];
  }
}
model{
  mu ~ gamma(1,0.001);    // prior
  kappa ~ gamma(1,0.7);
  delta ~ beta(1,1);
  phi ~ beta(1,1);
  alpha ~ normal(wmo/postmean,10) T[1,];
  mub ~ gamma(kappa,kappamu);           // likelihoods
  for (n in 1:J) {
    if (ystarbraw[n] == 0)
      target += log_sum_exp(bernoulli_lpmf(1 | phi),
                bernoulli_lpmf(0 | phi)+poisson_lpmf(ystarbraw[n] | lambdab[n]));
      else
        target += bernoulli_lpmf(0 | phi) + poisson_lpmf(ystarbraw[n] | lambdab[n]);
  }
  for (n in 1:J) {
    if (ystararaw[n] == 0)
      target += log_sum_exp(bernoulli_lpmf(1 | phi),
                bernoulli_lpmf(0 | phi)+poisson_lpmf(ystararaw[n] | lambdaa[n]));
    else
      target += bernoulli_lpmf(0 | phi) +
                  (w[n]*poisson_lpmf(ystararaw[n] | lambdaa[n])+
                  (1-w[n])*poisson_lpmf(ystararaw[n] | lambdaaAlpha[n]) );
  }
}
```

A.4. Unpaired Model with Zero-Inflation with Outliers

```
# unpaired with zero-inflation with outliers

data{
  int Ja; // number of animals
  int Jb;
  int ystararaw[Ja]; // after treatment McMaster count
  int ystarbraw[Jb]; // before treatment McMaster count
  int fpre[Ja];
  int fpost[Jb];
  real w[Ja];
  real wmo;
  real postmean;
}
parameters{
  real<lower=0> kappa;
  real<lower=0> mu;
  real<lower=0, upper=1> delta;
  real<lower=0> mub[Jb];
  real<lower=0> mua[Ja];
  real<lower=0,upper=1> phi;
  real <lower=1>alpha;
}
transformed parameters{
   real lambdaa[Ja];
   real lambdab[Jb];
   real kappamu;
   real lambdaaAlpha[Ja];
   for (i in 1:Jb){
     lambdab[i] = mub[i]/fpre[i];
   }
   for (i in 1:Ja){
     lambdaa[i] = delta*mua[i]/fpost[i];
   }
   kappamu = kappa/mu;
   for (i in 1:Ja){
     lambdaaAlpha[i] = alpha*lambdaa[i];
   }
}
model{
   mu ~ gamma(1,0.001);    // prior
   kappa ~ gamma(1,0.7);
   delta ~ beta(1,1);
   phi ~ beta(1,1);
   alpha ~ normal(wmo/postmean,10) T[1,];
   mub ~ gamma(kappa,kappamu);;   // likelihoods
   mua ~ gamma(kappa,kappamu);
   for (n in 1:Jb) {
    if (ystarbraw[n] == 0)
       target += log_sum_exp(bernoulli_lpmf(1 | phi),
                  bernoulli_lpmf(0 | phi)+poisson_lpmf(ystarbraw[n] | lambdab[n]));
    else
       target += bernoulli_lpmf(0 | phi) + poisson_lpmf(ystarbraw[n] | lambdab[n]);
   }
   for (n in 1:Ja) {
     if (ystararaw[n] == 0){
       target += log_sum_exp(bernoulli_lpmf(1 | phi),
                  bernoulli_lpmf(0 | phi)+poisson_lpmf(ystararaw[n] | lambdaa[n]));
    }else{
      target += bernoulli_lpmf(0 | phi) + (w[n]*poisson_lpmf(ystararaw[n] | lambdaa[n]) +
                (1-w[n])*poisson_lpmf(ystararaw[n]|lambdaaAlpha[n]));
    }
   }
}
```

B. R Code

This appendix contains the R Code of the new `fecr_stan_outlier()` function, which is an extension of `fecr_stan()` contained in the `eggCounts` package. Further, the R Code of the auxiliary functions needed in `fecr_stan_outlier()` are shown.

B.1. fecr_stan_outlier()

```
# fecr_stan() extended

fecr_stan_outlier <- function (preFEC, postFEC, rawCounts = FALSE, preCF = 50, postCF = preCF,
          paired = TRUE, zeroInflation = TRUE, outlierdetection = FALSE, outlier = c("quant","ratio"),
          muPrior, kappaPrior, deltaPrior, phiPrior, alphaPrior, nsamples = 12000, nburnin = 2000,
          thinning = 1, nchain = 1, ncore = 1, adaptdelta = 0.9, verbose = FALSE)
{
  #outlier detection with quant (1st def) or ratio (2nd def).
  #1st def for every case possible, 2nd def just for paired
  #if no outlierdetection: FALSE

  preN <- length(preFEC)
  postN <- length(postFEC)
  checkCF <- function(CF) {
    (CF < 0) | (ceiling(CF) != floor(CF))
  }
  if (any(sapply(preCF, checkCF)))
    stop("correction factor(s) should be a positive integer",
         call. = FALSE)
  if (any(sapply(postCF, checkCF)))
    stop("correction factor(s) should be a positive integer",
         call. = FALSE)
  if (length(preCF) > 1 && length(preCF) != preN)
    stop("Lengths of the vectors preCF and preFEC do not match\n")
  if (length(postCF) > 1 && length(postCF) != postN)
    stop("Lengths of the vectors postCF and postFEC do not match\n")
  preDilution <- preCF
  postDilution <- postCF
  if (rawCounts) {
    preDilution <- postDilution <- 1
  }

  checkpars(nburnin, nsamples, thinning, nchain, ncore, rawCounts,
            adaptdelta, zeroInflation, verbose)

  if (!is.logical(outlierdetection))
    stop("The outlierdetection argument must be a logical", call. = FALSE)

  quant_outlier <- outlier == "quant"
  ratio_outlier <- outlier == "ratio"
  if (!quant_outlier){
    if (!ratio_outlier){
      stop("The outlier method is not valid", call. = FALSE)
    }
  }
  if ( paired == FALSE){
   if (ratio_outlier){
     stop("In the unpaired case, outliers can not be detected using ratios.", call. = FALSE)
   }
  }


  if (!is.logical(paired))
    stop("The paired argument must be a logical", call. = FALSE)
  if (any(preFEC%%preDilution != 0))
    stop(paste(c("Correction factor preCF does not match the given pre-treatment faecal egg counts.
                 \n\n  A possible correction factor (the largest common divisor) is"),
               mGCD(preFEC)))
  preFEC <- preFEC/preDilution
  if (any(postFEC%%postDilution != 0))
    stop(paste(c("Correction factor postCF does not match the given post-treatment faecal egg counts.
                 \n\n  A possible correction factor (the largest common divisor) is"),
               mGCD(postFEC)))
  postFEC <- postFEC/postDilution
  if (!rawCounts) {
    if (mean(preFEC) < mean(postFEC))
      cat("NOTE: mean of pre-treatment is smaller of post-treatment. Results may be unreliable.\n")
    if (median(preFEC) < median(postFEC))
      cat("NOTE: median of pre-treatment is smaller of post-treatment. Results may be unreliable.\n")
  }else {
    if (mean(preFEC * preCF) < mean(postFEC * postCF))
      cat("NOTE: mean of pre-treatment is smaller of post-treatment. Results may be unreliable.\n")
    if (median(preFEC * preCF) < median(postFEC * postCF))
      cat("NOTE: median of pre-treatment is smaller of post-treatment. Results may be unreliable.\n")
  }
```

```
if(outlierdetection==FALSE){
  priors <- fecr_setPrior(muPrior = muPrior, kappaPrior = kappaPrior,
                          deltaPrior = deltaPrior, phiPrior = phiPrior)
}

if(outlierdetection==TRUE){
  if(quant_outlier){
    w <- w_quant(postFEC)
  }
  if(ratio_outlier){
    w <- w_ratio(preFEC, postFEC)
  }

  postmean <- mean(postFEC) # at this stage postFEC are rawcounts
  if(identical(0,postmean)){
    postmean <- 1
  }
  wmovec <- vector(length=postN)
  sumWeightvec <- vector(length=postN)

  for (i in 1:postN){
    if (w[i]<1){
      wmovec[i] <-  postFEC[i]*w[i];
      sumWeightvec[i] <-  w[i];
    } else {
      wmovec[i] <-  0;
      sumWeightvec[i] <-  0;
    }
  }
  wmo <-  sum(wmovec)/sum(sumWeightvec);

  if(is.na(wmo)){
    wmo <- 1
    cat("NOTE: There are no outliers detected.")
  }

  priors <- fecr_setPrior_outlier(muPrior = muPrior, kappaPrior = kappaPrior,
                          deltaPrior = deltaPrior, phiPrior = phiPrior, alphaPrior = alphaPrior)
}

if(outlierdetection==FALSE){
  if (paired & zeroInflation) {
    code <- ZI_paired_stan(priors)
    model <- "Zero-inflated Bayesian model for paired design"
  }
  if (paired & !zeroInflation) {
    code <- paired_stan(priors)
    model <- "Bayesian model without zero-inflation for paired design"
  }
  if (!paired & zeroInflation) {
    code <- ZI_unpaired_stan(priors)
    model <- "Zero-inflated Bayesian model for unpaired design"
  }
  if (!paired & !zeroInflation) {
    code <- unpaired_stan(priors)
    model <- "Bayesian model without zero-inflation for unpaired design"
 }
}

if(outlierdetection==TRUE){
  if (paired & zeroInflation) {
    code <- ZI_paired_outlier_stan(priors)
    model <- "Zero-inflated Bayesian model for paired design with outliers"
  }
  if (paired & !zeroInflation) {
    code <- paired_outlier_stan(priors)
    model <- "Bayesian model without zero-inflation for paired design with outliers"
  }
  if (!paired & zeroInflation) {
    code <- ZI_unpaired_outlier_stan(priors)
    model <- "Zero-inflated Bayesian model for unpaired design with outliers"
  }
  if (!paired & !zeroInflation) {
    code <- unpaired_outlier_stan(priors)
    model <- "Bayesian model without zero-inflation for unpaired design with outliers"
  }
}
```

```
if (paired && preN != postN) {
  stop("post sample size different to pre sample size\n")
}
if (preN == postN) {
  if (sum((preFEC - postFEC)^2) <= .Machine$double.eps)
    cat("Note: the pre-treatment and post-treatment counts are identical\n")
}
if (length(preCF) == 1)
  preCF <- rep(preCF, preN)

if (length(postCF) == 1)
  postCF <- rep(postCF, postN)

if(outlierdetection == FALSE){
  epg_data <- if (paired) {
    list(J = preN, ystarbraw = preFEC, ystararaw = postFEC,
         fpre = preCF, fpost = postCF)
  }else {
  list(Jb = preN, Ja = postN, ystarbraw = preFEC, ystararaw = postFEC,
       fpre = preCF, fpost = postCF)
  }

if (length(setdiff(priors, fecr_setPrior())) == 0) {
  if (paired & zeroInflation) {
    stanModel <- stanmodels$zipaired
  }
  if (paired & !zeroInflation) {
    stanModel <- stanmodels$paired
  }
  if (!paired & zeroInflation) {
    stanModel <- stanmodels$ziunpaired
  }
  if (!paired & !zeroInflation) {
    stanModel <- stanmodels$unpaired
  }
}
else {
  stanModel <- stan_model(model_name = paste(model), model_code = code)
}
}

if(outlierdetection == TRUE){
  epg_data <- if (paired) {
    list(J = preN, ystarbraw = preFEC, ystararaw = postFEC,
         fpre = preCF, fpost = postCF, w=w, wmo=wmo, postmean=postmean)
  }
  else {
    list(Jb = preN, Ja = postN, ystarbraw = preFEC, ystararaw = postFEC,
         fpre = preCF, fpost = postCF, w=w, wmo=wmo, postmean=postmean)
  }
  if (length(setdiff(priors, fecr_setPrior_outlier())) == 0) {  #_outlier#
    if (paired & zeroInflation) {
      stanModel <- stanmodels$zipairedoutlier # add to stanmodels zipairedoutlier.stan
    }
    if (paired & !zeroInflation) {
      stanModel <- stanmodels$pairedoutlier # add to stanmodels pairedoutlier.stan
    }
    if (!paired & zeroInflation) {
      stanModel <- stanmodels$ziunpairedoutlier # add to stanmodels ziunpairedoutlier.stan
    }
    if (!paired & !zeroInflation) {
      stanModel <- stanmodels$unpairedoutlier # add to stanmodels unpairedoutlier.stan
    }
  } else {
    stanModel <- stan_model(model_name = paste(model), model_code = code)
  }
}

 if (verbose) {
  samples <- sampling(stanModel, data = epg_data, iter = nsamples,
                      warmup = nburnin, chains = nchain, thin = thinning,
                      control = list(adapt_delta = adaptdelta), cores = ncore)
}else {
  samples <- suppressMessages(suppressWarnings(sampling(stanModel,
                       data = epg_data, iter = nsamples, warmup = nburnin,
                       chains = nchain, thin = thinning, control = list(adapt_delta = adaptdelta),
                       cores = ncore, refresh = -1)))
}
```

```
  if (paired & !zeroInflation) {
    meanEPG.untreated <- extract(samples, "mu")[[1]]
    meanEPG.treated <- extract(samples, "mu")[[1]] * extract(samples,
                                                             "delta")$delta
    fecr <- 1 - extract(samples, "delta")[[1]]
    result <- cbind(fecr, meanEPG.untreated, meanEPG.treated)
  }
  if (!paired & !zeroInflation) {
    meanEPG.untreated <- extract(samples, "mu")[[1]]
    meanEPG.treated <- extract(samples, "mu")[[1]] * extract(samples,
                                                             "delta")$delta
    fecr <- 1 - extract(samples, "delta")[[1]]
    result <- cbind(fecr, meanEPG.untreated, meanEPG.treated)
  }
  if (paired & zeroInflation) {
    meanEPG.untreated <- extract(samples, "mu")[[1]] * (1 -
                                                          extract(samples, "phi")$phi)
    meanEPG.treated <- extract(samples, "mu")[[1]] * extract(samples,
                                    "delta")$delta * (1 - extract(samples, "phi")$phi)
    fecr <- 1 - extract(samples, "delta")$delta
    result <- cbind(fecr, meanEPG.untreated, meanEPG.treated)
  }
  if (!paired & zeroInflation) {
    meanEPG.untreated <- extract(samples, "mu")[[1]] * (1 -
                                                          extract(samples, "phi")$phi)
    meanEPG.treated <- extract(samples, "mu")[[1]] * extract(samples,
                                     "delta")$delta * (1 - extract(samples, "phi")$phi)
    fecr <- 1 - extract(samples, "delta")$delta
    result <- cbind(fecr, meanEPG.untreated, meanEPG.treated)
  }
  cat("Model: ", model, "\n", "Number of Samples: ", nsamples,
      "\n", "Warm-up samples: ", nburnin, "\n", "Thinning: ",
      thinning, "\n", "Number of Chains", nchain, "\n")
  summarys <- as.data.frame(eggCounts:::printSummary(result))
  return(invisible(list(stan.samples = samples, posterior.summary = summarys)))
}
```

B.2. Additional Functions

fecr_setPrior_outlier()

```
# set default values for the priors

fecr_setPrior_outlier <- function(muPrior, kappaPrior, deltaPrior, phiPrior, alphaPrior){
  if(missing(muPrior)) muPrior = list(priorDist = "gamma",hyperpars=c(1,0.001))
  if(is.null(muPrior[["priorDist", exact = TRUE]])) muPrior$priorDist = "gamma"
  if(is.null(muPrior[["hyperpars", exact = TRUE]])) muPrior$hyperpars = c(1,0.001)

  if(missing(kappaPrior)) kappaPrior = list(priorDist = "gamma",hyperpars=c(1,0.7))
  if(is.null(kappaPrior[["priorDist", exact = TRUE]])) kappaPrior$priorDist = "gamma"
  if(is.null(kappaPrior[["hyperpars", exact = TRUE]])) kappaPrior$hyperpars = c(1,0.7)

  if(missing(deltaPrior)) deltaPrior <- list(priorDist="beta", hyperpars=c(1,1))
  if(is.null(deltaPrior[["priorDist", exact = TRUE]])) deltaPrior$priorDist = "beta"
  if(is.null(deltaPrior[["hyperpars", exact = TRUE]])) deltaPrior$hyperpars = c(1,1)

  if(missing(phiPrior)) phiPrior = list(priorDist = "beta",hyperpars=c(1,1))
  if(is.null(phiPrior[["priorDist", exact = TRUE]])) phiPrior$priorDist = "beta"
  if(is.null(phiPrior[["hyperpars", exact = TRUE]])) phiPrior$hyperpars = c(1,1)

  if(missing(alphaPrior)) alphaPrior = list(priorDist = "normal", hyperpars=c(wmo/postmean,10))
  if(is.null(alphaPrior[["priorDist", exact=TRUE]])) alphaPrior$priorDist = "normal"
  if(is.null(alphaPrior[["hyperpars", exact = TRUE]])) alphaPrior$hyperpars=c(wmo/postmean,10)

  return(list(mu=muPrior,kappa=kappaPrior,delta=deltaPrior, phi = phiPrior, alpha = alphaPrior))
}
```

paired_outlier_stan()

```
paired_outlier_stan <- function(priors){
  #hyperparameters for pre-treatment mean mu
  a.mu <- priors$mu$hyperpars[1]
  b.mu <- priors$mu$hyperpars[2]
  dist.mu <- priors$mu$priorDist
  #hyperparameters  for overdispersion parameter kappa
  a.kappa <- priors$kappa$hyperpars[1]
  b.kappa <- priors$kappa$hyperpars[2]
  dist.kappa <- priors$kappa$priorDist
  #hyperparameters  for change in mean delta
  a.delta <- priors$delta$hyperpars[1]
  b.delta <- priors$delta$hyperpars[2]
  dist.delta <- priors$delta$priorDist
  #hyperparameters for alpha
  a.alpha <- priors$alpha$hyperpars[1]
  b.alpha <- priors$alpha$hyperpars[2]
  dist.alpha <- priors$alpha$priorDist


  paste0('data{
            int J; // number of animals
            int ystararaw[J]; // after treatment McMaster count
            int ystarbraw[J]; // before treatment McMaster count
            int fpre[J];
            int fpost[J];
            real w[J]; // weights
            real wmo;  // weighted mean of the outliers
            real postmean; // mean of the post egg counts
          }
          parameters{
            real<lower=0> kappa;
            real<lower=0> mu;
            real<lower=0,upper=1> delta;
            real<lower=0> mub[J];
            real <lower=1>alpha;
          }
          transformed parameters{
            real lambdaa[J];
            real lambdab[J];
            real kappamu;
            real lambdaaAlpha[J];
            for (i in 1:J){
              lambdab[i] = mub[i]/fpre[i];
            }
            for (i in 1:J){
              lambdaa[i] = delta*mub[i]/fpost[i];
            }
            kappamu = kappa/mu;
            for (i in 1:J){
              lambdaaAlpha[i] = alpha*lambdaa[i];
            }
          }
          model{
            mu ~ ',dist.mu,'(',a.mu,',',b.mu,');            // prior
            kappa ~ ',dist.kappa,'(',a.kappa,',',b.kappa,');
            delta ~ ',dist.delta,'(',a.delta,',',b.delta,');
            alpha ~ ',dist.alpha,'(',a.alpha,',',b.alpha,') T[1,];
            target += gamma_lpdf(mub | kappa,kappamu);   // likelihoods
            ystarbraw ~ poisson(lambdab);
            for (n in 1:J){
              target += w[n]*poisson_lpmf(ystararaw[n] | lambdaa[n]) +
                        (1-w[n])*poisson_lpmf(ystararaw[n] | lambdaaAlpha[n] );
            }
          }
    ')
}
```

unpaired_outlier_stan()

```
unpaired_outlier_stan <- function(priors){
  #hyperparameters for pre-treatment mean mu
  a.mu <- priors$mu$hyperpars[1]
  b.mu <- priors$mu$hyperpars[2]
  dist.mu <- priors$mu$priorDist
  #hyperparameters  for overdispersion parameter kappa
  a.kappa <- priors$kappa$hyperpars[1]
  b.kappa <- priors$kappa$hyperpars[2]
  dist.kappa <- priors$kappa$priorDist
  #hyperparameters  for change in mean delta
  a.delta <- priors$delta$hyperpars[1]
  b.delta <- priors$delta$hyperpars[2]
  dist.delta <- priors$delta$priorDist
  #hyperparameters for alpha
  a.alpha <- priors$alpha$hyperpars[1]
  b.alpha <- priors$alpha$hyperpars[2]
  dist.alpha <- priors$alpha$priorDist

  paste0('data{
          int Ja; // number of animals
          int Jb;
          int ystararaw[Ja]; // after treatment McMaster count
          int ystarbraw[Jb]; // before treatment McMaster count
          int fpre[Ja];
          int fpost[Jb];
          real w[Ja]; // weights
          real wmo;   // weighted mean of the outliers
          real postmean; // mean of the post egg counts
          }
          parameters{
            real<lower=0> kappa;
            real<lower=0> mu;
            real<lower=0, upper = 1> delta;
            real<lower=0> mub[Jb]; # true epg before treatment
            real<lower=0> mua[Ja]; # true epg after treatment
            real <lower=1>alpha;
          }
          transformed parameters{
            real lambdaa[Ja];
            real lambdab[Jb];
            real kappamu;
            real lambdaaAlpha[Ja];
            for (i in 1:Jb){
              lambdab[i] = mub[i]/fpre[i];
            }
            for (i in 1:Ja){
              lambdaa[i] = delta*mua[i]/fpost[i];
            }
            kappamu = kappa/mu;
            for (i in 1:Ja){
              lambdaaAlpha[i] = alpha*lambdaa[i];
            }
          }
          model{
            mu ~ ',dist.mu,'(',a.mu,',',b.mu,');            // prior
            kappa ~ ',dist.kappa,'(',a.kappa,',',b.kappa,');
            delta ~ ',dist.delta,'(',a.delta,',',b.delta,');
            alpha ~ ',dist.alpha,'(',a.alpha,',',b.alpha,') T[1,];
            target += gamma_lpdf(mub | kappa,kappamu)+gamma_lpdf(mua | kappa,kappamu);   // likelihoods
            ystarbraw ~ poisson(lambdab);
            for (n in 1:Ja){
              target += w[n]*poisson_lpmf(ystararaw[n] | lambdaa[n]) +
                        (1-w[n])*poisson_lpmf(ystararaw[n] | lambdaaAlpha[n] );
            }
          }
  ')
}
```

ZI_paired_outlier_stan()

```
ZI_paired_stan <- function(priors){
  #hyperparameters for pre-treatment mean mu
  a.mu <- priors$mu$hyperpars[1]
  b.mu <- priors$mu$hyperpars[2]
  dist.mu <- priors$mu$priorDist
  #hyperparameters  for overdispersion parameter kappa
  a.kappa <- priors$kappa$hyperpars[1]
  b.kappa <- priors$kappa$hyperpars[2]
  dist.kappa <- priors$kappa$priorDist
  #hyperparameters  for change in mean delta
  a.delta <- priors$delta$hyperpars[1]
  b.delta <- priors$delta$hyperpars[2]
  dist.delta <- priors$delta$priorDist
  #hyperparameters  for zero-inflation
  a.phi <- priors$phi$hyperpars[1]
  b.phi <- priors$phi$hyperpars[2]
  dist.phi <- priors$phi$priorDist
  #hyperparameters for alpha
  a.alpha <- priors$alpha$hyperpars[1]
  b.alpha <- priors$alpha$hyperpars[2]
  dist.alpha <- priors$alpha$priorDist

  paste0('data{
            int J; // number of animals
            int ystararaw[J]; // after treatment McMaster count
            int ystarbraw[J]; // before treatment McMaster count
            int fpre[J];
            int fpost[J];
            real w[J]; // weights
            real wmo;  // weighted mean of the outliers
            real postmean; // mean of the post egg counts
          }
          parameters{
            real<lower=0> kappa;
            real<lower=0> mu;
            real<lower=0,upper=1> delta;
            real<lower=0> mub[J];
            real<lower=0,upper=1> phi;
            real <lower=1>alpha;
          }
          transformed parameters{
            real lambdaa[J];
            real lambdab[J];
            real kappamu;
            real lambdaaAlpha[J];
            for (i in 1:J){
              lambdab[i] = mub[i]/fpre[i];
              lambdaa[i] = delta*mub[i]/fpost[i];
            }
            kappamu = kappa/mu;
            for (i in 1:J){
              lambdaaAlpha[i] = alpha*lambdaa[i];
            }
          }
          model{
            mu ~ ',dist.mu,'(',a.mu,',',b.mu,');            // prior
            kappa ~ ',dist.kappa,'(',a.kappa,',',b.kappa,');
            delta ~ ',dist.delta,'(',a.delta,',',b.delta,');
            phi ~ ',dist.phi,'(',a.phi,',',b.phi,');
            alpha ~ ',dist.alpha,'(',a.alpha,',',b.alpha,') T[1,];
            mub ~ gamma(kappa,kappamu);           // likelihoods
            for (n in 1:J) {
              if (ystarbraw[n] == 0)
                target += log_sum_exp(bernoulli_lpmf(1 | phi),
                          bernoulli_lpmf(0 | phi)+poisson_lpmf(ystarbraw[n] | lambdab[n]));
              else
                target += bernoulli_lpmf(0 | phi) + poisson_lpmf(ystarbraw[n] | lambdab[n]);
            }
            for (n in 1:J) {
              if (ystararaw[n] == 0)
                target += log_sum_exp(bernoulli_lpmf(1 | phi),
                          bernoulli_lpmf(0 | phi)+poisson_lpmf(ystararaw[n] | lambdaa[n]));
              else
                target += bernoulli_lpmf(0 | phi) +  w[n]*poisson_lpmf(ystararaw[n] | lambdaa[n])+
                          (1-w[n])*poisson_lpmf(ystararaw[n] | lambdaaAlpha[n] );
            }
          }
  ')
}
```

ZI_unpaired_outlier_stan()

```
ZI_unpaired_outlier_stan <- function(priors){
  #hyperparameters for pre-treatment mean mu
  a.mu <- priors$mu$hyperpars[1]
  b.mu <- priors$mu$hyperpars[2]
  dist.mu <- priors$mu$priorDist
  #hyperparameters  for overdispersion parameter kappa
  a.kappa <- priors$kappa$hyperpars[1]
  b.kappa <- priors$kappa$hyperpars[2]
  dist.kappa <- priors$kappa$priorDist
  #hyperparameters  for change in mean delta
  a.delta <- priors$delta$hyperpars[1]
  b.delta <- priors$delta$hyperpars[2]
  dist.delta <- priors$delta$priorDist
  #hyperparameters  for prevalence
  a.phi <- priors$phi$hyperpars[1]
  b.phi <- priors$phi$hyperpars[2]
  dist.phi <- priors$phi$priorDist
  #hyperparameters for alpha
  a.alpha <- priors$alpha$hyperpars[1]
  b.alpha <- priors$alpha$hyperpars[2]
  dist.alpha <- priors$alpha$priorDist
  paste0('data{
            int Ja; // number of animals
            int Jb;
            int ystararaw[Ja]; // after treatment McMaster count
            int ystarbraw[Jb]; // before treatment McMaster count
            int fpre[Ja];
            int fpost[Jb];
            real w[Ja]; // weights
            real wmo;   // weighted mean of the outliers
            real postmean; // mean of the post egg counts
          }
          parameters{
            real<lower=0> kappa;
            real<lower=0> mu;
            real<lower=0, upper=1> delta;
            real<lower=0> mub[Jb];
            real<lower=0> mua[Ja];
            real<lower=0,upper=1> phi;
            real <lower=1>alpha;
          }
          transformed parameters{
            real lambdaa[Ja];
            real lambdab[Jb];
            real kappamu;
            real lambdaaAlpha[Ja];
            for (i in 1:Jb){
              lambdab[i] = mub[i]/fpre[i];
            }
            for (i in 1:Ja){
              lambdaa[i] = delta*mua[i]/fpost[i];
            }
            kappamu = kappa/mu;
            for (i in 1:Ja){
              lambdaaAlpha[i] = alpha*lambdaa[i];
            }
          }
          model{
            mu ~ ',dist.mu,'(',a.mu,',',b.mu,');            // prior
            kappa ~ ',dist.kappa,'(',a.kappa,',',b.kappa,');
            delta ~ ',dist.delta,'(',a.delta,',',b.delta,');
            phi ~ ',dist.phi,'(',a.phi,',',b.phi,');
            alpha ~ ',dist.alpha,'(',a.alpha,',',b.alpha,') T[1,];
            mub ~ gamma(kappa,kappamu);   // likelihoods
            mua ~ gamma(kappa,kappamu);
            for (n in 1:Jb) {
              if (ystarbraw[n] == 0)
                target += log_sum_exp(bernoulli_lpmf(1 | phi),
                          bernoulli_lpmf(0 | phi)+poisson_lpmf(ystarbraw[n] | lambdab[n]));
              else
                target += bernoulli_lpmf(0 | phi) + poisson_lpmf(ystarbraw[n] | lambdab[n]);
            }
            for (n in 1:Ja) {
              if (ystararaw[n] == 0){
                target += log_sum_exp(bernoulli_lpmf(1 | phi),
                          bernoulli_lpmf(0 | phi)+poisson_lpmf(ystararaw[n] | lambdaa[n]));
              }else{
                target += bernoulli_lpmf(0 | phi) + w[n]*poisson_lpmf(ystararaw[n] | lambdaa[n]) +
                          (1-w[n])*poisson_lpmf(ystararaw[n]| lambdaaAlpha[n]);
              }
            }
          }
  ')
}
```

w_quant()

```
# function to compute w with method quant

w_quant <- function(postFEC){

  n <- length(postFEC)

  # find out if there are outliers:
  postMax <- max(postFEC)
  q <- quantile(postFEC, probs = 0.75) #Q3
  upperLimit <- q + 1.5*IQR(postFEC) # outlier definition of boxplot
  cut <- upperLimit
  higher <- matrix(nrow=n, ncol=1)
  for (i in 1:n){
    if (postFEC[i] > cut){
      higher[i] <- TRUE  # if TRUE: too high, have to leave away for computation of truncated mean
    } else{
      higher[i] <-  FALSE
    }
  }
  nrTruncated <- 0
  sum <- 0
  for (i in 1:n){
    if (higher[i] == FALSE){
      sum <- sum + postFEC[i]
      nrTruncated <- nrTruncated + 1
    }
  }
  truncMean <- sum/nrTruncated
  quant95upper <- qpois(p=0.95, lambda = truncMean)

  #so which ones are outliers?
  outlier <- vector(length = n)
  for (i in 1:n){
    if (postFEC[i] > quant95upper){
      outlier[i] <- 1  # if 1 (=TRUE): outlier
    } else{
      outlier[i] <-  0
    }
  }

  #weights:

  # distance from quant95upper
  a <- integer()
  a <- which(outlier==1) # which index are outliers
  maxdist <- postMax - quant95upper # 100%
  dist <- postFEC[a]-quant95upper
  distPerc <- dist/maxdist
  w <- rep(1,n)
  w[a] <- 1-distPerc+0.01
  return(w)
}
```

w_ratio()

```
# function to compute w with method ratio

w_ratio <- function(preFEC, postFEC){

  n <- length(preFEC)

  ratios <- postFEC/preFEC
  notNr <- which(is.na(ratios))
  infinite <- which(is.infinite(ratios))

  outlier <- vector(length=n)
  index <- seq(1:n)

  if(length(notNr) != 0){
    for(i in notNr){
      outlier[i] <- 0 # no outlier (for ex 0/0)
    }
  }
  if(length(infinite) != 0){
    for(i in infinite){
      outlier[i] <- 1 # is outlier (for ex 5/0)
    }
  }

  out <- sort(c(notNr,infinite))
  index <- seq(1:n)
  index <- index[-out]

  for(i in index){
    if(ratios[i] < 1){
      outlier[i] <- 0
    }else{
      outlier[i] <- 1
    }
  }

  #weights:
  maxratios <- max(ratios[-out])

  w <- rep(1,n)
  w[infinite] <- 0.01

  for(i in index){
    if(ratios[i] >= 1){
      w[i] <- 1/ratios[i]
    }
  }
  return(w)
}
```

C. R Documentation for `fecr_stan_outlier()`

This appendix contains the R Documentation for the new function `fecr_stan_outlier()`. This file is added to the reference manual for the eggCounts package in R.

Modelling the reduction of faecal egg count data considering outliers

Description

Models the reduction in faecal egg counts data with a (un)paired (zero-inflated) Poisson-gamma model formulation using the Stan modelling language. In addition it detects outliers and mitigates its effects on the estimates. As modelled with Stan, it is computationally several-fold faster compare to conventional MCMC techniques. For the installation instruction of Stan, please read: Stan Installation.

Usage

```
fecr_stan_outlier(preFEC, postFEC,rawCounts = FALSE, preCF = 50,
  postCF = preCF, paired = TRUE, zeroInflation=TRUE, outlierdetection = TRUE, outlier = c("quant","ratio"),
  muPrior, kappaPrior, deltaPrior, phiPrior, alphaPrior,
  nsamples = 12000, nburnin = 2000, thinning = 1, nchain = 1,
  ncore = 1, adaptdelta = 0.9, verbose = FALSE)
```

Arguments

`preFEC`	vector of pre-treatment faecal egg counts
`postFEC`	vector of post-treatment faecal egg counts
`rawCounts`	logical. If true, `preFEC` and `postFEC` correspond to raw counts (as counted on equipment). Otherwise they correspond to calculated epgs (raw counts times correction factor). Defaults to `FALSE`.
`preCF`	correction factor(s) before treatment
`postCF`	correction factor(s) after treatment
`paired`	logical. If true, uses the model for the paired design. Otherwise uses the model for the unpaired design
`zeroInflation`	logical. If true, uses the model with zero-inflation. Otherwise uses the model without zero-inflation
`outlierdetection`	logical. If true, outliers are searched. Otherwise does the same as fecr_stan()
`outlier`	the outlier definition that is used, can be either "quant" or "ratio", with a default of "quant"
`muPrior`	a list with hyper-prior information for the baseline mean parameter μ. The default prior is `list(priorDist = "gamma",hyperpars=c(1,0.001))`, i.e. a gamma distribution with shape 1 and rate 0.001, its 90% probability mass lies between 51 and 2996
`kappaPrior`	a list with hyper-prior information for the dispersion parameter κ. The default prior is `list(priorDist = "gamma",hyperpars=c(1,0.7))`, i.e. a gamma distribution with shape 1 and rate 0.7, its 90% probability mass lies between 0.1 and 4.3 with a median of 1
`deltaPrior`	a list with hyper-prior information for the reduction δ. The default prior is `list(priorDist = "beta",hyperpars=c(1,1))`
`phiPrior`	a list with hyper-prior information for the zero-inflation parameter φ. The default prior is `list(priorDist = "beta",hyperpars=c(1,1))`
`alphaPrior`	a list with hyper-prior information for the scaling factor α. The default prior is `list(priorDist = "truncated normal",hyperpars=c(wmo/postmean,100))`, i.e. a normal distribution which is left-truncaed at 1 with mean following from the weighted mean of the outliers divided by the mean of the post values and a variance of 100
`nsamples`	a positive integer specifying how many iterations for each chain (including burn-in samples)
`nburnin`	number of burn-in samples
`thinning`	thinning parameter, a positive integer specifying the period for saving samples
`nchain`	a positive integer specifying the number of chains
`ncore`	number of cores to use when executing the chains in parallel
`adaptdelta`	the target acceptance rate, a value between 0 and 1
`verbose`	logical. If true, prints progress and debugging information

Details

The first time each model with non-default priors is applied, it can take up to 20 seconds for stan to compile the model. Currently the function only support prior distributions with two parameters. For a complete list of supported priors and their parameterization, please consult the list of distributions in Stan.

Sometimes the function outputs informational message from Stan regarding the Metropolis proposal rejections, this is due to the sampler hit the boundary of the parameter space. For some variables, the boundary point is not supported in the distribution. This is not a concern if there are only a few such warnings.

Value

Prints out the posterior summary of `fecr` as the reduction, `meanEPG.untreated` as the mean faecal egg counts before treatment, and `meanEPG.treated` as the mean faecal egg counts after treatment. The posterior summary contains the mean, standard deviation (sd), 2.5%, 25%, 50%, 75% and 97.5% percentiles, the 95% highest posterior density interval (HPDLow95 and HPDHigh95) and the posterior mode. NOTE: we recommend to use the 95% HPD interval and the mode for further statistical analysis.

The returned value is a list that consists of:

`stan.samples`	An object of S4 class `stanfit` representing the fitted results. For more information, please see the `stanfit-class` in `rstan` reference manual.
`posterior.summary`	A data frame that is the same as the printed posterior summary.

Author(s)

Anja Fallegger, based on work of Craig Wang craig.wang@uzh.ch

See Also

`simData2s` for simulating faecal egg counts data with two samples

Examples

```
## Not run:
## load the sample data as a vector
data(epgs)

## apply zero-infation model to the data vector, looking for outliers using method "quant"
model<-fecr_stan_outlier(epgs[,1],epgs[,2],rawCounts=FALSE,preCF=10,paired=TRUE,zeroInflation=TRUE, outlierdetection=TRUE, outlier="quant")
samples<-stan2mcmc(model$stan.samples)


## End(Not run)
```