PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

"Endpoint-inflated beta-binomial regression for correlated count data"

TESIS PARA OPTAR POR EL GRADO DE MAGÍSTER EN ESTADÍSTICA

Presentado por:

Boris Manuel Fazio Luna

Asesor: Victor Giancarlo Sal Y Rosas Celi

Miembros del jurado:
Dr. Cristian Bayes
Dr. Luis Benites
Dr. Giancarlo Sal y Rosas

Lima, Julio 2019

Agradecimentos

Gracias a:

- Las personas detrás de R, Stan y Libgen. Sin ellxs, realizar este trabajo habría sido enormemente más complicado.
- Los amigos y familiares que me preguntaron de qué trata mi tesis por hacerme caer en la cuenta de que aún soy pésimo comunicando los conocimientos que he adquirido.
- Los saludos espontáneos y la inusual fijación con el vello facial por permitirme descubrir el verdadero rostro de la estadística.

Resumen

El modelo de regresión binomial con inflación en los extremos permite modelar datos de conteo acotados en los que una alta proporción de las observaciones se encuentra en los extremos. Extendemos el modelo considerando una función de enlace de logit ordenado, la cual aprovecha la información de orden implícita en las probabilidades de inflación y exploramos el uso de efectos aleatorios y marginalización para manejar la presencia de observaciones repetidas. Empleamos un conjunto de datos previamente analizado en la literatura mediante un modelo de regresión binomial con inflación en los extremos que emplea el enlace softmax para mostrar el mejor ajuste logrado por nuestro modelo.

Abstract

The endpoint-inflated binomial regression model provides a way of modeling bounded count data with a high proportion of observations at the endpoints. We extended the model by considering an ordered logit link which exploits the natural ordering in the inflation probabilities and explore the utility of random effects and marginalization for dealing with repeated measures. We use a dataset previously analyzed in the literature with an endpoint-inflated binomial regression using a softmax link to show our model achieves an improved fit.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preliminary considerations

Inflated distributions are useful when a distribution which would otherwise be adequate for a given type of data fails to account for an excess number of observations at particular values of the sample space. This situation commonly arises when the recorded measurements assign the same value to observations generated by distinct but unobserved processes; for example, the number of outbreaks of water-associated infectious disease can be modeled using a Poisson distribution, but excess zeros will be recorded from areas with consistent access to safe water sources or which are not exposed to environmental conditions that are favorable to the emergence of disease (Yang, LeJeune, Alsdorf, Lu, Shum and Liang, 2012).

The use of regression models can allow the researcher to identify the circumstances driving each underlying process by examining the associations between covariates and the parameters of the inflated distribution. The first inflated regression model in the literature, proposed by Lambert (1992), used the zero-inflated Poisson (ZIP) distribution and was used to understand the circumstances driving the occurrence of manufacturing defects. Specifically, the outcome of interest was the number of soldering defects in the manufacturing process for components on printed wiring boards. In that situation, zeros could be generated under ideal manufacturing conditions which do not produce defects, but also under suboptimal conditions that lead to an error-prone process but which my still output defect-free products most of the time.

Hall (2000) brought zero inflation into the binomial regression model. In this case, the data were bounded counts corresponding to the number of insects, out of a known initial amount, which remained alive after application of a pesticide. A further extension to inflation in binomial counts has been recently described by Tian, Ma, Zhou and Deng (2015), who proposed the endpoint-inflated binomial (EIB) model. This model can represent bounded counts with inflation at both zero counts and at the maximum possible count, termed "one-inflation" by Deng and Zhang (2015) in reference to the effect of the latter type of inflation when measured in proportion scale. Tian et al. (2015) tested their model on the same dataset used by Hall (2000) and showed an improved fit as measured through both AIC and BIC, although they did not take into account the repeated nature of the measurements.

The present work expands on the model proposed by Tian et al. (2015) by taking into account the natural ordering of the inflation parameters through a link function based on the ordered probit model (Daykin and Moffatt, 2002). We fit the model to the same dataset using

a Bayesian framework and attempt to account for repeated measures through two different approaches: random effects and marginalization.

1.2 Objectives

The overall objective of this thesis is to describe an endpoint-inflated beta-binomial (EIBB) regression model and show its application in a real world problem under a Bayesian paradigm.

Specifically, our goals are as follows:

- Provide a brief literature review of available models for analyzing inflated count data.
- Describe the EIBB distribution and its parametrization as a regression model.
- Implement a flexible framework for our model using R and Stan programming languages.
- Conduct simulation studies to understand model behavior over the space of plausible parameters and at its boundaries.
- Show its use in analyzing a real world dataset, including model diagnostics.

Chapter 2

The endpoint-inflated beta-binomial distribution

In this chapter we show the steps to construct the endpoint-inflated beta-binomial distribution, examine its properties and define the parametrization that will be used through the rest of this paper.

We begin by introducing the simpler and familiar distributions that will be used as our building blocks.

2.1 Binomial distribution

When X indicates the total number of successes in a series of $n \in \mathbb{N}_0^+$ independent dichotomous trials, each with probability $p \in (0, 1)$, we say that it has a binomial distribution. The point mass function of a binomial random variable is

$$f_X(x \mid p; n) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, .., n,$$

with mean and variance given respectively by

$$\mathbb{E}[X] = np, \quad \mathbb{V}[X] = np(1-p). \tag{2.1}$$

The binomial upper bound n will be assumed to be known throughout this paper and excluded from discussions of the distribution's parameters, though the opposite can hold in more general treatments.

2.2 Beta distribution

A random variable $Y \in (0, 1)$ follows a beta distribution with parameters $\alpha, \beta > 0$ if its probability density function is given by

$$f_Y(y \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{\alpha-1}(1-y)^{\beta-1}, \quad y \in (0, 1),$$

with mean $\mathbb{E}[Y] = \alpha/(\alpha+\beta)$ and variance $\mathbb{V}[Y] = \alpha\beta/[(\alpha+\beta)^2(\alpha+\beta+1)]$.

For brevity, the normalizing factor in the above probability density function (pdf) will from this point on be expressed through the equivalent beta function:

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)}. \tag{2.2}$$

In the regression context, which will be developed later, it is more convenient to parametrize the beta distribution in terms of its mean $\mu \in (0, 1)$ and a precision parameter $\phi > 0$, as introduced by Ferrari and Cribari-Neto (2004). The equivalences $\alpha = \mu/\phi$ and $\beta = (1-\mu)/\phi$ result in the corresponding reparametrized probability mass function (pmf):

$$f_Y(y \mid \mu, \phi) = \frac{\left[y^{\mu-\phi}(1-y)^{1-\mu-\phi}\right]^{1/\phi}}{B(\mu/\phi, (1-\mu)/\phi)}, \quad y \in (0, 1),$$

which allows us to express the mean and variance respectively as

$$\mathbb{E}[Y] = \mu, \quad \mathbb{V}[Y] = \mu(1-\mu)\frac{\phi}{\phi+1}.$$

2.3 Beta-binomial distribution

Beta random variables have support over $(0, 1)$, which includes all values that are allowed for the p parameter in a non-degenerate binomial random variable. If repeated measurements on a binomial random variable X are thought to reflect a randomly drawn, beta-distributed $p = Y$, then the marginal distribution of X is beta-binomial. Notationally, the relationship is

$$\begin{aligned} X \mid Y = y &\sim \text{Binomial}(p = y; n) \\ Y &\sim \text{Beta}(\mu, \phi) \\ \Rightarrow X &\sim \text{Beta-binomial}(\mu, \phi; n). \end{aligned}$$

With $B(\alpha, \beta)$ as defined in 2.2, the beta-binomial pmf is

$$f_X(x \mid \mu, \phi; n) = \binom{n}{x} \frac{B(x + \mu/\phi, n - x + (1-\mu)/\phi)}{B(\mu/\phi, (1-\mu)/\phi)},$$

with respective mean and variance

$$\mathbb{E}[X] = n\mu, \quad \mathbb{V}[X] = n\mu(1-\mu)\frac{n\phi+1}{\phi+1}.$$

Comparing the above expressions with those for the binomial model in (2.1), it can be seen that the means take the same form, both being governed by a single centrality parameter on the unit interval. For identical values of μ and p, it can be seen that the beta-binomial variance will exceed that of the binomial by a factor of $(n\phi + 1)/(\phi + 1)$.

2.4 Endpoint-inflated beta-binomial (EIBB) distribution

A random variable Y that follows an EIBB distribution is defined as

$$\begin{aligned} Y \mid Z &\sim \begin{cases} \text{Degenerate}(0) & \text{if } Z = 0, \\ \text{Beta-binomial}(\mu, \phi; n) & \text{if } Z = 1, \\ \text{Degenerate}(n) & \text{if } Z = 2 \end{cases} \\ Z &\sim \text{Categorical}(p^0, p^1, p^2), \end{aligned} \tag{2.3}$$

where Z is a discrete latent variable. The degenerate distributions are used to indicate a deterministic result; for a random variable $X \sim \text{Degenerate}(k)$, its pmf is given by

$$f_X(x|k) = \begin{cases} 1 & \text{for } x = k, \\ 0 & \text{elsewhere.} \end{cases}$$

In order to write the complete pmf for the EIBB distribution, we define $I_c(y)$ to be the indicator function for point c, f_{BB} to be the pmf for a beta-binomial random variable and introduce $Y \sim \text{EIBB}(\mu, \phi, p^0, p^1, p^2; n)$. The EIBB random variable Y then has a pmf given by

$$\begin{aligned} f_Y(y \mid \mu, \phi, p^0, p^1, p^2; n) &= p^0 I_0(y) + p^1 f_{BB}(y \mid \mu, \phi) + p^2 I_n(y) \\ &= \begin{cases} p^0 + p^1 \frac{B(\mu/\phi, n+(1-\mu)/\phi)}{B(\mu/\phi,(1-\mu)/\phi)} & \text{if } y = 0 \\ p^1 \binom{n}{y} \frac{B(y+\mu/\phi, n-y+(1-\mu)/\phi)}{B(\mu/\phi,(1-\mu)/\phi)} & \text{if } y = 1, ..., n-1 \\ p^2 + p^1 \frac{B(n+\mu/\phi,(1-\mu)/\phi)}{B(\mu/\phi,(1-\mu)/\phi)} & \text{if } y = n \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2.4}$$

The mean and variance for this distribution are, respectively,

$$\begin{aligned} \mathbb{E}[Y] &= n(p^1\mu + p^2), \\ \mathbb{V}[Y] &= n^2 \left[p^1 \left(\mu + \frac{\mu(1-\mu)}{n} \frac{(n\phi+1)}{(\phi+1)} \right) + p^2 - (p^1\mu + p^2)^2 \right]. \end{aligned} \tag{2.5}$$

Plots of its pmf for selected parameter values are shown in Figure 2.1.

2.4.1 Alternative stochastic representations

In Tian et al. (2015), the authors show that one can arrive at the Endpoint-inflated binomial distribution through six differentt mechanisms, which they refer to as stochastic representations (SR).

Equation (2.3) is the first of such SRs they present, where first one of three bins is chosen (corresponding to zero inflation, no inflation and maximum inflation) and then values are

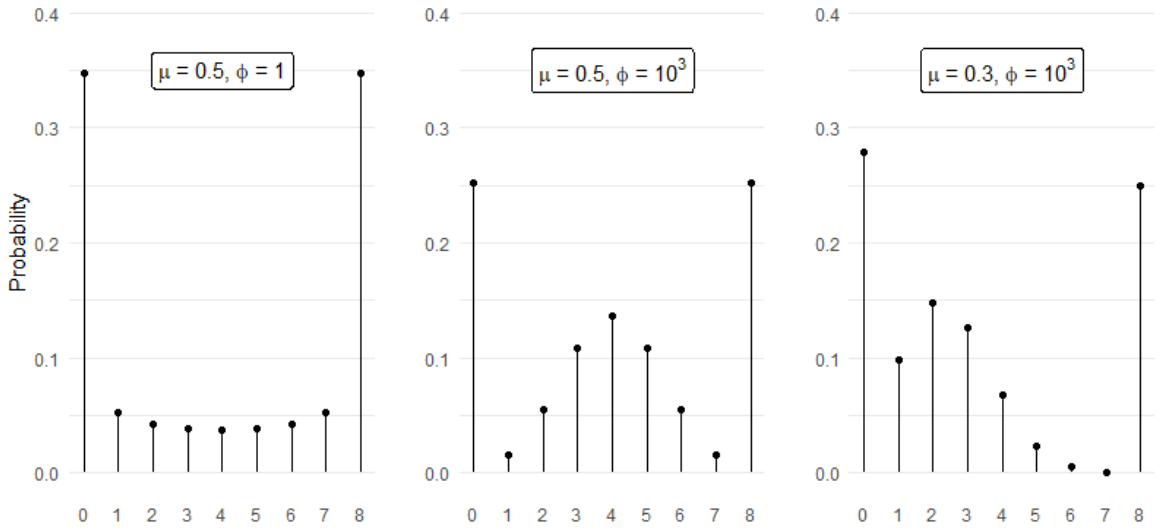

Figure 2.1: The EIBB distribution for three combinations of mean and precision parameters with fixed mixture proportions ($p^0 = p^1 = 0.25$) and $n = 8$.

drawn conditional on the choice: degenerate random variables for either inflation bin or the counting distribution (binomial in the original paper and beta-binomial in our case) for the second bin. We will refer to this representation as SR1.

In another SR, the first step is a binary choice between inflation or no inflation. If inflation is selected, then a second binary choice takes place to determine whether the inflation occurs on the zero or the maximum. Otherwise, the observation is drawn from the counting distribution. We will refer to this representation as SR2:

$$\begin{aligned} X &\sim \text{Beta-binomial}(\mu, \phi; n) \\ Z &\sim \text{Bernoulli}(\nu) \\ W &\sim \text{Bernoulli}(\omega) \\ Y = (1-Z)nW + ZX &\sim \text{EIBB}_{SR2}(\mu, \phi, \eta, \omega; n), \end{aligned}$$

with the corresponding pmf given by

$$f_Y(\mu, \phi, \eta, \omega; n) = (1-\nu)(1-\omega)I_0(y) + \nu f_{BB}(y \mid \mu, \phi) + (1-\nu)\omega I_n(y), \tag{2.6}$$

where ν is the probability of no inflation and ω is the probability of inflation at the maximum (as opposed to zero inflation), given that inflation did occur. The equivalence between (2.4) and (2.6) can be seen by setting

$$
\begin{aligned}
p^0 &= (1-\nu)(1-\omega), \\
p^1 &= \nu, \\
p^2 &= (1-\nu)\omega,
\end{aligned}
$$

which results in the following equations for the mean and variance, respectively:

$$
\begin{aligned}
\mathbb{E}[Y] &= n(\nu\mu + (1-\nu)\omega), \\
\mathbb{V}[X] &= n^2\left[\nu\left(\mu + \frac{\mu(1-\mu)}{n}\frac{(n\phi+1)}{(\phi+1)}\right) + (1-\nu)\omega - (\nu\mu + (1-\nu)\omega)^2\right].
\end{aligned}
$$

We do not consider the rest of SRs shown by Tian et al. (2015), as they are unlikely to represent the process we will be modelling in Section 5. It should be noted that, although the distribution for Y resulting from these representations can be made identical with appropriate choice of parameters, in general they will not lead to equivalent inferences once covariates are introduced. Therefore, the choice of SR should be guided by prior knowledge on the mechanisms which govern the phenomenon under study.

Chapter 3

The endpoint-inflated beta-binomial regression model

In this chapter we introduce our formulation of the EIBB regression model. We will first present the full model likelihood and a set of suggested priors for Bayesian inference. We finish with a brief discussion of applicable model comparison criteria.

3.1 Model definition

Let $Y_i = \begin{pmatrix} Y_{i1} & ... & Y_{iN_i} \end{pmatrix}^\top$, $i = 1, ..., N$ be a set of independent response vectors where

$$Y_i \sim \text{EIBB}(\mu_i, \phi_i, p_{0i}, p_{1i}, p_{2i}; n_i).$$

The following general model structure shows how we can relate covariate values to model parameters:

$$\begin{aligned} \mu_i &= h_1(\mathbf{x}_i^\top \alpha) \\ \phi_i &= h_2(\mathbf{z}_i^\top \beta) \\ \begin{pmatrix} p_{0i}, & p_{1i}, & p_{2i} \end{pmatrix} &= h_3(\mathbf{j}_i^\top \gamma, \mathbf{k}_i^\top \delta) \end{aligned}$$

where $\alpha = [\alpha_1, ..., \alpha_a]^\top$, $\beta = [\beta_1, ..., \beta_b]^\top$, $\gamma = [\gamma_1, ..., \gamma_c]^\top$, and $\delta = [\delta_1, ..., \delta_d]$ are coefficient vectors, each associated with its respective covariate vector given by $\mathbf{x}_i = [x_{i1}, ..., x_{ia}]^\top$, $\mathbf{z}_i = [z_{i1}, ..., z_{ib}]^\top$, $\mathbf{j}_i = [j_{i1}, ..., j_{ic}]^\top$, $\mathbf{k}_i = [k_{i1}, ..., k_{id}]^\top$. Under Bayesian inference, all parameters are random variables so we not consider it necessary to make an explicit distinction between fixed and random effects.

In order to map the linear predictors to parameter space, the link functions must be defined so that

$$\begin{aligned} h_1 &: \mathbb{R} \ \to (0, 1) \\ h_2 &: \mathbb{R} \ \to \mathbb{R}^+ \\ h_3 &: \mathbb{R}^2 \to (0, 1)^3, \end{aligned}$$

with the additional constraint that the entries of h_3 must sum to 1.

In the analyses that follow, we will take the usual selecton of logit for h_1. We will not be working with predictors on the dispersion parameter, but the log function is a common choice that meets the requirements for h_2.

3.1.1 Mixture link for SR1

Under SR1, given in (2.3), mixture proportions are determined as the parameters of a multinomial distribution. As described in Chapter 11 of McElreath (2016), the canonical link function for this distribution is the softmax, which for a K-dimension input vector $\mathbf{x}$ is defined as

$$\text{softmax}(\mathbf{x})_k = \frac{e^{x_k}}{\sum_{j=1}^{K} e^{x_j}}, \tag{3.1}$$

which results in an output vector where all entries are in the $(0, 1)$ range and sum up to 1. The function is invariant under translation by a constant, that is $\text{softmax}(\mathbf{x}) = \text{softmax}(\mathbf{x}+c)$, so an arbitrary component of the input vector is generally given a fixed value of 0.

While the softmax is a reasonable choice for unordered categorical outcomes, mixture probabilities for the EIBB model have a natural ordering. Therefore, we motivate our choice for the link function on the mixture proportions, h_3, by noting that the mixture components can be ordered in terms of their expected values: $0 < n\mu < n$, for the first, second and third components respectively and the latent Z in (2.3) which selects the mixture components can be regarded as an ordered categorical variable. The ordered probit model (see Daykin and Moffatt (2002) for an in-depth treatment) gives us a starting point by which to introduce a linear predictor on its parameters p_0, p_1, p_2.

If $\Phi(.; \mu', \sigma'^2)$ is the cumulative distribution function of a normal with mean μ' and variance σ'^2 and C_0, C_1 are two real numbers such that $C_0 < C_1$, the ordered probit regression model is given by

$$\begin{aligned} p_0 &= \Phi(C_0; \mu', \sigma'^2) \\ p_1 &= \Phi(C_1; \mu', \sigma'^2) - \Phi(C_0; \mu', \sigma'^2) \\ p_2 &= 1 - \Phi(C_1; \mu', \sigma'^2). \end{aligned}$$

Due to the restriction $\sum_s p_s = 1$, we see that the above system relates two free parameters with four unknowns. Therefore some values must be fixed to obtain a unique solution. We choose to set $C_0 = -1$ and $C_1 = 1$. Figure 3.1 illustrates this setup. This allows any combination of p_0, p_1, p_2 to be reached, which retains the flexibility of the softmax link, while offering a more interpretable parametrization for the inflation behavior: a value of $\mu' = 0$ corresponds to symmetric inflation at both endpoints, whereas higher magnitudes indicate that the inflation proportion is comparatively higher at one of the endpoints; meanwhile, the σ' parameter governs the total proportion of inflation, with lower values corresponding to less inflation.

In order to place linear predictors on μ' and σ'^2, we use the identity and log link functions as follows:

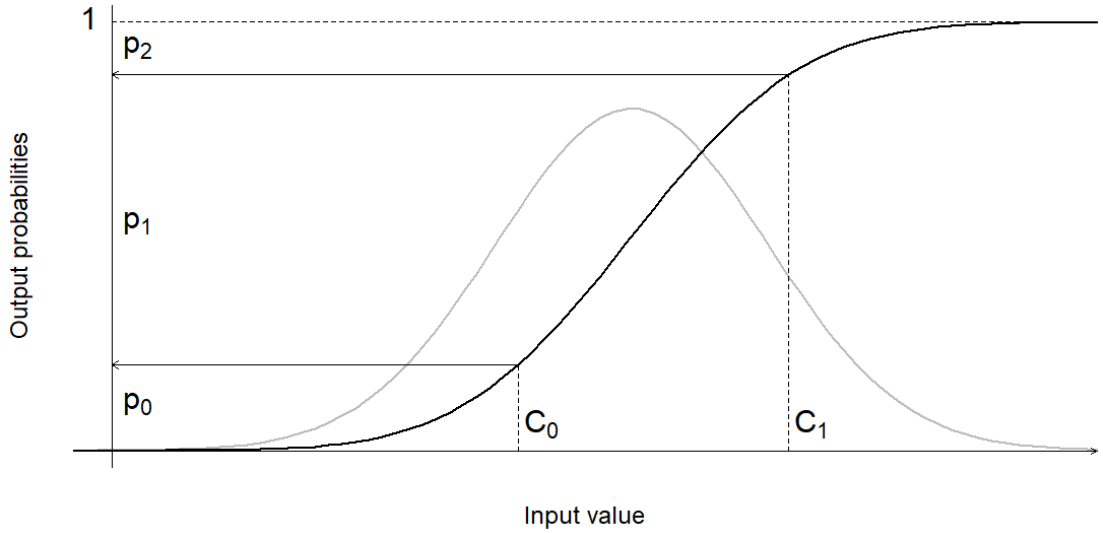

Figure 3.1: The black curve shows a μ', σ'^2 normal cumulative distribution function (cdf) and how the cutpoints C_0, C_1 are used to retrieve the vector of probabilities. The corresponding normal pdf is shown in gray.

$$\mu'_i = \mathbf{j}_i^\top \gamma,$$
$$\sigma'^2_i = log(\mathbf{k}_i^\top \delta),$$

with $\mathbf{j}_i^\top \gamma$ and $\mathbf{k}_i^\top \delta$ defined as in (3.1).

3.1.2 Mixture link for SR2

Under SR2, given in 2.6, we don't attempt to model a simultaneous influence on the proportions p_{0i}, p_{1i}, p_{2i}. Instead, we model one proportion at a time: the probability of inflation, given by $1 - \nu_i$, and the probability of maximum inflation (as opposed to zero inflation) given by ω_i. This leads to a straightforward choice of logit for both parameters:

$$\begin{aligned}
Y_i &\sim \text{EIB}_{SR2}(\mu_i, p_{0i}, p_{1i}, p_{2i}; n_i) \\
p_{0i} &= \nu_i(1 - \omega_i) \\
p_{1i} &= 1 - \nu_i \\
p_{2i} &= \nu_i\omega_i \\
\nu_i &= \text{logit}^{-1}(j_i^\top \gamma) \\
\omega_i &= \text{logit}^{-1}(k_i^\top \delta).
\end{aligned}$$

3.2 Considerations for Bayesian inference

Statistical modeling involves choosing a likelihood function that is thought to approximate the process that generated the observed data. Under Bayesian inference, the model also includes a prior distribution for the likelihood's parameters. As new data arrives, the likelihood governs the way in which the prior is updated to reflect what has been learned about

the model parameters and the updated distribution is known as the posterior. Sampling from the resulting posterior in an efficient way often requires sophisticated computational methods. In this section we provide certain guidelines for Bayesian inference and describe the computational tools we used.

3.2.1 Selection of priors

Priors encode information known before taking into account the data at hand. However, there is often an absence of substantive knowledge about the phenomenon being studied, in which case Gelman, Carlin, Stern, Dunson, Vehtari and Rubin (2014) suggests using weakly informative priors. These restrict the amount of probability placed over mathematically admissible but empirically implausible parameter values. Doing this serves a dual purpose; it is effectively a regularization procedure, which protects against extreme inferences based on limited data, and also provides a computational advantage by constraining the parameter space that must be explored during numerical evaluation.

Some authors suggest that improper priors can be used as a way of "letting the data speak for itself", however, such a choice does not guarantee a proper posterior (Tak and Morris, 2015). Furthermore, diffuse priors are not intrinsically non-informative criteria, as their impact on the posterior depends on the specific parameter under consideration and the form in which it enters the likelihood (Gelman et al., 2017).

3.2.2 The Stan probabilistic programming language

Most prior-likelihood combinations do not result in closed form posteriors. While the posterior can be numerically evaluated at any point over the parameter space, exploring it efficiently requires an algorithm that prioritizes regions of high probability. Markov chain Monte Carlo (MCMC) methods are a class of algorithms that allow exploration of a distribution in a series of discrete steps over parameter space, such that the long run occupancy of states approximates the target distribution.

Stan is a probabilistic programming language which allows the user to specify priors and likelihoods for continuous parameters (Betancourt, 2017). It uses an algorithm described as Hamiltonian Monte Carlo (HMC) to explore target distributions. HMC is based on the equations for physical motion and, conceptually, simulates the movement of a particle through a vector field constructed from the posterior. A detailed description of the implementation is availabe in Betancourt (2017).

We used Stan through the interface provided by the `brms` library for the R programming language (R Core Team, 2013). The `brms` library allows the user to specify models by using established R formula syntax and family specification, based on which it automatically generates and executes Stan code (Burkner, 2018). The R code for the analyses conducted in this document can be found in appendix A.

3.3 Measures for model assessment

3.3.1 Information criteria

Information criteria provide a way to compare the relative predictive prowess of two or more models. Several variants have been proposed, among which Deviance Information

Criterion (DIC), see Spiegelhalter, Best, Carlin and Van Der Linde (2002), and Widely Applicable Information Criterion (WAIC), see Watanabe (2010), enjoy widespread use. The key quantity used to calculate them is the deviance, defined as

$$\mathcal{D}(\theta) = -2\log\{L(\theta \mid Y)\},$$

where L is the likelihood function of the model and θ represents all of its parameters. Given a Montecarlo posterior with S samples θ^s, we define the mean deviance as

$$\bar{\mathcal{D}}(\theta) = \sum_{s=1}^{S} \frac{\mathcal{D}(\theta^s)}{S}.$$

We will also use the deviance at the posterior mean, denoted by $\mathcal{D}(\bar{\theta})$, where

$$\bar{\theta} = \sum_{s=1}^{S} \frac{\theta^s}{S}.$$

Then, DIC is calculated as follows:

$$\begin{aligned} \text{DIC} &= \mathcal{D}(\bar{\theta}) + 2 \times (\bar{\mathcal{D}}(\theta) - \mathcal{D}(\bar{\theta})) \\ &= \mathcal{D}(\bar{\theta}) + 2 \times p_D, \end{aligned}$$

where the p_D term is a penalty known as the effective number of parameters.

WAIC uses the log pointwise predictive density (lppd) to assess predictive performance instead of deviance as well as a different penalty term. The following definition is given in Gelman, Carlin, Stern, Dunson, Vehtari and Rubin (2014):

$$\begin{aligned} \text{WAIC} &= -2[\quad \textstyle\sum_{i=1}^{n} \log \int p(y_i \mid \theta) p(\theta \mid y) p(\theta) d\theta \quad - \quad \sum_{i=1}^{n} \text{Var}(\log(p(y_i \mid \theta))) \quad] \\ &= -2[\qquad\qquad \text{lppd} \qquad\qquad - \qquad\qquad p_W \qquad\qquad]. \end{aligned}$$

DIC and WAIC measure the performance of the model on the same dataset used to fit the model. Models with lower values of the first term indicate better fit and the second term adds a penalty which corresponds to the number of effective parameters, which corrects for overfitting. Both criterions have a share of limitations, however: Gelman, Hwang and Vehtari (2014) show that DIC fails to produce consistent results for general posteriors, while both DIC and WAIC are asymptotic approximations which exhibit high variance when tested on datasets of realistic size, as shown in Vehtari, Gelman and Gabry (2017).

3.3.2 Pareto-smoothed importance sampling leave-one-out (PSIS-LOO)

The utility of WAIC derives from its asymptotic equivalence to cross validation, but alternatives have recently been developed which provide stable estimates at smaller sample sizes, as we will now discuss.

Leave-one-out cross validation (LOOCV) is a technique that approximates cross validation by refitting the model while holding out one observation at a time and using the held out

observation as the test case. However, performing LOOCV can incur a high computational cost when models are complex or sample sizes are large.

Vehtari et al. (2017) reviews approaches for estimating LOOCV results through approximations based on the single fit which produces the whole-dataset posterior. The approach they propose weighs the contribution of each data point y_i and posterior sample θ^s to the likelihood through an importance ratio, defined as follows:

$$r_i^s = \frac{1}{p(y_i \mid \theta^s)} = \frac{p(\theta^s \mid y_{-i})}{p(\theta^s \mid y)}.$$

The equality holds if the data is exchangeable conditional on parameter values. A definition of exchangeability due to Bernardo (1996), describes exchangeable observations as providing the same information regardless of the order in which they are collected. Formally, for exchangeability to hold, one must have

$$p(x_1, ..., x_n) = p(x_{\pi(1)}, ...x_{\pi(n)}),$$

for every permutation of the values of π in the set $\{1, ..., n\}$. When this holds, one can separate each observation's contribution to the likelihood into a product:

$$\begin{aligned}
\frac{p(\theta^s \mid y_{-i})}{p(\theta^s \mid y)} &= \frac{p(y_{-i} \mid \theta^s)p(\theta^s)}{p(y \mid \theta^s)p(\theta^s)} \\
&= \frac{p(y_1 \mid \theta^s) \times ... \times p(y_{i-1} \mid \theta^s) \times p(y_{i+1} \mid \theta^s) \times ... \times p(y_n \mid \theta^s)}{p(y_1 \mid \theta^s) \times ... \times p(y_i \mid \theta^s) \times ... \times p(y_n \mid \theta^s)} \\
&= \frac{1}{p(y_i \mid \theta^s)}.
\end{aligned}$$

Note that this requirement implies that this approximation cannot be used without the ability to readily express the likelihood as n separate components, which can make implementation difficult for models with complex data structures, such as time series or spatial data (Gelman, Carlin, Stern, Dunson, Vehtari and Rubin, 2014).

If $\tilde{y_i}$ denotes an arbitrary observation which was not used to fit the model, Gelfand et al. (1992) showed that the predictive distribution can be approximated by using the importance ratios:

$$p(\tilde{y_i} \mid y_{-i}) \approx \frac{\sum_{s=1}^{S} r_i^s p(\tilde{y_i} \mid \theta^s)}{\sum_{s=1}^{S} r_i^s}.$$

In practice, the above equation is evaluated by replacing y_i with held-out observations y_i. However, using the raw importance ratios r_i^s directly can lead to estimates with high or even infinite variance. Pareto smoothing is the procedure whereby the largest raw importance ratios are replaced by more stable smoothed weights, w_i^s. Vehtari et al. (2017) describes the process in three steps:

1. A generalized Pareto distribution is fit using the 20% largest raw importance ratios,

once for each held-out observation i.

2. Define M as the number of raw importance ratios that were used to fit the Pareto. Then, the M largest raw ratios are replaced by the expected value of the order statistics of the fitted distribution,

$$\tilde{w}_m = F^{-1}\left(\frac{m - 1/2}{M}\right), \quad m = 1, ..., M,$$

where F^{-1} is the fitted Pareto cdf.

3. Finally, take the average of the smoothed weights,

$$\bar{w}_i = \sum_{m=1}^{M} \tilde{w}_m / M,$$

and truncate all the smoothed weights $\tilde{w}_m$ so that they do not exceed the value of $\bar{w}_i \times S^{3/4}$, where S is the total number of posterior draws available; note that this quantity was proposed by Vehtari et al. (2017) and chosen based purely on empirical considerations. The resulting set of weights are labeled as w_i.

Now, the smoothed and truncated weights can be used for a more stable approximation of the predictive density,

$$\hat{\text{elpd}}_{\text{psis-loo}} = \sum_{i=1}^{n} \log\left(\frac{\sum_{s=1}^{S} w_i^s p(y_i \mid \theta^s)}{\sum_{s=1}^{S} w_i^s}\right)$$

and this concludes the PSIS-LOO procedure.

Multiplying the above value by -2 converts the estimate to deviance scale, setting it to a scale that allows direct comparison with WAIC. Such a transformation results in the criterion known as LOOIC. For large sample sizes, LOOIC and WAIC achieve remarkably similar values.

It is shown in Vehtari et al. (2017) that the method described above exhibits a more robust behavior in small samples or under weak prior information and it is specifically shown that the resulting estimates are more stable than WAIC.

The PSIS-LOO procedure is implemented in the `loo` R package and it will be used for model selection in the sections that follow.

Chapter 4

Simulation study

In this chapter, we explore the conditions under which the full model is likely to be useful by simulating data from an EIBB distribution parameters close to their boundary values. We compare the model against simpler alternatives by considering the following factors: whether the model converges at all, the time it takes to do so, the bias in its estimated regression coefficients and predictive performance as evaluated by the PSIS-LOO criterion, introduced in Section 3.3.

4.1 Simulation setup

We compared the binomial and beta-binomial models in order to determine the minimal level of overdispersion, governed by ϕ, that can be reliably detected over a range of sample sizes, starting at $n = 100$. The same was done for binomial and endpoint-inflated binomial models to find the minimal level of reliably detectable inflation. In all cases, a linear predictor is placed on the mean of the binomial components to simultaneously explore the effects on parameter recovery. The linear predictor features two covariates drawn from a $N(0, 0.5)$ distribution with zero intercept and coefficient vector $\beta = [0.5, -0.5]^\top$.

One hundred datasets were simulated for each combination of parameter values, sample size and generating distribution. Then, each of those datasets was fit to the candidate models: a binomial one and a beta-binomial or endpoint-inflated binomial, respectively. The posterior means for each regression coefficient where compared against the true coefficient values to calculate bias, root mean square error (RMSE), the time taken for model fitting was recorded and, if the model converged, LOOIC was calculated. All of these quantities are then averaged for presentation in a summary table. As LOOIC can be badly biased in the presence of models which failed to converge, we only calculate it if all the simulations converged; the number of simulations where the model did not converge can be found under the "Failed to converge" column of the summary tables.

4.2 Results

4.2.1 Binomial vs Beta-binomial

In this simulation, observations were drawn from

$$Y \sim \text{Beta-binomial}(0.5X_1 - 0.5X_2, \phi; 7)$$

where X_1 and X_2 were generated independently from a $N(0, 0.5)$ distribution for each of $\phi = 0.05, 0.1, 0.2, 0.5$. Each dataset was then fit to binomial and beta-binomial distributions.

Table 4.1 shows that even with moderately large samples, fitting issues for the beta-binomial disappear only when the dispersion parameter is large enough ($\phi = 0.2$). Furthermore, the beta-binomial model is slower to fit by orders of magnitude and only approaches the binomial in speed for very large dispersion values. It is interesting to note that even if the PSIS-LOO criterion correctly identifies the beta-binomial as the correct model for the data, the coefficients are in general retrieved with a smaller bias by means of the binomial distribution. In conclusion, it seems that ignoring dispersion values below 0.1 may be safe and even beneficial, depending on the target of inference.

			Bias		RMSE		Failed to			Time
ϕ	n	Model	β_1	β_2	β_1	β_2	converge	LOOIC	Std. Err.	(seconds)
0.05	100	Binomial	0.00	0.03	0.23	0.24	0	368.96	15.92	1.22
0.05	100	Beta-binomial	0.26	0.23	0.89	0.82	18	–	–	614.82
0.05	500	Binomial	-0.00	0.00	0.1	0.1	0	1816.22	35.14	5.80
0.05	500	Beta-binomial	0.12	0.01	0.39	0.42	8	–	–	1269.84
0.05	1000	Binomial	-0.00	-0.00	0.07	0.07	0	3631.94	49.90	11.64
0.05	1000	Beta-binomial	0.03	-0.02	0.17	0.32	5	–	–	2274.36
0.1	100	Binomial	-0.01	0.02	0.25	0.24	0	388.03	18.04	1.22
0.1	100	Beta-binomial	0.07	0.02	0.38	0.39	7	–	–	281.13
0.1	500	Binomial	0.00	0.00	0.1	0.1	0	1950.37	41.33	5.95
0.1	500	Beta-binomial	-0.01	0.26	0.22	0.18	1	–	–	983.04
0.2	100	Binomial	-0.02	-0.03	0.26	0.27	0	438.47	22.80	1.23
0.2	100	Beta-binomial	-0.46	0.49	0.81	0.96	0	371.41	8.80	250.12
0.5	100	Binomial	0.02	0.04	0.28	0.28	0	533.82	28.92	1.20
0.5	100	Beta-binomial	0.02	0.03	0.35	0.36	0	412.85	6.50	2.73

Table 4.1: Leave One Out criterion and fitting time for the binomial and beta-binomial models tested on simulated data; one hundred datasets where simulated for each pair of comparisons.

4.2.2 Binomial vs. Endpoint-inflated binomial

For this simulation, observations were drawn from

$$Y \sim \text{EIBB}(0.5X_1 - 0.5X_2, p, 1 - 2p, p; 7)$$

where X_1 and X_2 were generated independently from a $N(0, 0.5)$ distribution for each of $p = 0.025, 0.05, 0.0625, 0.075$. Each dataset was then fit to binomial and endpoint-inflated binomial distributions.

Table 4.2 shows the results of testing the endpoint-inflated model. As it can be seen for all n, comparatively modest increases in sample size rapidly allow the model to retrieve the parameters without issue. Additionally, the time difference between both methods is not large, even if the binomial is cleared favored for its simplicity.

			Bias		RMSE		Failed to			Time
ϕ	n	Model	β_1	β_2	β_1	β_2	converge	LOOIC	Std. Err.	(seconds)
0.05	100	Binomial	0.01	-0.09	0.25	0.24	0	411.47	24.35	1.25
0.05	100	EI Binomial	-0.00	-0.00	0.25	0.25	40	–	–	4.05
0.05	250	Binomial	0.01	-0.04	0.15	0.16	0	1004.38	37.85	2.96
0.05	250	EI Binomial	-0.00	0.01	0.16	0.17	4	–	–	7.44
0.05	400	Binomial	-0.00	-0.04	0.11	0.11	0	1603.68	47.59	4.61
0.05	400	EI Binomial	-0.00	0.02	0.12	0.12	0	1487.96	24.01	11.84
0.1	100	Binomial	-0.00	-0.13	0.28	0.26	0	470.51	29.48	1.22
0.1	100	EI Binomial	-0.01	-0.04	0.31	0.27	6	–	–	3.04
0.1	200	Binomial	0.01	-0.13	0.19	0.18	0	938.92	41.84	2.38
0.1	200	EI Binomial	-0.00	-0.01	0.22	0.21	0	776.31	14.01	5.72
0.125	100	Binomial	-0.02	-0.07	0.28	0.29	0	497.31	31.52	1.17
0.125	100	EI Binomial	-0.00	0.07	0.27	0.32	2	–	–	2.90
0.125	150	Binomial	0.00	-0.13	0.22	0.22	0	751.97	38.67	1.78
0.125	150	EI Binomial	-0.00	0.00	0.25	0.25	0	586.34	11.45	4.25
0.15	100	Binomial	-0.03	-0.16	0.29	0.29	0	528.92	33.16	1.17
0.15	100	EI Binomial	0.00	0.00	0.31	0.30	0	389.12	9.04	2.85

Table 4.2: Leave One Out criterion and fitting time for the binomial and endpoint-inflated binomial models tested on simulated data; one hundred datasets where simulated for each pair of comparisons.

Chapter 5

Application

In this chapter, we explore the inferences obtained by applying the EIBB regression model to data from a horticultural experiment on pesticide efficacy. We then discuss the model's merits in relation to the alternatives.

5.1 The whitefly dataset

In van Iersel et al. (2000), a randomized complete block experiment with repeated measures was conducted for the purpose of evaluating the efficacy of six treatment conditions for the control of silverleaf whiteflies on poinsettia plants. Each week, a known number of insects was placed on one of the plant's leaves and the number of survivors was recorded after treatment, resulting in a bounded count outcome and a total of 640 observations. The six conditions which were tested are a control with no pesticide, pesticide application through subirrigation preceded by a 0, 1, 2 and 4-day no-irrigation period and application via hand watering.

The resulting dataset contains the following variables:

- `week`: marks the week in which the measurement was taken; integers 1–12
- `rep`: identifier for the randomization block; integers 1–3
- `trt`: indicates which treatment was assigned; integers 0–5
- `bindenom`: number of insects at the beginning of the week; integers 1–15
- `nlive`: number of surviving insects at the end of the week, equal or less than `bindenom`; integers 0–13
- `plantid`: identifier for each plant; integers 1–54

In Figure 5.1, we see that the proportion of surviving insects shows a clear concentration of probability at both bounds, with zeros (no survivors) accounting for 53% of the observations and ones (no deaths) accounting for 12% of the observations. This suggests a case where our endpoint inflated model could be useful and the presence of repeated measures means that clustering effects should also be evaluated.

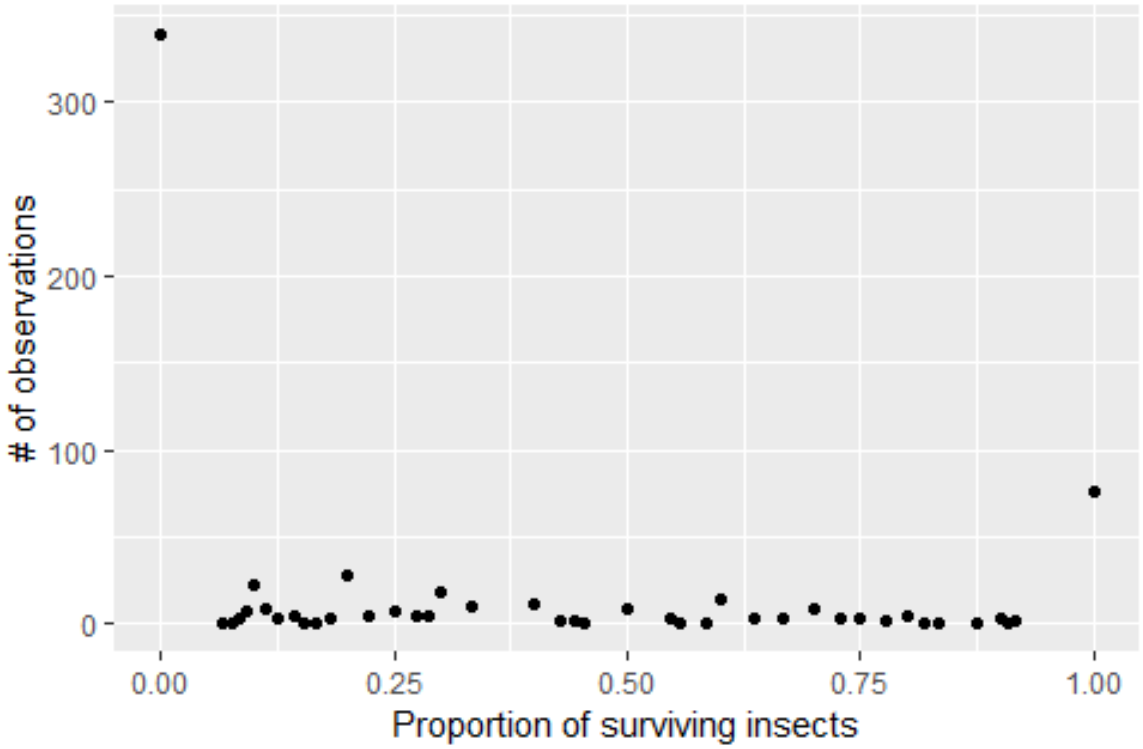

Figure 5.1: Empirical distribution of proportion of survived insects

5.2 Model structure

Two analyses of this dataset with inflated models have already been published , see Hall (2000) and Hall and Zhang (2004), but these only accounted for zero inflation. We will follow the same model specification used by the authors for the mean of the binomial component of the mixture and compare whether the introduction of inflation at the upper bound results in improved fit.

We fit the following models: binomial, zero-inflated binomial with and without fixed effects on inflation probability, and endpoint-inflated binomial with SR1 using softmax and normal cdf links and with SR2.

Taking the same model specification from Hall and Zhang (2004), the linear predictor with coefficient vector β at observation i as $\eta_{\beta i}$ for the binomial mean takes the following form:

$$\eta_{\beta i} = \beta_0 + \beta_{week} week_i + \beta_{block2} block2_i + \beta_{block3} block3_i + \beta_{trt2} trt2_i + \\ + \beta_{trt3} trt3_i + \beta_{trt4} trt4_i + \beta_{trt5} trt5_i + \beta_{trt6} trt6_i,$$

where the *trt* and *block* variables are generated by dummy coding the six treatments and block membership, respectively. Because all the parameters on which we will place linear predictors have an influence on the mean of the distribution (see 2.5), we initially attempted to include all the variables as predictors on all other parameters, so that $\eta_{\beta i} = \eta_{\gamma i} = \eta_{\delta i}$.

The link functions we use reach values of zero or one rapidly for relatively small input

values, we will place a $N(0, 5)$ prior for all coefficients in every model considered below; this places little probability on values with implausibly large effect sizes and, as mentioned in 3.2, it also helps the fitting process to be more efficient.

Among the models we will consider, we include logistic regression as a minimal benchmark. In this model, the observations Y_i follow a simple binomial distribution with no inflation:

$$\begin{aligned} Y_i &\sim \text{Binomial}(\mu_i; n_i) \\ \mu_i &= \text{logit}^{-1}(\eta_{\beta i}). \end{aligned}$$

We also fit the model used by Hall and Zhang (2004), which uses a zero-inflated binomial distribution and extend it to include predictors on p_0, the probability of zero-inflation:

$$\begin{aligned} Y_i &\sim \text{ZIB}(\mu_i, p_{0i}; n_i) \\ \mu_i &= \text{logit}^{-1}(\eta_{\beta i}) \\ p_{0i} &= \text{logit}^{-1}(\eta_{\gamma i}). \end{aligned}$$

The first model we consider for endpoint inflation, given by Tian et al. (2015), uses SR1 and the softmax link to place predictors directly on the inflation probabilities:

$$\begin{aligned} Y_i &\sim \text{EIB}_{SR1}(\mu_i, p_{0i}, p_{1i}, p_{2i}; n_i) \\ \mu_i &= \text{logit}^{-1}(\eta_{\beta i}) \\ \begin{pmatrix} p_{0i}, & p_{1i}, & p_{2i} \end{pmatrix} &= \text{softmax}(\eta_{\gamma i}, 0, \eta_{\delta i}). \end{aligned}$$

The second variant we consider is our own and also uses SR1 but links inflation probabilities to predicts through an ordered probit link:

$$\begin{aligned} Y_i &\sim \text{EIB}_{SR1}(\mu_i, p_{0i}, p_{1i}, p_{2i}; n_i) \\ \mu_i &= \text{logit}^{-1}(\eta_{\beta i}) \\ p_{0i} &= \Phi(-1; \eta_{\gamma i}, \sigma^2) \\ p_{1i} &= \Phi(1; \eta_{\gamma i}, \sigma^2) - \Phi(-1; \eta_{\gamma i}, \sigma^2) \\ p_{2i} &= 1 - \Phi(1; \eta_{\gamma i}, \sigma^2). \end{aligned}$$

We additionally consider SR2, where the probability of inflation at either endpoint, ν_i, and the mean of the inflation by both endpoints, ω_i, are connected to covariates through a logit link:

$$
\begin{aligned}
Y_i &\sim \text{EIB}_{SR2}(\mu_i, p_{0i}, p_{1i}, p_{2i}; n_i) \\
\mu_i &= \text{logit}^{-1}(\eta_{\beta i}) \\
\nu_i &= \text{logit}^{-1}(\eta_{\delta i}) \\
\omega_i &= \text{logit}^{-1}(\eta_{\gamma i}) \\
p_{0i} &= \nu_i(1-\omega_i) \\
p_{1i} &= 1-\nu_i \\
p_{2i} &= \nu_i\omega_i.
\end{aligned}
$$

Although the different specifications we use for the endpoint-inflated models are all equally capable of modeling any combination of inflation probabilities, the way in which they relate covariates and inflation can offer insight into the underlying processes being studied. The ordinal probit model, through the normal cdf mean parameter, models a situation where the amount of inflation at one endpoint is directly in opposition with the other one and is best suited to represent a smooth transitions from one endpoint to the other; any exchange of probability between either endpoint is first allocated to the binomial distribution, which can be thought of as representing an intermediate state of response intensity (see Figure 3.1). On the other hand, the SR2 model is most useful when one seeks to explain the presence of inflation at either endpoint; this can be useful if the inflated responses are thought to come from a qualitatively different population and one wishes to separate them from the results obtained from the binomial distribution.

In order to account for the repeated measures, we first selected the best-fitting model from the above set using the LOO criterion and then refit two versions of the model, one using random intercepts and another with the beta-binomial distribution.

5.3 Results

5.3.1 Model comparison

The original Hall and Zhang (2004) article did not place any predictors on the inflation parameters. We first explored whether doing so could improve model fit. As shown in table 5.1, including the treatment variable as a predictor on the zero-inflation probability leads to a non-trivial improvement in the LOO criterion (2186.59 to 2055.67). On the other hand, the week or block variables do not improve model fit, and this is reflected by a worsening of the LOO criterion (2055.67 to 2060.11). Based on this result, we only use the treatment variable for the inflation parameters on all models that follow.

Model	LOOIC	Std. Err.
Fixed zero probability	2186.59	97.68
Treatment on zero probability	2055.67	99.55
Treatment + week + block on zero probability	2060.11	99.47

Table 5.1: Comparison of LOO criterion for three different specifications of the zero-inflation probability in the ZIB model

Table 5.2 shows that all inflated models clearly outperform a naive binomial model, however, the difference in performance between the zero-inflated model and endpoint-inflated

models is small in general, with the exception of the model using a normal cdf link, which shows a sizable improvement.

Model	LOOIC	Std. Err.
Binomial	2774.60	138.59
Zero-inflated binomial	2055.67	99.55
Endpoint-inflated binomial (SR1 softmax)	2020.72	92.28
Endpoint-inflated binomial (SR1 normal cdf)	1878.42	78.95
Endpoint-inflated binomial (SR2)	2025.42	95.17

Table 5.2: Comparison of LOO criterion for determining the best-fitting model

In the SR1 softmax model, we were only able to include covariate information on the zero-inflation parameter as models with predictors for one-inflation did not converge. This was likely due to the small fraction of observations in that category.

Finally, we attempted to model the potential overdispersion due to repeated measures by using a random intercept approach and a marginal approach using a beta-binomial distribution instead of the binomial. While the beta-binomial approach does not directly model a correlation within units, the results in Hall and Zhang (2004) show that such correlation is quite low. Therefore, a beta-binomial model could be more computationally efficient as it can account for overdipersion with only one additional parameter, while the random intercept model would introduce 54 additional parameters, one for each plant in the experiment. Ultimately, however, the beta-binomial model failed to converge, as seen in the trace plots of Figure 5.2. Therefore, the endpoint-inflated binomial model with mixed effects and SR1 normal cdf link appears to reflect the best fit for this dataset.

The good fit of our model can be further shown by using a posterior predictive plot, which simulates random draws from the model at each datapoint. The plots in Figure 5.3 show that our selected model is the only one which adequately models the behavior of observations at the upper bound.

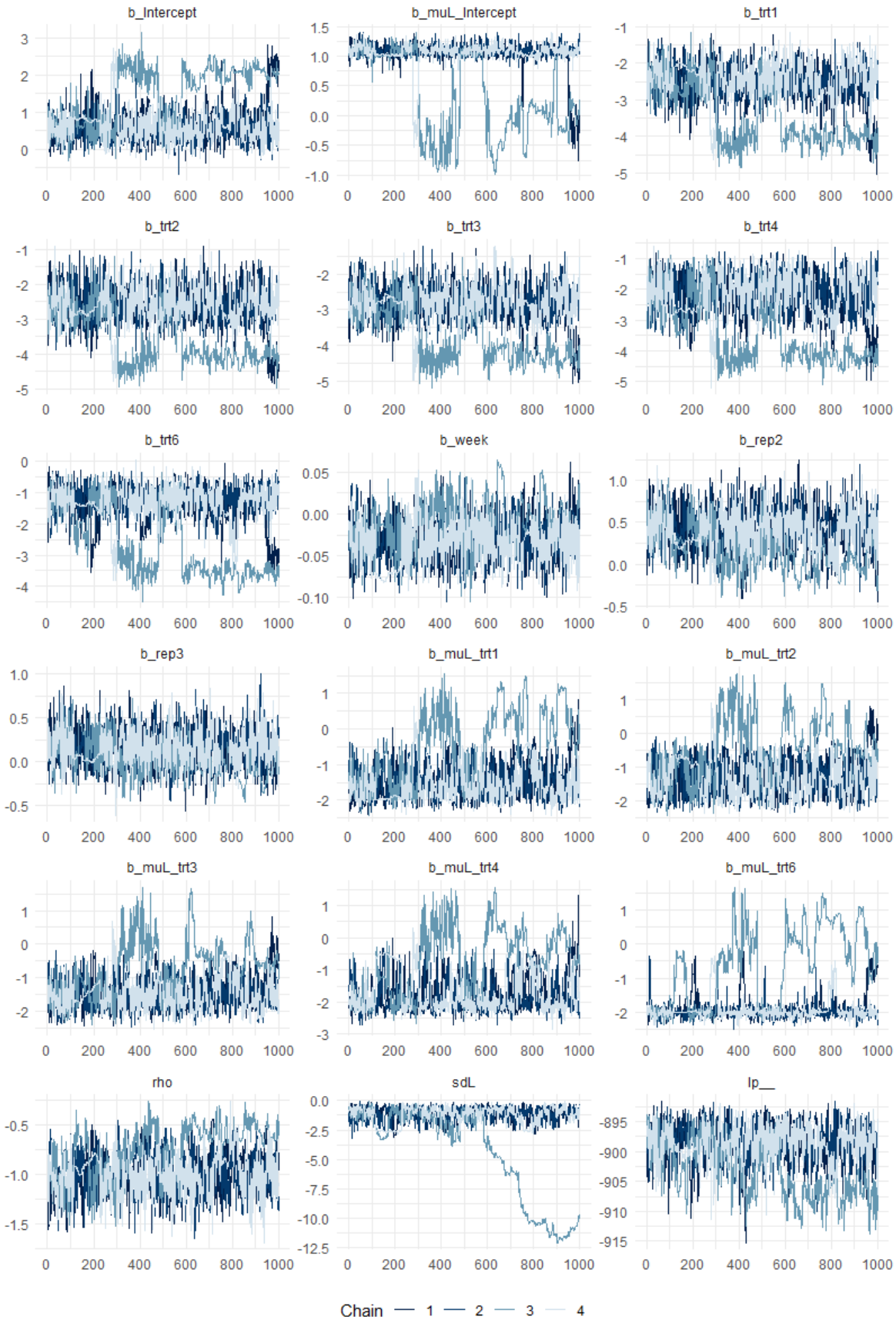

Figure 5.2: Trace plots for a beta-binomial model fit to the whitefly dataset

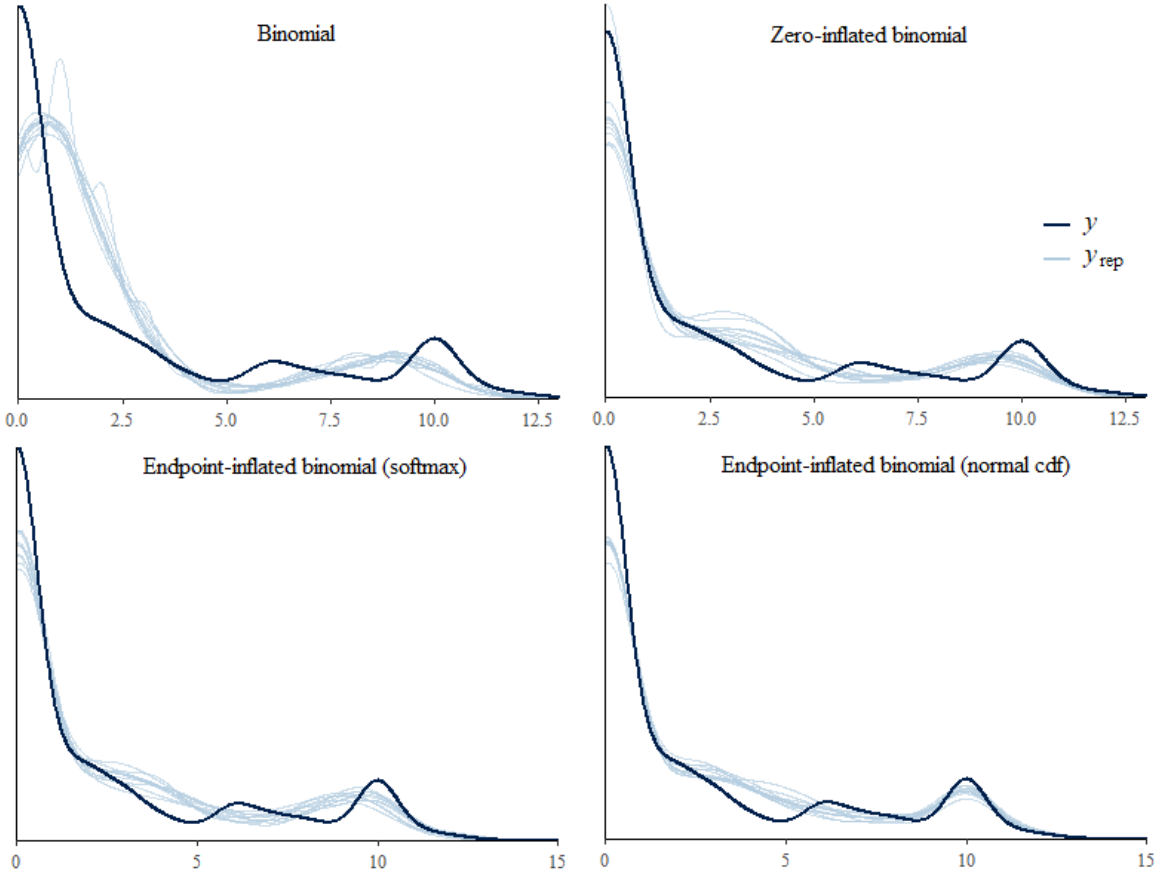

Figure 5.3: Posterior predictive graphs comparing four models fit on the whitefly dataset

5.3.2 Model interpretation

As the dataset used comes from an experimental study, its main purpose is providing information on the effectiveness of the treatments. In a traditional regression model, one may do so by directly interpreting the magnitude and sign of the coefficients in the linear predictor. However, this becomes harder to do in models with multiple regressions on its different parameters, all of which may be simultaneously affecting the mean response (as is the case in our model). Therefore, we resort to marginal effects plots, which allow us to graphically observe the overall effect of the variable of interest on the response.

Marginal plots are created by calculating the mean response for varying levels of a given variable while every other variable is set to a fixed value: numeric variables are set to their mean value across the sample while categorical variables to their reference level; in this case the graph plot shows varying treatment levels with $week = 6.5$ and $block2 = block3 = 0$.

The resulting plot, shown in Figure 5.4, indicates that there is little difference among the inferences produced by the models, including the traditional binomial regression, although the latter underestimates the uncertainty aroun the estimates. Overall, one can conclude that any treatment has an effect which is clearly distinguishable from the control, and among treatments subirrigation is slightly better than hand watering but they are otherwise similar.

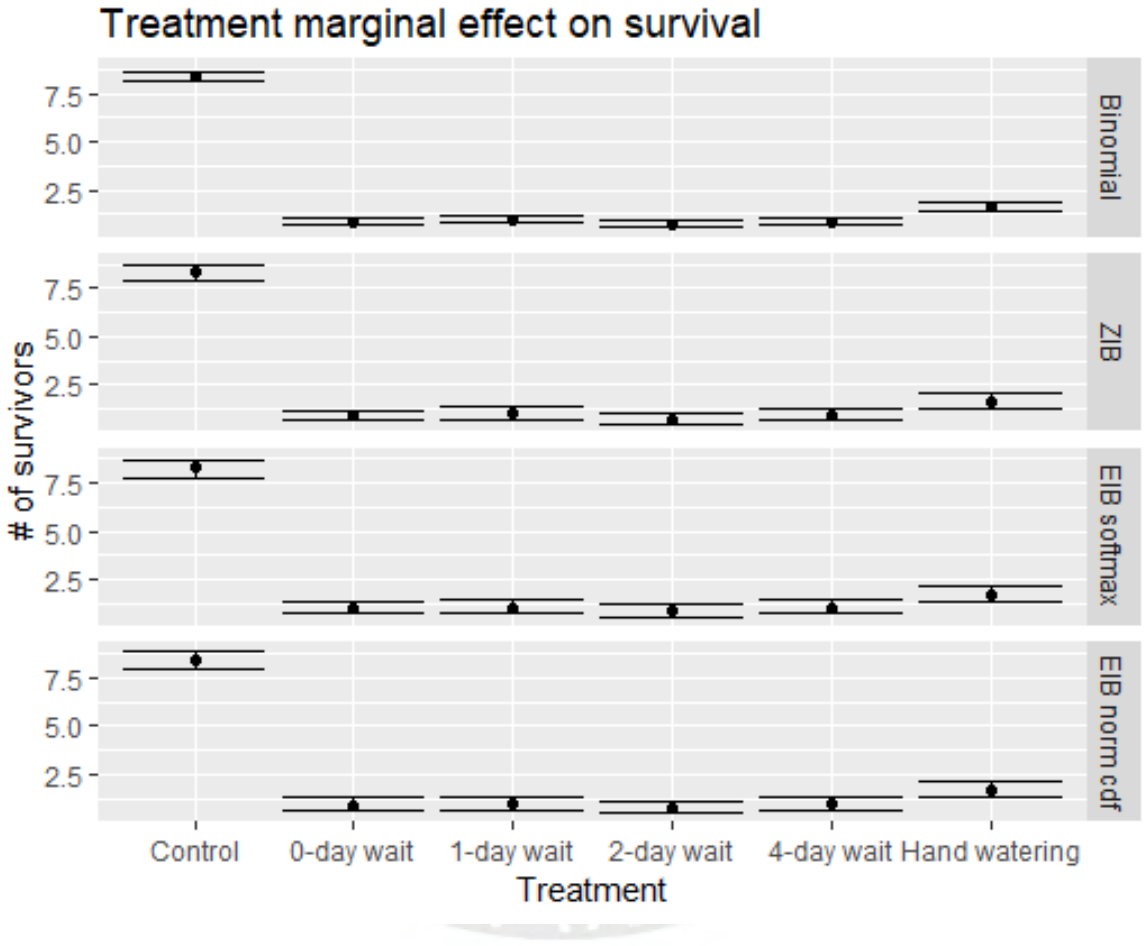

Figure 5.4: Marginal effects of treatment on pest survival under four different models

Chapter 6

Conclusions

Our simulation study shows that exploring the presence of inflation using MCMC can be relatively inexpensive in terms of computation time and does not require a very large sample size. On the other hand, models that handle overdispersion through marginalization were shown to be far slower to fit and required a larger sample size before reliable estimates could be obtained. Our analysis with the whitefly dataset reinforces our findings as the beta-binomial model could not be fit on the data, despite having repeated measures and sufficient sample size to reliably estimate the rest of parameters.

A crucial lesson to be learned from the whitefly dataset is that even when endpoint inflation is clearly an appropriate modeling choice, it is not guaranteed to meaningfully improve model fit unless an appropriate link function and predictor specification is used. In particular, we discovered that the softmax, despite being a common link choice for categorical distributions, was outperformed by a normal cdf link.

We anticipate that the normal cdf link could be a generally appropriate choice for endpoint inflated data due to its taking advantage of the order information present in the mixture. Because predictors placed on the cdf's mean parameter simultaneously affect all three mixture probabilities, this removes the need for an arbitrary choice of which probabilities to model introduced when using a softmax link. Furthermore, as estimation of the cdf mean parameter uses the entire mixture data, it is less affected by situations where there is a limited number of observations at the endpoints where attempts to individually estimate each probability would be unreliable.

Appendix A

Code

A.1 Simulation study

```
### SETUP
library(drake)
library(tidyverse)
library(bayesplot)
library(rstan)
library(loo)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
options(stringsAsFactors = FALSE)
pkgconfig::set_config("drake::strings_in_dots" = "literals")

# Inverse logistic
invlogit <- function(x)(1/(1+exp(-x)))

# Softmax
softmax <- function(x)exp(x)/sum(exp(x))

# Beta distribution mean-dispersion reparametrizer
repar.beta <- function(mu, rho) {
  if (all(0 < mu & mu < 1 & 0 < rho)) {
    list(alpha = mu / rho,
         beta = (1 - mu) / rho)
  } else {
    stop("Parameters are out of bounds")
  }
}

# Draw from reparametrized beta
rbeta.repar <- function(n, mu, rho) {
    with(repar.beta(mu, rho),
       rbeta(n, alpha, beta))
```

```
}

# Draw from beta-binomial (uses reparametrized beta)
rbbin <- function(n = 1000,  # samples
                  tries = 10,# binomial attempts
                  mu = 0.5, rho = 1) {
  rbinom(n, tries, rbeta.repar(n, mu, rho))
}

# Label variables have a 'labels' attribute
putlabel <- function(x) {
  if(is.null(attr(x, "labels"))) {
    stop("No 'labels' attribute")
  } else {
    return(factor(x, levels = attr(x, "labels"),
              labels = names(attr(x, "labels"))))
  }
}

# Cumulative normal-based probability vector
cumu.norm <- function(mu, s){
  c(
    pnorm( (0 - mu)/s ),
    pnorm( (1 - mu)/s ) - pnorm( (0 - mu)/s ),
    1 - pnorm( (1 - mu)/s )
  ) %>% matrix(ncol = 3)
}

# Simulate data for EIBB regression model estimation
eibb.sim <- function(N,      # Number of observations
                     n,      # Binomial size
                     bx = 0, bz = 0.5,    # Coeffs (beta, latent normal means)
                     rho = 0.0001, s = 0, # Default: no overdispersion/inflation
                     sx = 0.5, sz = 0.3,  # Dispersion for covariate generation
                     seed = 1, # Random seed
                     fullinfo = FALSE) {

    # Covariate generation
  Kx <- length(bx)
  x <- cbind(rep(1, N),
             sapply(rep(N, Kx - 1), rnorm, sd = sx)) %>%
 unlist %>% matrix(ncol = Kx)
```

```
  Kz <- length(bz)
  z <- cbind(rep(1, N),
             sapply(rep(N, Kz - 1), rnorm, sd = sz)) %>%
 unlist %>% matrix(ncol = Kz)

    # Linear predictors
  mu.beta <- invlogit(x %*% bx)
  mu.norm <- z %*% bz
    # Mixture probabilities
  p <- cumu.norm(mu.norm, s)

    # Latent mixture component selector
  Z <- apply(p, 1, function(prob.vector) sample(1:3, size = 1, prob = prob.vector))

  y <- ifelse(Z == 1, 0,
              ifelse(Z == 2, rbbin(N, n, mu.beta, rho),
                     ifelse(Z == 3, n, NA))
              )

  if (fullinfo) {
    list(N = N, Kx = Kx, Kz = Kz,
         n = ifelse(length(n) == 1, rep(n, N), n),
         y = y,
         x = x, z = z,
         p = p,
         Z = Z,
         mu.norm = mu.norm,
         mu.beta = mu.beta) %>% return
  } else {
    list(N = N, Kx = Kx, Kz = Kz,
         n = if(length(n)==1){rep(n,N)}else{n},
         y = y,
         x = x, z = z) %>% return
  }
}

tab_looic_divergent <- function(fit){
  data.frame(model = attr(fit, "model_name"),
             N = attr(fit, "sim")$dims_oi$log_lik,
             looic = loo(fit)$estimates[3,1],
             waic = waic(extract(fit, "log_lik")$log_lik)$estimates[3,1],
             divergent = get_num_divergent(fit))
```

```
}

# Endpoint-inflated binomial pmf/cdf
deibi <- function(y, mu, p1, p2, p3, n) {
  if(p1+p2+p3!=1){stop("p elements must sum to 1")}
  p1*ifelse(y==0, 1, 0) +
    p2*dbinom(y, n, mu) +
    p3*ifelse(y==n, 1, 0)
}

peibi <- function(y, mu, p1, p2, p3, n) {
  pacote <- data.frame(y, mu, p1, p2, p3, n)
  apply(pacote, 1,
        function(x) sum(
          deibi(0:x[[1]], x[[2]], x[[3]], x[[4]], x[[5]], x[[6]], x[[7]])
          ))
}
# Endpoint-inflated beta-binomial pmf/cdf
deibb <- function(y, mu, rho, p1, p2, p3, n) {
  if(p1+p2+p3!=1){stop("p elements must sum to 1")}
  p1*ifelse(y==0, 1, 0) +
    p2*rmutil::dbetabinom(y, n, mu, 1/rho) +
    p3*ifelse(y==n, 1, 0)
}

peibb <- function(y, mu, rho, p1, p2, p3, n) {
  pacote <- data.frame(y, mu, rho, p1, p2, p3, n)
  apply(pacote, 1,
        function(x) sum(
          deibb(0:x[[1]], x[[2]], x[[3]], x[[4]], x[[5]], x[[6]], x[[7]])
          ))
}

# Create divergence table from simulation fits
diverg.tb <- function(...) {
  ddd <- list(...)
  simu.label <- strsplit(
    c(
      names(ddd)
      ),"_") %>%
    sapply(function(x){x[3:6]}) %>%
    data.frame(row.names = c("model", "data", "pars", "n")) %>%
```

```
    t %>% as.tibble %>%
    mutate(n = as.numeric(substr(n, 2, 99)))
  as.tibble(cbind(simu.label, divergent = unlist(ddd)))
}

diverg.tb2 <- function(...) {
  ddd <- list(...)
  simu.label <- strsplit(
    c(
      names(ddd)
      ),"_") %>%
    sapply(function(x){x[1:4]}) %>%
    data.frame(row.names = c("model", "data", "pars", "n")) %>%
    t %>% as.tibble %>%
    mutate(n = as.numeric(substr(n, 2, 99)))
  as.tibble(cbind(simu.label, divergent = unlist(ddd)))
}

### COMPILE MODELS
compilestan_plan <- drake_plan(
  binom.model =
      stan_model(file_in("Stan/bin-regression-model.stan"),
                 model_name = "Binomial regression"),
  betab.model =
      stan_model(file_in("Stan/bb-regression-model.stan"),
                 model_name = "BB regression"),
  binRE.model =
      stan_model(file_in("Stan/binRE-regression-model.stan"),
                 model_name = "Binomial regression with RE"),
  binRE_zero.model =
      stan_model(file_in("Stan/binRE_zero-regression-model.stan"),
                 model_name = "Binomial regression with zero-mean RE"),
  eibin.model =
      stan_model(file_in("Stan/eibi-regression-model.stan"),
                 model_name = "EIBi regression"),
  eibeb.model =
      stan_model(file_in("Stan/eibb-regression-model.stan"),
                 model_name = "EIBB regression"),
  eibiP.model =
      stan_model(file_in("Stan/eibi-regression-model-P.stan"),
                 model_name = "EIBi regression - P parametrization")
)
```

```
### SIMULATIONS
seeds_plan <- data.frame(
  target =
    paste0("seed", sprintf("%03d", 1:100)),
  command =
    as.character(1:100),
  stringsAsFactors = FALSE
  )

# Binomial vs Beta-Binomial simulations
simu_betab_pars <- drake_plan(
  simu.betab.data_0.05_n100 =
    eibb.sim(N = 10**2, n = 7,
             bx = c(0, 0.5, -0.5),
             rho = 0.05,
             seed = seed__),
  simu.betab.data_0.05_n500 =
    eibb.sim(N = 5*10**2, n = 7,
             bx = c(0, 0.5, -0.5),
             rho = 0.05,
             seed = seed__),
  simu.betab.data_0.05_n1k =
    eibb.sim(N = 10**3, n = 7,
             bx = c(0, 0.5, -0.5),
             rho = 0.05,
             seed = seed__),
  simu.betab.data_0.1_n100 =
    eibb.sim(N = 10**2, n = 7,
             bx = c(0, 0.5, -0.5),
             rho = 0.1,
             seed = seed__),
  simu.betab.data_0.1_n500 =
    eibb.sim(N = 5*10**2, n = 7,
             bx = c(0, 0.5, -0.5),
             rho = 0.1,
             seed = seed__),
  simu.betab.data_0.2_n100 =
    eibb.sim(N = 10**2, n = 7,
             bx = c(0, 0.5, -0.5),
             rho = 0.2,
             seed = seed__),
```

```
  # Evaluar sesgo en coeficientes
  simu.betab.data_0.5_n100 =
    eibb.sim(N = 10**2, n = 7,
             bx = c(0, 0.5, -0.5),
             rho = 0.5,
             seed = seed__)
)

simu_betab_plan <- evaluate_plan(
  plan = simu_betab_pars,
  wildcard = "seed__",
  values = seeds_plan$target
)

simu_betab_fits <- drake_plan(
  sampling(binom.model,
           data = data__,
           chains = 1, iter = 2000),
  sampling(betab.model,
           data = data__,
           chains = 1, iter = 2000,
            control =
            list(adapt_delta = 0.9,
                 max_treedepth = 15
            ))
)

simu_betab.fits_plan <- evaluate_plan(
  plan = simu_betab_fits,
  wildcard = "data__",
  values = simu_betab_plan$target
)

# Binomial vs Endpoint-Inflated Binomial simulations
simu_eibin_pars <- drake_plan(
  simu.eibin.data_p0.05_n100 =
    eibb.sim(N = 10**2, n = 7,
             bx = c(0, 0.5, -0.5),
             s = 0.3039784,
             seed = seed__),
  simu.eibin.data_p0.05_n250 =
    eibb.sim(N = 2.5*10**2, n = 7,
```

```
            bx = c(0, 0.5, -0.5),
            s = 0.3039784,
            seed = seed__),
  simu.eibin.data_p0.05_n400 =
    eibb.sim(N = 4*10**2, n = 7,
            bx = c(0, 0.5, -0.5),
            s = 0.3039784,
            seed = seed__),
  simu.eibin.data_p0.1_n100 =
    eibb.sim(N = 10**2, n = 7,
            bx = c(0, 0.5, -0.5),
            s = 0.3901521,
            seed = seed__),
  simu.eibin.data_p0.1_n200 =
    eibb.sim(N = 2*10**2, n = 7,
            bx = c(0, 0.5, -0.5),
            s = 0.3901521,
            seed = seed__),
  simu.eibin.data_p0.125_n100 =
    eibb.sim(N = 10**2, n = 7,
            bx = c(0, 0.5, -0.5),
            s = 0.4346506,
            seed = seed__),
  simu.eibin.data_p0.125_n150 =
    eibb.sim(N = 1.5*10**2, n = 7,
            bx = c(0, 0.5, -0.5),
            s = 0.4346506,
            seed = seed__),
  simu.eibin.data_p0.15_n100 =
    eibb.sim(N = 10**2, n = 7,
            bx = c(0, 0.5, -0.5),
            s = 0.4824237,
            seed = seed__)
)

simu_eibin_plan <- evaluate_plan(
  plan = simu_eibin_pars,
  wildcard = "seed__",
  values = seeds_plan$target
)

simu_eibin_fits <- drake_plan(
```

```
  sampling(binom.model,
           data = data__,
           chains = 1, iter = 2000),
  sampling(eibin.model,
           data = data__,
           chains = 1, iter = 2000,
            control =
             list(adapt_delta = 0.9,
                  max_treedepth = 15
             ))
)

simu_eibin.fits_plan <- evaluate_plan(
  plan = simu_eibin_fits,
  wildcard = "data__",
  values = simu_eibin_plan$target
)

simu_binRE_fits <- drake_plan(
  binREfit =
    sampling(binRE.model, data = data__, chains = 1, iter = 2000,
             control = list(adapt_delta = 0.9, max_treedepth = 15))
)

simu_binRE.fits_plan <- evaluate_plan(
  plan = simu_binRE_fits,
  wildcard = "data__",
  values = simu_betab_plan$target
)

simu_eibiP_fits <- drake_plan(
  eibiPfit =
    sampling(eibiP.model, data = data__, chains = 1, iter = 2000,
             control = list(adapt_delta = 0.9, max_treedepth = 15))
)

simu_eibiP.fits_plan <- evaluate_plan(
  plan = simu_eibiP_fits,
  wildcard = "data__",
  values = simu_eibin_plan$target[c(201:300,401:500,601:700,701:800)]
)
```

```
# All sims
simu_plan <- bind_plans(
  simu_betab_plan,
  simu_betab.fits_plan,
  simu_eibin_plan,
  simu_eibin.fits_plan,
  simu_eibiP.fits_plan,
  simu_binRE.fits_plan
)

betab_dnumbs <- tibble(
  target = paste0(simu_betab.fits_plan$target,"_divergences"),
  command = paste("get_num_divergent(",simu_betab.fits_plan$target,")")
)

betab_divergences <- gather_plan(
  betab_dnumbs,
  target = "betab_ndiv",
  gather = "diverg.tb"
)

eibin_dnumbs <- tibble(
  target = paste0(simu_eibin.fits_plan$target,"_divergences"),
  command = paste("get_num_divergent(",simu_eibin.fits_plan$target,")")
)

eibin_divergences <- gather_plan(
  eibin_dnumbs,
  target = "eibin_ndiv",
  gather = "diverg.tb"
)

eibiP_dnumbs <- tibble(
  target = paste0(simu_eibiP.fits_plan$target,"_divergences"),
  command = paste("get_num_divergent(",simu_eibiP.fits_plan$target,")")
)

eibiP_divergences <- gather_plan(
  eibiP_dnumbs,
  target = "eibiP_ndiv",
  gather = "diverg.tb2"
```

```
)

divergent_plan <- bind_plans(
  betab_dnumbs,
  betab_divergences,
  eibin_dnumbs,
  eibin_divergences,
  eibiP_dnumbs,
  eibiP_divergences
)

### RUN ALL
plan_final <- bind_plans(
  compilestan_plan,
  seeds_plan,
  simu_plan,
  divergent_plan
)

make(plan_final, seed = 1991)
```

A.2 Whitefly dataset analysis

```
### SETUP
library(tidyverse)
library(brms)

# Convenience functions
  # Paste operator
  # Saves space when writing folder paths
`%p0%` <- function(x, y)paste0(x, y)

  # Fit & save model
save_fit <- function(fitname, fitfun, formula_, ...) {
  if(file.exists("out/fits/" %p0% fitname %p0% ".rds"))stop("File already exists")
  saveRDS(object = fitfun(formula_, ...),
          file = "out/fits/" %p0% fitname %p0% ".rds")
}

  # Matrix columns to list
col2list <- function(x)as.list(unname(as.data.frame(x)))

  # Split every n rows
split_nrow <- function(x, n)lapply(1:n, function(y)x[seq(y, nrow(x), by = n),])
```

```
# Math functions
softmax <- function(x)(exp(x)/sum(exp(x)))
softmax2 <- function(x, y)mapply(function(...)softmax(c(...)), x, y, 0)
cutnorm <- function(mu, sd) {
  p1 <- pnorm(-1, mean = mu, sd = sd)
  p2 <- pnorm(1, mean = mu, sd = sd)
  c(p1,
    1 - p2,
    p2 - p1)
}
cutnorm2 <- function(mu, sd)mapply(cutnorm, mu, sd)

### DEFINE BRMS MODELS
# Fit a Zero-Inflated Binomial model with constant inflation probability
fit_bin <- function(formula_, ...){
  priors <- set_prior("normal(0,5)", class = "b") # weak regularization
  brm(formula = formula_,
      data = wf,
      prior = priors,
      family = zero_inflated_binomial())
}

# Fit a Zero-Inflated Binomial model with constant inflation probability
fit_zib <- function(formula_, ...){
  priors <- set_prior("normal(0,5)", class = "b") # weak regularization
  brm(formula = formula_,
      data = wf,
      prior = priors,
      family = zero_inflated_binomial())
}

# Fit an Endpoint-Inflated Binomial model w/softmax mixture link
fit_eiBinomialSM <- function(formula_, ...){
  eiBinomialSM <- custom_family(
    "eiBinomialSM", dpars = c("mu", "so", "sm"), # unrestricted mixture scores
    links = c("identity", "identity", "identity"),
    type = "int", vars = "trials[n]",
    log_lik =
      function(i, draws) {
        mu <- if(is.null(dim(draws$dpars$mu))) {
          draws$dpars$mu
```

```
    } else{
      draws$dpars$mu[, i]
    }
    so <- if(is.null(dim(draws$dpars$so))) {
      draws$dpars$so
    } else{
      draws$dpars$so[, i]
    }
    sm <- if(is.null(dim(draws$dpars$sm))) {
      draws$dpars$sm
    } else{
      draws$dpars$sm[, i]
    }
    trials <- draws$data$trials[i]
    y <- draws$data$Y[i]
    eiBinomialSM_lpmf(y, mu, so, sm, trials)
  },
predict =
  function(i, draws, ...) {
    mu <- if(is.null(dim(draws$dpars$mu))) {
      draws$dpars$mu
    } else{
      draws$dpars$mu[, i]
    }
    so <- if(is.null(dim(draws$dpars$so))) {
      draws$dpars$so
    } else{
      draws$dpars$so[, i]
    }
    sm <- if(is.null(dim(draws$dpars$sm))) {
      draws$dpars$sm
    } else{
      draws$dpars$sm[, i]
    }
    trials <- draws$data$trials[i]
    eiBinomialSM_rng(mu, so, sm, trials)
  },
fitted =
  function(draws) {
    mu <- draws$dpars$mu
    so <- draws$dpars$so
    sm <- draws$dpars$sm
```

```
      p <- split_nrow(mapply(softmax2, so, sm), 3)[-1]
      trials <- median(draws$data$trials)

      (p[[1]] + p[[2]]*inv_logit_scaled(mu))*trials
    }

)

stan_funs <- "
vector linkfunc(real so, real sm){
return softmax([so, sm, 0]');
}

int[] ei(int y, int ntrial) {
return {(1 - min({1, y})), (1 - min({1, ntrial - y}))};
}

real eiBinomialSM_lpmf(int y, real mu, real so, real sm, int trials) {
int yom[2] = ei(y, trials);
vector[3] pom = linkfunc(so, sm);

return log(yom[1]*pom[1] + yom[2]*pom[2] +
       pom[3]*exp(binomial_logit_lpmf(y | trials, mu)));
}

int eiBinomialSM_rng(real mu, real so, real sm, int trials) {
int which_component = categorical_rng(linkfunc(so, sm));

if (which_component == 1) {
return 0;
}
if (which_component == 2) {
return trials;
}

return binomial_rng(trials, inv_logit(mu));
}
"

stanvars <- stanvar(scode = stan_funs, block = "functions") +
  stanvar(as.integer(wf$bindenom), name = "trials")
```

```
  # Reasonable weak regularization for logistic coeffs
  priors <- set_prior("normal(0,5)", class = "b")
  brm(formula = formula_,
      data = wf,
      prior = priors,
      family = eiBinomialSM,
      stanvars = stanvars,
      ...)
  }

# Fit an Endpoint-Inflated Binomial model w/latent normal link
fit_eiBinomialLN <- function(formula_, ...){
  eiBinomialLN <- custom_family(
    "eiBinomialLN", dpars = c("mu", "muL", "sdL"), # mixture scores
    links = c("identity", "identity", "identity"),
    type = "int", vars = "trials[n]",
    log_lik =
      function(i, draws) {
        mu <- if(is.null(dim(draws$dpars$mu))) {
          draws$dpars$mu
        } else{
          draws$dpars$mu[, i]
        }
        muL <- if(is.null(dim(draws$dpars$muL))) {
          draws$dpars$muL
        } else{
          draws$dpars$muL[, i]
        }
        sdL <- if(is.null(dim(draws$dpars$sdL))) {
          draws$dpars$sdL
        } else{
          draws$dpars$sdL[, i]
        }
        trials <- draws$data$trials[i]
        y <- draws$data$Y[i]
        eiBinomialLN_lpmf(y, mu, muL, sdL, trials)
      },
    predict =
      function(i, draws, ...) {
        mu <- if(is.null(dim(draws$dpars$mu))) {
          draws$dpars$mu
        } else{
```

```
        draws$dpars$mu[, i]
      }
      muL <- if(is.null(dim(draws$dpars$muL))) {
        draws$dpars$muL
      } else{
        draws$dpars$muL[, i]
      }
      sdL <- if(is.null(dim(draws$dpars$sdL))) {
        draws$dpars$sdL
      } else{
        draws$dpars$sdL[, i]
      }
      trials <- draws$data$trials[i]
      eiBinomialLN_rng(mu, muL, sdL, trials)
    },
  fitted =
    function(draws) {
      mu <- draws$dpars$mu
      muL <- draws$dpars$muL
      sdL <- draws$dpars$sdL
      p <- split_nrow(mapply(cutnorm2, muL, exp(sdL)), 3)[-1]
      trials <- median(draws$data$trials)

      (p[[1]] + p[[2]]*inv_logit_scaled(mu))*trials
    }

)

stan_funs <- "
vector linkfunc(real muL, real sdL){
real lsdL = exp(sdL);
real p1 = normal_cdf(-1, muL, lsdL);
real p2 = normal_cdf(1, muL, lsdL);

  return [p1, 1-p2, p2-p1]';
}

int[] ei(int y, int ntrial) {
return {(1 - min({1, y})), (1 - min({1, ntrial - y}))};
}

real eiBinomialLN_lpmf(int y, real mu, real muL, real sdL, int trials) {
```

```
  int yom[2] = ei(y, trials);
  vector[3] pom = linkfunc(muL, sdL);

  return log(yom[1]*pom[1] + yom[2]*pom[2] +
         pom[3]*exp(binomial_logit_lpmf(y | trials, mu)));


  int eiBinomialLN_rng(real mu, real muL, real sdL, int trials) {
  int which_component = categorical_rng(linkfunc(muL, sdL));

  if (which_component == 1) {
  return 0;
  }
  if (which_component == 2) {
  return trials;
  }

  return binomial_rng(trials, inv_logit(mu));
  }
  "

  stanvars <- stanvar(scode = stan_funs, block = "functions") +
    stanvar(as.integer(wf$bindenom), name = "trials")

  # Reasonable weak regularization for logistic coeffs
  priors <- set_prior("student_t(3, 0, 10)", class = "b")
  brm(formula = formula_,
      data = wf,
      prior = priors,
      family = eiBinomialLN,
      stanvars = stanvars,
      ...)
  }

# Fit an Endpoint-Inflated Binomial model w/latent normal link
fit_eiBinomialSR2 <- function(formula_, ...){
  eiBinomialSR2 <- custom_family(
    "eiBinomialSR2", dpars = c("mu", "pom", "pei"), # mixture scores
    links = c("identity", "identity", "identity"),
    type = "int", vars = "trials[n]",
    log_lik =
      function(i, draws) {
```

```
    mu <- if(is.null(dim(draws$dpars$mu))) {
      draws$dpars$mu
    } else{
      draws$dpars$mu[, i]
    }
    pom <- if(is.null(dim(draws$dpars$pom))) {
      draws$dpars$pom
    } else{
      draws$dpars$pom[, i]
    }
    pei <- if(is.null(dim(draws$dpars$pei))) {
      draws$dpars$pei
    } else{
      draws$dpars$pei[, i]
    }
    trials <- draws$data$trials[i]
    y <- draws$data$Y[i]
    eiBinomialSR2_lpmf(y, mu, pom, pei, trials)
  },
predict =
  function(i, draws, ...) {
    mu <- if(is.null(dim(draws$dpars$mu))) {
      draws$dpars$mu
    } else{
      draws$dpars$mu[, i]
    }
    pom <- if(is.null(dim(draws$dpars$pom))) {
      draws$dpars$pom
    } else{
      draws$dpars$pom[, i]
    }
    pei <- if(is.null(dim(draws$dpars$pei))) {
      draws$dpars$pei
    } else{
      draws$dpars$pei[, i]
    }
    trials <- draws$data$trials[i]
    eiBinomialSR2_rng(mu, pom, pei, trials)
  },
fitted =
  function(draws) {
    mu <- draws$dpars$mu
```

```
      pom <- inv_logit_scaled(draws$dpars$pom)
      pei <- inv_logit_scaled(draws$dpars$pei)
      trials <- median(draws$data$trials)

      ((1-pei)*pom + pei*inv_logit_scaled(mu))*trials
    }

)

stan_funs <- "
int[] ei(int y, int ntrial) {
return {(1 - min({1, y})), (1 - min({1, ntrial - y}))};
}

real eiBinomialSR2_lpmf(int y, real mu, real pom, real pei, int trials) {
int yom[2] = ei(y, trials);

return log((1-inv_logit(pei))*(yom[1]*(1 - inv_logit(pom)) +
       yom[2]*inv_logit(pom)) +
inv_logit(pei)*exp(binomial_logit_lpmf(y | trials, mu)));
}

int eiBinomialSR2_rng(real mu, real pom, real pei, int trials) {
int which_component = categorical_rng([(1 - inv_logit(pei))*(1 - inv_logit(pom)),
                      (1 - inv_logit(pei))*inv_logit(pom), inv_logit(pei)]');

if (which_component == 1) {
return 0;
}
if (which_component == 2) {
return trials;
}

return binomial_rng(trials, inv_logit(mu));
}
"

stanvars <- stanvar(scode = stan_funs, block = "functions") +
  stanvar(as.integer(wf$bindenom), name = "trials")

# Reasonable weak regularization for logistic coeffs
priors <- set_prior("student_t(3, 0, 10)", class = "b")
```

```
  brm(formula = formula_,
      data = wf,
      prior = priors,
      family = eiBinomialSR2,
      stanvars = stanvars,
      ...)
}

# Fit an Endpoint-Inflated Binomial model w/latent normal link
fit_eiBetaBinomialLN <- function(formula_, ...){
  eiBetaBinomialLN <- custom_family(
    "eiBetaBinomialLN", dpars = c("mu", "rho", "muL", "sdL"),
    links = c("identity", "identity", "identity", "identity"),
    type = "int", vars = "trials[n]",
    log_lik =
      function(i, draws) {
        mu <- if(is.null(dim(draws$dpars$mu))) {
          draws$dpars$mu
        } else{
          draws$dpars$mu[, i]
        }
        muL <- if(is.null(dim(draws$dpars$muL))) {
          draws$dpars$muL
        } else{
          draws$dpars$muL[, i]
        }
        sdL <- if(is.null(dim(draws$dpars$sdL))) {
          draws$dpars$sdL
        } else{
          draws$dpars$sdL[, i]
        }
        trials <- draws$data$trials[i]
        y <- draws$data$Y[i]
        eiBetaBinomialLN_lpmf(y, mu, rho, muL, sdL, trials)
      },
    predict =
      function(i, draws, ...) {
        mu <- if(is.null(dim(draws$dpars$mu))) {
          draws$dpars$mu
        } else{
          draws$dpars$mu[, i]
        }
```

```
      muL <- if(is.null(dim(draws$dpars$muL))) {
        draws$dpars$muL
      } else{
        draws$dpars$muL[, i]
      }
      sdL <- if(is.null(dim(draws$dpars$sdL))) {
        draws$dpars$sdL
      } else{
        draws$dpars$sdL[, i]
      }
      trials <- draws$data$trials[i]
      eiBetaBinomialLN_rng(mu, rho, muL, sdL, trials)
    },
  fitted =
    function(draws) {
      mu <- draws$dpars$mu
      muL <- draws$dpars$muL
      sdL <- draws$dpars$sdL
      p <- split_nrow(mapply(cutnorm2, muL, exp(sdL)), 3)[-1]
      trials <- median(draws$data$trials)

      (p[[1]] + p[[2]]*inv_logit_scaled(mu))*trials
    }

)

stan_funs <- "
vector linkfunc(real muL, real sdL){
real lsdL = exp(sdL);
real p1 = normal_cdf(-1, muL, lsdL);
real p2 = normal_cdf(1, muL, lsdL);

  return [p1, 1-p2, p2-p1]';
}

int[] ei(int y, int ntrial) {
return {(1 - min({1, y})), (1 - min({1, ntrial - y}))};
}

real eiBetaBinomialLN_lpmf(int y, real mu, real rho, real muL,
                           real sdL, int trials) {
int yom[2] = ei(y, trials);
```

```
  vector[3] pom = linkfunc(muL, sdL);
  real lrho = exp(rho);
  real lmu = inv_logit(mu);

  return log(yom[1]*pom[1] + yom[2]*pom[2] +
         pom[3]*exp(beta_binomial_lpmf( y | trials, lmu/lrho, (1-lmu)/lrho)));
  }

  int eiBetaBinomialLN_rng(real mu, real rho, real muL, real sdL, int trials) {
  int which_component = categorical_rng(linkfunc(muL, sdL));
  real lrho = exp(rho);
  real lmu = inv_logit(mu);

  if (which_component == 1) {
  return 0;
  }
  if (which_component == 2) {
  return trials;
  }

  return beta_binomial_rng(trials, lmu/lrho, (1-lmu)/lrho);
  }
  "

  stanvars <- stanvar(scode = stan_funs, block = "functions") +
    stanvar(as.integer(wf$bindenom), name = "trials")

  # Reasonable weak regularization for logistic coeffs
  priors <- set_prior("student_t(3, 0, 10)", class = "b")
  brm(formula = formula_,
      data = wf,
      prior = priors,
      family = eiBetaBinomialLN,
      stanvars = stanvars,
      ...)
}

### LOAD DATA
wf <- read_delim(
  "in/whitefly.txt",
  delim = " ", skip = 10
  ) %>%
```

```
  mutate(
    trt =
      as.factor(ifelse(trt == 5, 0, trt)),
    rep = as.factor(rep)
    )

### FIT MODELS
set.seed(69420)

# Model 0: binomial
save_fit("bin_fixed", fit_bin,
         nlive|trials(bindenom) ~ trt + week + rep)

# Model 1: ZIB
save_fit("zib_fixed", fit_zib,
         nlive|trials(bindenom) ~ trt + week + rep)

save_fit("zib_0trt", fit_zib,
         bf(nlive|trials(bindenom) ~ trt + week + rep,
            zi ~ trt))

save_fit("zib_0pred", fit_zib,
         bf(nlive|trials(bindenom) ~ trt + week + rep,
            zi ~ trt + week + rep))

# Model 2: EIB - softmax SR1
save_fit("eib_fixed", fit_eiBinomialSM,
         nlive ~ trt + week + rep)

save_fit("eib_0trt", fit_eiBinomialSM,
         bf(nlive ~ trt + week + rep,
            so ~ trt))

save_fit("eib_0pred", fit_eiBinomialSM,
         bf(nlive ~ trt + week + rep,
            so ~ trt + week + rep))

# Model 3: EIB - normal cdf SR1
save_fit("neib_fixed", fit_eiBinomialLN,
         nlive ~ trt + week + rep)

save_fit("neib_0trt", fit_eiBinomialLN,
```

```
         bf(nlive ~ trt + week + rep,
            muL ~ trt),
         control = list(max_treedepth = 12))

save_fit("neib_0pred", fit_eiBinomialLN,
         bf(nlive ~ trt + week + rep,
            muL ~ trt + week + rep))

# Model 4: EIB - SR2
save_fit("sr2_fixed", fit_eiBinomialSR2,
         bf(nlive ~ trt + week + rep))

save_fit("sr2_pei", fit_eiBinomialSR2,
         bf(nlive ~ trt + week + rep,
            pei ~ trt))

# Model 5: EIB - normal cdf SR1 + betabinomial/randeff
save_fit("eibb_0trt", fit_eiBetaBinomialLN,
         bf(nlive ~ trt + week + rep,
            muL ~ trt),
         control = list(max_treedepth = 14, adapt_delta = 0.9))

save_fit("sr1_rand", fit_eiBinomialLN,
         bf(nlive ~ trt + week + rep + (1|plantid),
            muL ~ trt),
         control = list(max_treedepth = 14, adapt_delta = 0.9))
```

Bibliography

Albert, J. and Chib, S. (1997). Bayesian methods for cumulative, sequential and two-step ordinal data regression models, *Technical report*.

Bernardo, J. M. (1996). The concept of exchangeability and its applications, *Far East Journal of Mathematical Sciences* **4**: 111–122.

Betancourt, M. (2017). A conceptual introduction to hamiltonian monte carlo, *arXiv preprint arXiv:1701.02434* .

Betancourt, M. and Girolami, M. (2015). Hamiltonian monte carlo for hierarchical models, *Current trends in Bayesian methodology with applications* **79**: 30.

Burkner, P.-C. (2018). Advanced Bayesian multilevel modeling with the R package brms, *The R Journal* **10**(1): 395–411.

Carpenter, B., Gelman, A., Hoffman, M. D., 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* **76**(1).

Daykin, A. R. and Moffatt, P. G. (2002). Analyzing ordered responses: A review of the ordered probit model, *Understanding Statistics: Statistical Issues in Psychology, Education, and the Social Sciences* **1**(3): 157–166.

Deng, D. and Zhang, Y. (2015). Score tests for both extra zeros and extra ones in binomial mixed regression models, *Communications in Statistics-Theory and Methods* **44**(14): 2881–2897.

Dupuy, J. (2017). Inference in a generalized endpoint-inflated binomial regression model, *Statistics* .

Ferrari, S. and Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions, *Journal of Applied Statistics* **31**(7): 799–815.

Gelfand, A. E., Dey, D. K. and Chang, H. (1992). Model determination using predictive distributions with implementation via sampling-based methods, *Technical report*, STANFORD UNIV CA DEPT OF STATISTICS.

Gelman, A. (2013). Understanding posterior p-values, *Electronic Journal of Statistics* .

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A. and Rubin, D. (2014). *Bayesian Data Analysis*, CRC Press.

Gelman, A., Hwang, J. and Vehtari, A. (2014). Understanding predictive information criteria for bayesian models, *Statistics and computing* **24**(6): 997–1016.

Gelman, A., Simpson, D. and Betancourt, M. (2017). The prior can generally only be understood in the context of the likelihood, *arXiv preprint arXiv:1708.07487* .

Greene, W. (2012). *Econometric Analysis*, Prentice Hall.

Hall, D. B. (2000). Zero-inflated poisson and binomial regression with random effects: a case study, *Biometrics* **56**(4): 1030–1039.

Hall, D. B. and Zhang, Z. (2004). Marginal models for zero inflated clustered data, *Statistical Modelling* **4**(3): 161–180.

Johnson, V. (2004). A bayesian χ^2 test for goodness-of-fit, *The Annals of Statistics* **32**: 2361–2384.

Johnson, V. E. et al. (2004). A bayesian χ^2 test for goodness-of-fit, *The Annals of Statistics* **32**(6): 2361–2384.

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

McElreath, R. (2016). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*, Vol. 122, CRC Press.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *http://www.R-project.org/*

Spiegelhalter, D. J., Best, N. G., Carlin, B. P. and Van Der Linde, A. (2002). Bayesian measures of model complexity and fit, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **64**(4): 583–639.

Stan Development Team (2017). *Stan Modeling Language: User's Guide and Reference Manual*. `https://github.com/stan-dev/stan/releases/download/v2.17.0/stan-reference-2.17.0.pdf`.

Stan Development Team (2018). RStan: the R interface to Stan. R package version 2.17.3.
URL: *http://mc-stan.org/*

Tak, H. and Morris, C. (2015). Data-dependent posterior propriety of bayesian beta-binomial-logit model. *Available from arXiv*.

Tian, G., Ma, H., Zhou, Y. and Deng, D. (2015). Generalized endpoint-inflated binomial model, *Computational Statistics and Data Analysis* .

van Iersel, M. W., Oetting, R. D. and Hall, D. B. (2000). Imidacloprid applications by subirrigation for control of silverleaf whitefly (homoptera: Aleyrodidae) on poinsettia, *Journal of economic entomology* **93**(3): 813–819.

Vehtari, A., Gelman, A. and Gabry, J. (2015). Pareto smoothed importance sampling, *arXiv preprint arXiv:1507.02646* .

Vehtari, A., Gelman, A. and Gabry, J. (2017). Practical bayesian model evaluation using leave-one-out cross-validation and waic, *Statistics and Computing* **27**(5): 1413–1432.

Wang, K. (2017). Availability and consumption of fruits and vegetables among non-hispanic whites, blacks, hispanics, and asians in the usa: findings from the 2011–2012 california health interview adult survey, *Journal of racial and ethnic health disparities* **4**(3): 497–506.

Watanabe, S. (2010). Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory, *Journal of Machine Learning Research* **11**(Dec): 3571–3594.

Yang, K., LeJeune, J., Alsdorf, D., Lu, B., Shum, C. and Liang, S. (2012). Global distribution of outbreaks of water-associated infectious diseases, *PLoS neglected tropical diseases* **6**(2): e1483.

Yen, S. T., Tan, A. K. et al. (2011). Fruit and vegetable consumption in malaysia: a count system approach, *International Congress*.