Computational Statistics (2026) 41:42
https://doi.org/10.1007/s00180-026-01720-y

SOFTWARE ARTICLE

FlexReg: an R package for fitting a general class of mixture regression models with bounded responses in a Bayesian framework

Roberto Ascari[1] · **Agnese Maria Di Brisco**[2] · **Sonia Migliorati**[1] · **Andrea Ongaro**[1]

Received: 24 September 2025 / Accepted: 13 January 2026

Abstract
Bounded responses, either continuous or discrete, are common in many fields, such as ecology, economics, and biomedical sciences. Standard regression models often fail to capture features like bimodality, heavy tails, overdispersion, or excess zeros, which frequently arise in these contexts. The `FlexReg` package, which provides a unified Bayesian framework for regression modeling of bounded outcomes, addresses all these challenges through flexible distributional assumptions and robust estimation based on Hamiltonian Monte Carlo implemented via the Stan language. The package implements beta-type and binomial-type models, along with their flexible and variance-inflated extensions, and allows for handling values on the boundary of the response support, when needed. Beyond model fitting, `FlexReg` includes tools for convergence diagnostics, posterior summaries, residual analysis, and predictive checking. By integrating and enriching recent methodological advances within a user-friendly Bayesian framework, this work delivers a computational infrastructure that facilitates the application of flexible regression methods for bounded data.

Keywords Augmentation · Beta · Binomial · Hamiltonian Monte Carlo · Stan

1 Introduction

This paper introduces the `FlexReg` R package, which implements a comprehensive framework for regression modeling of bounded continuous and discrete responses building on recent methodological developments. Rates and proportions are typical examples of bounded continuous outcomes, while counts from fixed number of tri-

Extended author information available on the last page of the article

Published online: 07 February 2026

als, such as the number of successes in a binomial experiment, represent widespread cases of discrete responses.

It is well established that standard regression approaches may be inadequate in these settings. For example, linear models applied to proportions violate the distributional assumption of normality and can produce predicted values outside the support (0, 1).

One possible strategy is to map bounded responses to Euclidean space via transformations (e.g., logit transformation), allowing the application of standard regression methods (Aitchison 2003). However, this approach can hinder interpretability, as the estimated parameters refer to the transformed scale rather than the original one. Moreover, key assumptions such as symmetry and homoscedasticity are frequently violated in practice on the transformed scale.

For continuous bounded responses, an increasingly popular alternative is the beta regression model (Ferrari and Cribari-Neto 2004), which assumes a beta-distributed response. For bounded discrete responses standard approaches include the binomial regression model and its extensions to account for overdispersion, such as the beta-binomial (BB) regression model.

Recently, several more flexible regression models for bounded data have been proposed. These models share a common mixture-based structure characterized by specific relationships between the mixture components. This design ensures desirable theoretical and computational properties, such as identifiability, likelihood boundedness, and tractable expressions for the moments. A key feature of these models is that they directly regress the marginal mean onto covariates while also allowing for mixture component-specific regressions. From a practical perspective, they offer increased flexibility to capture data features that standard models typically do not explain, including bimodality, heavy tails, the presence of outliers, or overdispersion.

In the continuous case, the flexible beta (FB) and variance-inflated beta (VIB) regression models (Migliorati et al. 2018; Di Brisco et al. 2020) address these data characteristics on the unit interval. The extended framework in Di Brisco and Migliorati (2019) further accommodates responses that take values at the boundary of the support (i.e., 0 and/or 1).

For discrete bounded responses, the flexible beta-binomial (FBB) regression model (Ascari and Migliorati 2021) has been proposed to handle overdispersion and excess zeros.

All the models described above can be estimated using a Bayesian approach, which is particularly well-suited to handle their inherent mixture structure.

The `FlexReg` package facilitates Bayesian estimation for these models through an accessible interface designed to support practitioners and applied statisticians, even in the absence of advanced Bayesian expertise, by providing functions deliberately designed to mirror widely used regression tools (e.g., `lm()` and `glm()`), together with carefully chosen default arguments that relieve users from specifying prior distributions or constructing custom Markov Chain Monte Carlo (MCMC) algorithms. The name of the package highlights its focus on flexibility, emphasizing

that it enables the estimation of regression models for bounded responses beyond standard approaches. Importantly, `FlexReg` substantially extends the functionality of existing R packages for bounded data. For bounded continuous responses, it goes beyond classical beta models by including more flexible alternatives, such as the FB and VIB. In contrast, the `betareg` package (Cribari-Neto and Zeileis 2010) is limited to beta regression with maximum likelihood estimation, whereas the `zoib` package (Liu and Kong 2015) focuses exclusively on beta regression models (including 0 and/or 1 responses) estimated via MCMC methods. For bounded discrete responses, the `FlexReg` package supports both binomial and beta-binomial regression models, as well as the FBB regression to accommodate complex patterns of overdispersion within a Bayesian framework. Standard models can be estimated via maximum likelihood using `stats` (binomial) or `aod` (Lesnoff and Lancelot 2012) and `VGAM` (Yee 2025) (beta-binomial) packages.

In addition to model fitting, `FlexReg` includes a rich set of functionalities to support the full analysis workflow: distribution-related functions, convergence diagnostics, summaries, goodness of fit measures, and graphical displays.

This paper presents the `FlexReg` package as a unified computational framework for the estimation and analysis of flexible regression models for bounded data. Through illustrative applications, we highlight both the methodological foundations and the practical relevance of these models in capturing complex features of real-world data. The outline of the paper is as follows. Section 2 provides the statistical background concerning distributions and regression models for bounded responses. Section 3 describes the package structure and is arranged into four subsections corresponding to four macro-categories of functions: (i) distribution-related functions (i.e., probability density/mass functions, cumulative distribution and quantile functions, and random generation functions), (ii) core functions for regression models, (iii) convergence functions and posterior checks, and (iv) summary, goodness of fit, and visualization functions. Each subsection includes practical examples using one of the datasets available in the `FlexReg` package. Finally, Sect. 4 provides some concluding remarks.

2 Statistical methods

This section outlines the key statistical concepts underlying the models implemented in the `FlexReg` package. While the exposition is not intended to be exhaustive, readers interested in the core methodological details are referred to Migliorati et al. (2018), Di Brisco and Migliorati (2019), Di Brisco et al. (2020), and Ascari and Migliorati (2021). In addition to reviewing these established results, we also present new theoretical developments that extend previous contributions and strengthen the modeling and estimation framework.

For clarity and consistency with the structure of the package, which provides two separate functions for continuous and discrete bounded responses, the presentation is divided into two subsections, each addressing one of the two model classes.

2.1 Distributions and regression models for continuous bounded responses

This subsection first reviews some distributions suitable for modeling a continuous random variable (rv) Y defined on the (0, 1) interval. It then introduces augmented distributions to cope with values on the boundary of the support (i.e., when the response is defined on (0, 1], [0, 1), or [0, 1]). Finally, it describes how to specify regression structures for the mean, precision, and, when applicable, the augmentation-related parameters.

The beta distribution is perhaps the most well-known for modeling responses on the (0, 1)-bounded interval. In its mean-precision parameterization (Ferrari and Cribari-Neto 2004), it is characterized by the following probability density function (pdf):

$$f_B(y|\mu,\phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, \qquad 0<y<1, \quad (1)$$

where $\Gamma(\cdot)$ is the Gamma function. The first two order moments of the beta are

$$\mathbb{E}[Y] = \mu \quad \text{and} \quad \text{Var}(Y) = \frac{\mu(1-\mu)}{\phi+1};$$

thus the parameter $\mu \in (0,1)$ represents the expected value of Y, whereas $\phi > 0$ is a precision parameter, being in the denominator of the variance of the response.

A flexible generalization of the beta distribution, which accommodates a wide range of density shapes in terms of tail behavior, asymmetry, and multimodality, is the FB distribution (Migliorati et al. 2018). It is defined as a special two-component mixture of beta distributions:

$$f_{FB}(y|\lambda_1,\lambda_2,\phi,p) = pf_B(y|\lambda_1,\phi) + (1-p)f_B(y|\lambda_2,\phi), \qquad 0<y<1, \quad (2)$$

where $0<\lambda_2<\lambda_1<1$ represent the means of the second and first mixture components, respectively, ϕ is a common precision parameter, and p is the mixing proportion.

In a regression setting, it is convenient to adopt an alternative parameterization that explicitly includes the overall mean parameter μ. In this formulation, the density can be written as $f_{FB}(y|\mu,\phi,p,w)$, where $0<\mu = p\lambda_1 + (1-p)\lambda_2 < 1$ is the overall mean, and $w \in (0,1)$ represents the normalized distance between the two component-means, defined as

$$w = \frac{\lambda_1 - \lambda_2}{\min\left\{\frac{\mu}{p}, \frac{1-\mu}{1-p}\right\}}. \tag{3}$$

The first two moments of the FB are:

$$\mathbb{E}[Y] = \mu \quad \text{and} \quad \text{Var}(Y) = \frac{\mu(1-\mu) + p(1-p)\tilde{w}^2\phi}{\phi + 1},$$

where $\tilde{w} = w \min\left\{\frac{\mu}{p}, \frac{1-\mu}{1-p}\right\}$ is the non-normalized version of the parameter w. Finally, the component-specific means can be expressed in terms of the new parameterization as:

$$\lambda_1 = \mu + (1-p)\tilde{w}, \qquad \lambda_2 = \mu - p\tilde{w}. \tag{4}$$

Another generalization of the beta distribution, designed to account for heavier tails and to accommodate outlying observations, is the VIB distribution (Di Brisco et al. 2020). It is defined as a special two-component mixture of beta distributions, where the components share the same mean but differ in their variances:

$$f_{VIB}(y|\mu, \phi, k, p) = p f_B(y|\mu, \phi k) + (1-p) f_B(y|\mu, \phi), \qquad 0 < y < 1, \tag{5}$$

where $0 < \mu < 1$ represents both the mean of the components and the overall mean, ϕ is the precision parameter, $k \in (0, 1)$ determines the extent of the variance inflation, and p is the mixing proportion. The first two moments of the VIB are

$$\mathbb{E}[Y] = \mu \quad \text{and} \quad \text{Var}(Y) = p\left(\frac{\mu(1-\mu)}{\phi k + 1}\right) + (1-p)\left(\frac{\mu(1-\mu)}{\phi + 1}\right).$$

It is important to clarify that the term "inflation" in the name of the VIB distribution refers to its ability to accommodate a wider range of variance shapes compared to the standard beta distribution and does not indicate an increased probability mass at specific points in the support.

Cases where the response variable can take boundary values (e.g., in (0, 1], [0, 1), or [0, 1]), are handled through a strategy referred to as "augmentation" in Ospina and Ferrari (2012), a term that we also adopt throughout the paper. Within this framework, the augmented distribution is defined as a three-part mixture, assigning positive probabilities to 0 and/or 1 and a continuous density to the open interval (0, 1):

$$f_{\bullet, aug}(y|\cdot) = \begin{cases} q_0, & \text{if } y = 0, \\ q_1, & \text{if } y = 1, \\ q_2 f_\bullet(y|\cdot), & \text{if } 0 < y < 1, \end{cases} \tag{6}$$

where $0 \leq q_0 < 1$, $0 \leq q_1 < 1$, and $0 < q_2 = 1 - q_0 - q_1$. In our approach, the density $f_\bullet(y|\cdot)$ can be the beta (1), FB (2), or VIB (5). When values equal to 0, 1, or both are not present in the response variable, Eq. (6) can be simplified by setting one or both of the probabilities q_0 and q_1 to zero. The marginal mean of the augmented distribution is $\mathbb{E}[Y] = q_1 + q_2\mathbb{E}[Y|0 < Y < 1] = \nu$, and the variance is $\text{Var}(Y) = q_2\text{Var}(Y|0 < Y < 1) + q_1 + q_2\nu^2 - (q_1 + q_2\nu)^2$.

Since (6) is expressed as a three-component mixture, the general cumulative distribution function takes the form:

$$F_{\bullet,\,aug}(t|\cdot) = q_0\mathbb{I}(t \geq 0) + q_1\mathbb{I}(t \geq 1) + q_2F_\bullet(t|\cdot),$$

where $t \in \mathbb{R}$, $\mathbb{I}(\cdot)$ denotes the indicator function and $F_\bullet(t|\cdot) = \int_{-\infty}^{t} f_\bullet(y|\cdot)dy$ is the cumulative distribution function of the non-augmented distribution. For example, by exploiting the mixture structure expressed in Eq. (2), the cumulative distribution function of a zero and one augmented FB can be written as

$$\begin{aligned} F_{FB,\,aug}(t|\lambda_1, \lambda_2, \phi, p) =& q_0\mathbb{I}(t \geq 0) + q_1\mathbb{I}(t \geq 1) + \\ &+ q_2\left[p\int_{-\infty}^{t} f_B(y|\lambda_1, \phi)dy + (1-p)\int_{-\infty}^{t} f_B(y|\lambda_2, \phi)dy\right], \end{aligned}$$

where the integrals in the square brackets correspond to the cumulative distribution functions of particular beta rvs. This three-component mixture representation of the zero and/or one augmented distribution will be used to develop a novel and computationally efficient Bayesian estimation procedure, which will be specified in Sect. 2.3.

Let us now consider a sample of size N, with y_i denoting the response variable, and x_i a vector of covariates for observation i. We assume that $y_1, \ldots, y_N$ are realizations of independent rvs $Y_1, \ldots, Y_N$. Based on the chosen distribution (augmented or not) for the response variable Y_i, a GLM-type approach allows us to define a regression model for the mean μ_i and, optionally, for the precision parameter ϕ_i. Furthermore, when applicable, the probabilities associated with the augmentation terms, q_{0i} and/or q_{1i}, can also be regressed onto covariates. In general, the regression structure we consider is:

$$g_1(\mu_i) = \mathrm{x}_i^\intercal\boldsymbol{\beta}, \quad g_2(\phi_i) = \mathrm{z}_i^\intercal\boldsymbol{\psi}, \quad g_3(q_{0i}) = \mathrm{x}_{0i}^\intercal\boldsymbol{\omega}_0, \quad \text{and} \quad g_4(q_{1i}) = \mathrm{x}_{1i}^\intercal\boldsymbol{\omega}_1, \tag{7}$$

where x_i, z_i, x_{0i}, and x_{1i} are vectors of (possibly overlapping) covariates, and $\boldsymbol{\beta}$, $\boldsymbol{\psi}$, $\boldsymbol{\omega}_0$, $\boldsymbol{\omega}_1$ are vectors of regression coefficients. The link function $g_1(\cdot)$, used in the mean model, is strictly monotonic and maps (0, 1) to $\mathbb{R}$. A common choice is the logit function, due to its straightforward interpretation of the regression coefficients in terms of log-odds ratios. Other possibilities include probit, cloglog, and loglog functions.

The function $g_2(\cdot)$, used in the precision model, is a link from $\mathbb{R}^+$ to $\mathbb{R}$, with possible choices being the logarithmic and square root functions.

For the augmentation-related link functions $g_3(\cdot)$ and $g_4(\cdot)$, it is necessary to distinguish between the case of single-boundary augmentation (either zero or one) and

simultaneous augmentation at both boundaries. In the former case, the link function $g_3(\cdot)$ or $g_4(\cdot)$, usually logit, is from (0, 1) to $\mathbb{R}$. In the latter case, a bivariate logit link is required to ensure the constraint $0 < q_{0i} + q_{1i} < 1$, while preserving a convenient interpretation of the regression parameters in terms of log-odds ratios:

$$g_3(q_{0i}) = \log(q_{0i}/(1 - q_{0i} - q_{1i})), \quad g_4(q_{1i}) = \log(q_{1i}/(1 - q_{0i} - q_{1i})).$$

The three-component mixture representation in (6) is exploited to efficiently estimate the parameters of a zero and/or one augmented model. Since the true mixture allocations are known, the augmented and non-augmented parts can be fitted separately. Specifically, the estimation of the non-augmented component depends on the chosen regression model (Beta, FB, or VIB) and is based on the subset of responses in the open interval (0, 1). When only zero or one augmentation is present, the corresponding augmentation-related parameters ($\boldsymbol{\omega}_0$ or $\boldsymbol{\omega}_1$, respectively) are estimated through a binomial regression model, where the success is defined as $Y = 0$ for a zero-augmented model and $Y = 1$ for a one-augmented model. If both augmentations are present, estimating $\boldsymbol{\omega}_0$ and $\boldsymbol{\omega}_1$ requires a multinomial regression model with three outcome categories: $Y = 0$, $Y = 1$, and $Y \in (0, 1)$.

It is worth noting that this separate estimation approach represents a novelty compared with the methodology in Di Brisco and Migliorati (2019), where the augmented part of the model was estimated jointly with the continuous part. Although this separate approach is theoretically equivalent to the joint estimation used in earlier works, it yields a substantial improvement in the computational efficiency of the estimation procedure.

2.2 Distributions and regression models for bounded discrete responses

In this section, we focus on a discrete bounded response Y taking values in the set $\mathcal{S}_n = \{0, 1, \ldots, n\}$, for any fixed n. The variable Y can be interpreted as the number of "successes" in n Bernoulli trials. A widespread model for such data is the binomial distribution. However, it is well-known that both the binomial distribution and the corresponding regression model may suffer from overdispersion (Hinde and Demétrio 1998), which occurs when the sample variance is larger than the theoretical variance implied by the model. To overcome this problem, a widely used alternative is the BB distribution, originally proposed by Williams (1975).

The BB distribution for a discrete rv Y with support $\mathcal{S}_n = \{0, 1, \ldots, n\}$ can be derived through a compound specification, assuming that $Y|\Pi = \pi \sim \text{Binomial}(n, \pi)$ and $\Pi \sim \text{Beta}(\mu, \phi)$. The probability mass function (pmf) is obtained by marginalizing the joint distribution of $(Y, \Pi)^\intercal$ and is expressed as

$$f_{BB}(y|\mu, \phi) = \binom{n}{y} \frac{B(\phi\mu + y, \phi(1 - \mu) + n - y)}{B(\phi\mu, \phi(1 - \mu))}, \qquad y \in \{0, 1, \ldots, n\}, \quad (8)$$

where $n \in \mathbb{N}$, $\mu = \mathbb{E}[Y/n] \in (0,1)$, $\phi \in \mathbb{R}^+$, and $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ is the Beta function. If $Y \sim \text{BB}(n, \mu, \phi)$, its first two moments are defined as

$$\mathbb{E}[Y] = n\mu \quad \text{and} \quad \text{Var}(Y) = n\mu(1-\mu)\left[1 + \frac{(n-1)}{(\phi+1)}\right], \tag{9}$$

where the parameter $\theta = \frac{1}{(\phi+1)}$ can be interpreted as an overdispersion parameter, since the variance of a BB distribution reduces to the binomial variance when $\theta \to 0$. Equation (9) shows that the BB distribution handles overdispersion through the extra term inside the square brackets. However, the inclusion of a single additional parameter (i.e., $\phi = (1-\theta)/\theta$) may still be insufficient to fully account for the extra variability in the observed data.

A generalization of the BB distribution can be obtained by assuming that $Y|\Pi = \pi \sim \text{Binomial}(n, \pi)$ and that $\Pi \sim \text{FB}(\lambda_1, \lambda_2, \phi, p)$. The resulting marginal distribution for Y is the FBB (Ascari and Migliorati 2021), whose pmf is

$$f_{FBB}(y|\lambda_1, \lambda_2, \phi, p) = p f_{BB}(y|\lambda_1, \phi) + (1-p) f_{BB}(y|\lambda_2, \phi), \quad y \in \{0, 1, \ldots, n\},$$

where $f_{BB}(y|\cdot)$ is defined in Eq. (8), $0 < \lambda_2 < \lambda_1 < 1$ are component-specific (scaled) means, $\phi > 0$, and $0 < p < 1$ is the mixing proportion. Thus, the FBB can be expressed as a finite, structured mixture model with two BB-distributed components that share a common precision parameter ϕ. Its first two moments are

$$\mathbb{E}[Y] = n\mu \quad \text{and} \quad \text{Var}(Y) = n\mu(1-\mu)\left[1 + \frac{(n-1)}{(\phi+1)} + \frac{(n-1)}{(\phi+1)}\phi w^2 m(\mu, p)\right],$$

where $m(\mu, p) = \min\left(\frac{\mu(1-p)}{p(1-\mu)}, \frac{p(1-\mu)}{\mu(1-p)}\right)$. The FBB variance extends the BB variance and successfully addresses overdispersion by including an additional term that depends on p and w (i.e., the parameters characterizing the latent groups identified by the mixture components). This extra term allows the model to capture various sources of extra-variation, such as the presence of latent groups, outliers, and an excess of zero observations. Since in many applications the overdispersion parameter is of greater interest than the precision parameter, the FBB variance can be rewritten in terms of θ as:

$$\text{Var}(Y) = n\mu(1-\mu)\left[1 + (n-1)\theta + (n-1)(1-\theta)w^2\, m(\mu, p)\right].$$

Even in this case, it is convenient to consider a parameterization of the FBB distribution that is more suitable for regression purposes by explicitly including the marginal (scaled) mean $\mu = \mathbb{E}[Y/n] = p\lambda_1 + (1-p)\lambda_2$. Thus, the FBB inherits the parameterization of the FB based on μ, ϕ, p, and w, the latter parameter being the normalized distance between λ_1 and λ_2 defined in Eq. (3).

As in Subsection 2.1, let us consider a sample of size N, a vector of covariates x_i and a response y_i, counting the number of successes among n_i Bernoulli trials for the i-th observation, $i = 1, \ldots, N$. Furthermore, we assume that each y_i is a realization of the rv Y_i. Following the same arguments as before, different regression models can be defined by considering different distributions for the rv Y_i. Indeed, a binomial, BB, or FBB regression model is recovered by assuming that Y_i follows the corresponding distribution. In the context of these regression models for discrete bounded responses, a common choice is to regress the (scaled) mean μ and/or the overdispersion parameter θ to covariates. It is worth noting that the binomial regression model does not possess an overdispersion parameter and that its scaled mean is $\mu = \pi$.

Both the parameters μ and θ are defined in the (0, 1) interval, so any of the link functions specified in the previous subsection can be used, mapping from (0, 1) to $\mathbb{R}$ (e.g., logit, probit, cloglog, and loglog). The regression structure for these models is then defined as follows:

$$g_1(\mu_i) = \mathrm{x}_i^\intercal \boldsymbol{\beta} \quad \text{and} \quad g_2^*(\theta_i) = \mathrm{z}_i^\intercal \boldsymbol{\psi}, \tag{10}$$

where x_i and z_i are vectors of possibly overlapping covariates, $\boldsymbol{\beta}$ and $\boldsymbol{\psi}$ are vectors of regression coefficients, and $g_2^*(\cdot)$ is a link function from (0, 1) to $\mathbb{R}$.

2.3 Bayesian inference

Let $\boldsymbol{\eta}$ denote the vector of unknown parameters, which depends on the chosen response distribution and the selected regression structure. The joint posterior distribution of $\boldsymbol{\eta}$ given the vector of responses $\mathrm{y} = (y_1, \ldots, y_n)^\intercal$ is $\varphi(\boldsymbol{\eta}|\mathrm{y}) \propto L(\mathrm{y}|\boldsymbol{\eta})\varphi(\boldsymbol{\eta})$, where $L(\mathrm{y}|\boldsymbol{\eta})$ is the likelihood function, which is straightforward to derive for all models, and $\varphi(\boldsymbol{\eta})$ is the joint prior distribution. As an illustrative example, consider a regression model for the mean based on the FB distribution, without regression for the precision parameter ϕ and no augmentation terms. In this case, the vector of unknown parameters is $\boldsymbol{\eta} = (\boldsymbol{\beta}, \phi, p, w)^\intercal$, where $\boldsymbol{\beta}$ is the vector of regression coefficients for the mean model, and ϕ, p, and w are the additional parameters of the FB distribution. The likelihood can be written as $L(\mathrm{y}|\boldsymbol{\eta}) = \prod_{i=1}^{N} f_{FB}(y_i|\mu_i, \phi, p, k)$, where $\mu_i = g_1^{-1}(\mathrm{x}_{1i}^\intercal \boldsymbol{\beta})$ is consistent with Eq. (7). In the presence of augmentation, either a binomial or a multinomial model is estimated separately, depending on whether only a single augmentation term is present or both are included. Analogous definitions and computations hold for the other models.

As for the specification of the prior distribution, a standard simplifying assumption is prior independence, which is feasible in this context due to the proposed parameterizations that guarantee the absence of constraints among parameters. This choice is consistent with the non-informative rationale, which is well-established in the Bayesian framework. Indeed, the assumption of prior independence avoids the need to rely on strong prior information to specify prior dependence structures. As a

result, regardless of the model, the joint prior can be factorized into marginal priors to be elicited. In particular, the `FlexReg` package allows for the following priors:

- ▷ $\varphi(\boldsymbol{\beta})$, $\varphi(\boldsymbol{\psi})$, $\varphi(\boldsymbol{\omega}_0)$ and/or $\varphi(\boldsymbol{\omega}_1)$: priors for the regression coefficients of the mean model, precision/overdispersion model, and augmentation models (in zero and/or one), respectively. These can be either diffuse Normal distributions centered at 0 with a suggested standard deviation of 100, or Cauchy distributions centered at 0 with a suggested scale parameter of 2.5;
- ▷ $\varphi(\theta)$: if overdispersion is not regressed onto covariates, this is a mean-precision parameterized Beta(m_θ, v_θ);
- ▷ $\varphi(\phi)$: if the precision is not regressed onto covariates, this can be either a Gamma(g, g) or a Uniform$(0, A)$;
- ▷ $\varphi(p)$: the prior of the mixing proportion parameter is a Beta(m_p, v_p);
- ▷ $\varphi(w)$: the prior of the normalized distance between the two component means is a Beta(m_w, v_w);
- ▷ $\varphi(k)$: the prior of the variance inflation parameter is a Beta(m_k, v_k).

Details concerning priors, the values of their hyperparameters, as well as their default settings, are provided in Sect. 3.2 and summarized in Tables 4 and 5.

Simulated samples $\boldsymbol{\eta}^{(s)}(s = 1, \ldots, S)$ of length S are drawn from the posterior distribution of $\boldsymbol{\eta}$ using the Hamiltonian Monte Carlo (HMC) via the Stan language (Stan Development Team 2022). HMC is a class of MCMC that exploits gradient information of the target (posterior) density to construct efficient proposal mechanisms. Thus, the `FlexReg` package depends on the `rstan` (Stan Development Team 2023) package for the Bayesian estimation step. The implementation is designed to be user-friendly: only basic knowledge of R is required to specify and estimate the desired model, without the need to directly write code in the Stan modeling language.

Table 1 Summary of datasets available in `FlexReg` package classified with respect to the nature of the response variable(s)

Dataset	Description of the response variable(s)	Continuous	Discrete
`Atomic`	counts/percentage of cells with chromosomal abnormalities	✓	✓
`Bacteria`	counts/percentage of eggs parasitized by female parasitoids	✓	✓
`Consumption`	percentage of household income that is spent rather than saved	✓	
`Election`	percentage/number of votes got by some Italian parties	✓	✓
`Reading`	rate of accuracy in reading on (0, 1] and its adjusted version on (0, 1)	✓	
`Stress`	rates of stress and anxiety	✓	

Datasets marked for both types of models contain both a continuous and a discrete version of the response variable

3 The FlexReg package

A stable version of the `FlexReg` package is available on CRAN (`install.packages("FlexReg")`); this article refers to version 1.4.2. The `FlexReg` package is built using `rstantools` (Gabry et al. 2023) to interface with Stan, and it imports several dependencies, among which `rstan` for Bayesian inference through HMC, `ggplot2` (Wickham 2016) for visualization, and `loo` (Vehtari et al. 2017) and `bayesplot` (Gabry et al. 2019) for convergence diagnostics.

`FlexReg` supports Bayesian estimation of regression models for bounded continuous and discrete responses. In addition to model fitting, it provides several comprehensive tools for post-estimation analysis, including, but not limited to, summaries and graphical representations of the regression curves, predictions, residual analysis, and convergence checks.

The package also contains a collection of datasets, summarized in Table 1, intended to facilitate direct application of the available regression models to real data. In particular, the `Consumption` dataset represents a novel contribution to the literature, while the remaining datasets serve as well-established references for analyses in the field. The `Election` dataset (Di Brisco and Migliorati 2020; Ascari and Migliorati 2022) can be analyzed using both continuous and discrete models, as it includes the percentage of votes obtained by some Italian parties along with the total number of votes (`NVotes`). This allows the recovery of the actual counts of votes for each party through multiplication. Similarly, the `Atomic` and `Bacteria` datasets (Ascari and Migliorati 2021) contain both continuous responses (denoted `y.perc`) and discrete responses (denoted `y`). In contrast, the `Consumption`, `Reading` (Ferrari and Cribari-Neto 2004; Migliorati et al. 2018; Di Brisco and Migliorati 2019), and `Stress` datasets contain only continuous percentage responses, making them suitable exclusively for continuous models.

In total, the package provides 41 functions, 13 of which are implemented as S3 methods. These functions can be grouped into four main categories:

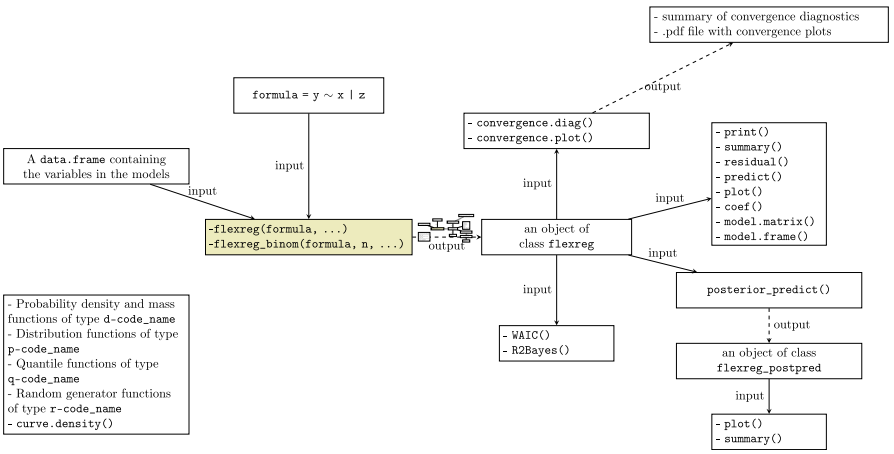

Fig. 1 `FlexReg` structure and functions' dependencies

i) distribution-related functions, including pdf/pmf, cumulative distribution, and quantile functions, as well as random generation functions;
ii) core functions for estimating regression models with bounded continuous and discrete responses;
iii) functions for convergence assessment, through numerical diagnostics and graphical outputs;
iv) functions for exploring and summarizing fitted models, including:

   – predictions and residuals;
   – Bayesian goodness-of-fit measures and posterior predictive checks;
   – summary statistics and visualization tools.

The diagram in Fig. 1 provides an overview of the package structure by illustrating the relationships between functions, their input arguments, and outputs. The two core functions of the package, `flexreg()` and `flexreg_binom()`, are highlighted in yellow in the center of the diagram and implement regression models for bounded continuous and discrete responses, respectively. To make these tools accessible to practitioners with limited experience in regression models for bounded responses and Bayesian inference, the input arguments of the main functions are designed to closely resemble those of standard regression functions, such as `lm()` and `glm()`. The output is an object of class `flexreg`, from which different analyses can be carried out, including convergence diagnostics, model summaries, and posterior predictive evaluations.

Distribution-related functions are positioned slightly outside the main flow of the diagram, as they serve more general purposes related to the distributions themselves. The use of all other functions and methods in the `FlexReg` package is illustrated through the `Bacteria` dataset, which supports both bounded continuous and discrete regression models.

Table 2 Summary of distributions available in `FlexReg` package classified with respect to the nature of the support

Name	`code_name`	Required Args.	Continuous	Discrete
beta	`Beta`	`(mu, phi)`	✓	
flexible beta	`FB`	`(mu, phi, p, w)`	✓	
variance-inflated beta	`VIB`	`(mu, phi, p, k)`	✓	
beta-binomial	`BetaBin`	`(mu, phi)`		✓
flexible beta-binomial	`FBB`	`(mu, phi, p, w)`		✓

3.1 Distribution-related functions

The functions in this category enable the use of all the distributions listed in Table 2. More details on their pdf and pmf can be found in Sect. 2. These functions are designed in analogy with the standard distribution functions in the `stats` package. Each distribution is identified by a specific `code_name` (listed in the second column of Table 2), while the prefix determines the type of output. Specifically, functions starting with `"d"` compute the probability density or mass function associated with the chosen `code_name`, while those with prefixes `"p"` and `"q"` compute the cumulative distribution function and the quantile function, respectively. Finally, functions with the prefix `"r"` generate pseudo-random values. It is important to note that, consistently with the parameterization adopted in Sect. 2.1, functions related to the beta (and BB) distribution are defined using the mean-precision parameterization. This differs from the parameterization used by the beta-related functions in the `base` package, which is based on two real positive shape parameters.

These vectorized functions require specific arguments corresponding to the parameters of the distribution of the selected rv, as listed in the third column of Table 2. Coherently with the corresponding functions for standard distributions, the `"d"` functions include an additional argument, `log`, to return densities or probabilities on the log scale, while the `"p"` and `"q"` functions include the argument `log.prob` to receive or return probabilities on the log scale.

For continuous rvs, it is also possible to specify the probability of augmentation at zero and/or one by the additional arguments `q0` and `q1`, respectively. Note that, by default, both arguments are set to `NULL`, implying no augmentation. If provided, the values of `q0` and/or `q1` must be in the unit interval (0, 1) and their sum must be less than 1. For discrete rvs, it is possible to specify the overdispersion parameter `theta`, lying in (0, 1), as an alternative to the precision parameter `phi`.

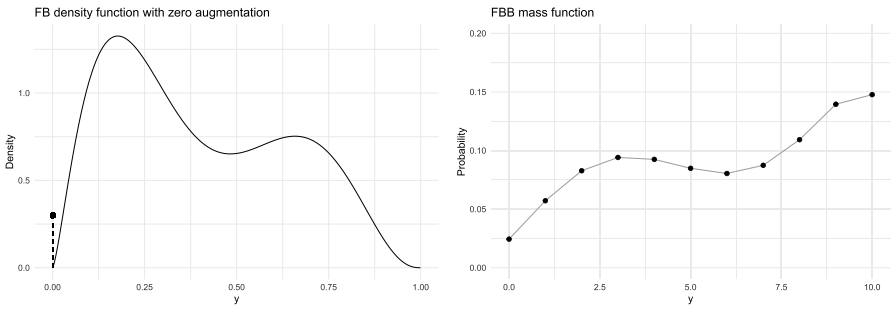

Fig. 2 *Left panel:* pdf of an augmented FB rv $f_{FB}(y|\mu = 0.4, \phi = 10, p = 0.4, w = 0.4)$ with probability of augmentation in zero $q_0 = 0.3$. *Right panel:* pmf of an FBB rv $f_{FBB}(y|\mu = 0.6, \phi = 10, p = 0.4, w = 0.9)$

```
# p.m.f. of FBB rv:
> dFBB(x = c(5,7,8), size = 10, mu = .3, phi = 20, p = .5, w = .5)
#[1] 0.10714398 0.04681211 0.02137555

# Cumulative distribution function of FBB specifying theta instead of phi:
> pFBB(q = c(5,7,8), size = 10, mu = .3, theta = 1/(20+1), p = .5, w = .5)
#[1] 0.8455959 0.9708319 0.9922075

# Quantile function of VIB:
> qVIB(prob = c(.5,.7,.8), mu = .3, phi = 20, p = .5, k = .5)
#[1] 0.2903419 0.3562967 0.3987025

set.seed(42)
# Pseudo-random numbers from an FB with zero augmentation:
> rFB(n = 6, mu = .5, phi = 30, p = .3, w = .4, q0 = .2)
#[1] 0.0000000 0.0000000 0.7099949 0.0000000 0.4046991 0.6574665
```

Finally, the `curve.density()` function produces a graphical representation of the pdf or pmf of the rv specified in the `type` argument. The remaining arguments of the function correspond to the parameters of the chosen distribution. Two illustrative examples are provided in Fig. 2. It is worth noting that when the rv is continuous with zero and/or one augmentation, the corresponding plot displays a continuous curve over (0, 1) with discrete mass at zero and/or one (see the left-hand side of the figure).

```
# The following code returns the plot on the left-hand side of Figure 2
> curve.density(type = "FB", mu = .4, phi = 10, p = .4, w = .4, q0 = .3)

# The following code returns the plot on the right-hand side of Figure 2
> curve.density(type = "FBB", size = 10, mu = .6, phi = 10, p = .4, w = .9)
```

3.2 Core functions for regression models

The functions `flexreg()` and `flexreg_binom()` form the core of the `FlexReg` package and are designed to estimate regression models for continuous and discrete bounded responses, respectively. As stated previously, model estimation is performed within a Bayesian framework, relying on the `sampling()` function from the `rstan` package. Both functions return an object of class `flexreg`, which is a list of elements, including but not limited to a `stanfit` object containing the fitted model. Defining a dedicated `flexreg` class enables the use of specific methods, such as `summary()`, `plot()`, and others, that facilitate the analysis of the results of the fitted regression models, as described below.

The main arguments of the `flexreg()` and `flexreg_binom()` functions, some of which are common, are listed in Tables 3–6 and organized into four blocks to facilitate their description.

The first block (Table 3) concerns the fundamental arguments for specifying the regression model. The required argument `formula`, of the form `y ~ x | z`, provides a symbolic description of the regression model to be fitted. Here, `y` is the response

Table 3 Arguments of core functions

Name	Description	flexreg()	flexreg_binom()
formula	an object of class formula: a symbolic description of the mean model (y∼x) or the mean and precision model (y∼x | z) to be fitted	✓	✓
zero.formula	an object of class formula of type (∼ x0) for the zero augmentation	✓	
one.formula	an object of class formula of type (∼ x1) for the one augmentation	✓	
data	an optional data.frame, list, etc., containing the variables in the model	✓	✓
type	a character specifying the type of regression model. For flexreg(), current options are "FB" (default), "VIB", and "Beta". For flexreg_binom(), current options are "FBB" (default), "BetaBin", and "Bin"	✓	✓
n	a character specifying the name of the variable containing the total number of trials		✓
link.mu	a character specifying the link function for the mean model. Available options are "logit" (default), "probit", "cloglog", and "loglog"	✓	✓
link.phi	a character specifying the link function for the precision model. Available options are "identity" (default), "log", and "sqrt"	✓	
link.theta	a character specifying the link function for the overdispersion model. Available options are "identity" (default), "logit", "probit", "cloglog", and "loglog". If link.theta = "identity", the prior distribution for theta is a beta		✓

Main block: formula, data, and link functions

variable, x is a collection of covariates (e.g., x1 + x2 + x3) for the mean model, and z is a collection of covariates for the precision model (which may partially or fully overlap with x). If z is omitted (e.g., y ∼ x), only the regression model for the mean parameter is evaluated. For flexreg() only, the additional arguments zero.formula and one.formula allow to specify the regression models for zero and one augmentation, respectively. These take objects of class formula, such as y ∼ x0 and y ∼ x1. For simplicity, the left-hand side of the latter formulas, namely y, can be omitted. Argument data requires the specification of a data.frame containing the variables specified in formula and, when appropriate, in zero.formula and/or one.formula. If data is omitted, the regression variables are taken from the environment from which the functions flexreg() and flexreg_binom() are called. The argument type allows the user to specify the distribution of the response and, consequently, the type of regression model to be fitted. For bounded continuous responses, available options are "Beta", "FB" (default), and "VIB"; for bounded

Table 4 Arguments of core functions

Name	Description	`flexreg()`	`flexreg_binom()`
`prior.beta`	a character specifying the prior distribution for the regression coefficients of the mean model. Currently, `"normal"` (default) and `"cauchy"` are supported	✓	✓
`prior.phi`	a character specifying the prior distribution for precision parameter ϕ if `link.phi = "identity"`. Currently, `"gamma"` (default) and `"unif"` are supported	✓	
`prior.psi`	a character specifying the prior distribution for the regression coefficients of the precision/overdispersion model for `flexreg()`/`flexreg_binom()`, respectively. Currently, `"normal"` (default) and `"cauchy"` are supported (not supported if `link.phi = "identity"`)	✓	✓
`prior.omega0`	a character specifying the prior distribution for the regression coefficients of the augmented model in zero, ω_0. Currently, `"normal"` (default) and `"cauchy"` are supported	✓	
`prior.omega1`	a character specifying the prior distribution for the regression coefficients of the augmented model in one, ω_1. Currently, `"normal"` (default) and `"cauchy"` are supported	✓	

Priors block

discrete responses, the supported choices are `"Bin"`, `"BetaBin"`, and `"FBB"` (default). Argument `n`, required only for the `flexreg_binom()` function, must be a character indicating the name of the variable containing the total number of Bernoulli trials. Finally, the arguments `link.mu`, `link.phi`, and `link.theta` allow to specify the type of link functions to be included in the corresponding regression model for the mean, precision, and overdispersion models, respectively.

The second and third blocks of arguments define the degree of flexibility available to users for specifying prior distributions and their related hyperparameters. The second block of arguments (summarized in Table 4) concerns the specification of the prior distributions for the parameters in the model. The theoretical aspects concerning the' specification of priors are illustrated in Subsection 2.3 and are implemented in practice through the following arguments. Priors for all regression coefficients can be either `"normal"` (the default) or `"cauchy"`. This applies to coefficients β of the mean model, ω_0 and/or ω_1 of the zero and/or one augmented models, and coefficients ψ either of the precision model in `flexreg()` function or of the overdispersion model in `flexreg_binom()` function. In the `flexreg()` function, the parameter ϕ, when not regressed onto covariates, can have either a `"gamma"` (default) or a uniform (`"unif"`) prior, specified via the argument `prior.phi`.

Table 5 Arguments of core functions

Name	Description	flexreg()	flexreg_binom()
hyperparam.beta	a positive number specifying the standard deviation for the beta regression coefficients. A value of 100 is suggested if prior.beta = "normal", of 2.5 if prior.beta = "cauchy"	✓	✓
hyperparam.phi	a positive number specifying the hyperprior parameter for the prior distribution of phi. If prior.phi = "gamma" the default is 0.001; If prior.phi = "uniform" the number (no default given) identifies the upper limit of the support of phi	✓	
hyperparam.psi	a positive number specifying the standard deviation for the psi regression coefficients. A value of 100 is suggested if prior.psi = "normal", of 2.5 if prior.psi = "cauchy"	✓	✓
hyperparam.p	a vector of length 2 with positive elements specifying the hyperparameters for the Beta prior distribution of p. If NULL (i.e., the default), a uniform prior distribution is considered	✓	✓
hyperparam.w	a vector of length 2 with positive elements specifying the hyperparameters for the Beta prior distribution of w. If NULL (i.e., the default), a uniform prior distribution is considered	✓	✓
hyperparam.k	a vector of length 2 with positive elements specifying the hyperparameters for the Beta prior distribution of k. If NULL (i.e., the default), a uniform prior distribution is considered	✓	
hyperparam.theta	a vector of length 2 with positive elements specifying the hyperparameters for the Beta prior distribution of theta. If NULL (i.e., the default), a uniform prior distribution is considered		✓

Table 5 (continued)

Name	Description	flexreg()	flexreg_binom()
hyperparam.omega0	a positive number specifying the standard deviation for the omega0 regression coefficients. A value of 100 is suggested if prior.omega0 = "normal", of 2.5 if prior.omega0 = "cauchy"	✓	
hyperparam.omega1	a positive number specifying the standard deviation for the omega1 regression coefficients. A value of 100 is suggested if prior.omega1 = "normal", of 2.5 if prior.omega1 = "cauchy"	✓	

Hyperparameters block

Table 6 Arguments of core functions

Name	Description	flexreg()	flexreg_binom()
n.chain	a positive integer specifying the number of Markov chains (default is 1)	✓	✓
n.iter	a positive integer specifying the number of iterations for each chain (including warm-up). The default is 5000	✓	✓
warmup.perc	the percentage of iterations per chain to discard (default is 0.5)	✓	✓
thin	a positive integer specifying the period for saving samples (default is 1)	✓	✓
verbose	a logical indicating whether to print intermediate output (default is TRUE)	✓	✓

Chain specification block

The third block of arguments (listed in Table 5) refers to the hyperparameters of the prior distributions previously defined in Table 4. Users may freely specify these hyperparameters according to their own prior beliefs; when not explicitly provided, default weakly informative values are used. As for the regression coefficients, which are a priori distributed as either "normal" or "cauchy", the corresponding scale hyperparameters have to be specified in the arguments hyperparam.beta, hyperparam.omega0, hyperparam.omega1, and hyperparam.psi. Recommended values for diffuse, and therefore noninformative, priors are 100 for Normal and 2.5 for Cauchy distributions. The argument hyperparam.phi controls either the hyperparameter g of a Gamma(g, g) prior if prior.phi = "gamma" or the upper-bound hyperparameter A of a Uniform(0, A) prior when prior.phi = "unif". When using flexreg_binom() without regressing the overdispersion parameter θ on covariates, a Beta prior is assigned to θ, parameterized by its mean and precision. These can be specified via the first and second elements of the hyperparam.theta argument, respectively. If hyperparam.theta = NULL (i.e., the default), a uniform prior is assumed. Similarly, mean-precision

parameterized Beta priors are assigned to the parameters p, w, and/or k, with their corresponding hyperparameters specified through the arguments `hyperparam.p`, `hyperparam.w`, and `hyperparam.k`. Setting any of these arguments to `NULL` (the default) results in a uniform prior.

The final block of arguments (listed in Table 6), all of which are common to both the `flexreg()` and `flexreg_binom()` functions, concerns the settings of the HMC approach, among which are the number of chains, their length (i.e., the number of iterations), and the percentage of warm-up, that is, the percentage of elements of the chain(s) to be discarded.

It is undeniable that the large number of arguments available for the two core functions may seem intimidating. However, it is important to note that these functions are designed so that the only mandatory arguments are `formula` and, eventually, `n` (as illustrated in the diagram in Fig. 1). All remaining arguments are optional, have clearly defined default values, and allow users to highly customize both the regression model and the Bayesian setting.

Let us now illustrate the estimation of regression models for bounded continuous and discrete responses by taking advantage of the `Bacteria` dataset (Ascari and Migliorati 2021). The data, originally collected by Demétrio (2014), come from a randomized experiment that evaluates the ability of female insects to parasitize the eggs of an alternative host, varying the total number of female insects. The dataset includes the number of parasitized eggs (`y`) out of a total of 128 eggs (`n`), the proportion of parasitized eggs (`y.perc`), the number of female insects (`females`) and its standardized version (`females_std`). An interesting feature of these data is the presence of zero values in the response (37 out of 70, representing 53% of the data points), while the remaining response values are approximately uniformly distributed between 1 and 120.

The number of parasitized eggs `y` can be understood as a discrete bounded response. Instead, the percentage of parasitized eggs `y.perc` can be handled with a model for bounded continuous data with zero augmentation. For illustrative purposes, we fit an FBB regression model for the response `y` by defining a regression structure for the mean that includes a quadratic function of the covariate `females_std`:

$$\text{logit}(\mu_i) = \beta_0 + \beta_1 \text{females\_std}_i + \beta_2 \text{females\_std}_i^2.$$

This regression model is implemented using the `flexreg_binom()` function as follows:

```
> data("Bacteria")
# Fit of the FBB regression model:
> FBB <- flexreg_binom(y ~ females_std + I(females_std^2), n = "n",
                       data = Bacteria, n.chain = 2, n.iter = 10000, seed = 123)
```

Moreover, we consider an FB regression model for `y.perc` lying on [0, 1), thus including a zero augmentation term. In particular, the regression structure is defined as follows:

$$\begin{cases} \text{logit}(\mu_i) = \beta_0 + \beta_1 \text{females\_std}_i + \beta_2 \text{females\_std}_i^2, \\ \text{logit}(q_{0i}) = \omega_{0,0} + \omega_{0,1} \text{females\_std}_i. \end{cases}$$

Such a regression model is implemented through the `flexreg()` function as follows:

```
#Fit of the FB model with zero augmentation
> FB.aug0 <- flexreg(y.perc ~ females_std + I(females_std^2),
                     zero.formula = ~ females_std, hyperparam.omega0 = 10,
                     data = Bacteria,  n.chain = 2, n.iter = 10000, seed = 123)
```

As mentioned above, the core functions make the choice of the priors straightforward. In the FB example code, we keep the default normal prior for the zero augmentation regression coefficients $\boldsymbol{\omega}_0 = (\omega_{0,0}, \omega_{0,1})^\intercal$; alternatively, a Cauchy prior could be chosen by setting `prior.omega0 = "cauchy"`. Moreover, we modify the scale hyperparameter by setting `hyperparam.omega0 = 10` to illustrate how to specify a less diffuse prior than the default.

3.3 Convergence and posterior checks

In Bayesian analysis of regression models, careful attention should be paid to diagnosing the convergence of the Markov chains to the equilibrium distribution. The statistical literature offers a wide variety of graphical tools as well as diagnostic measures, most of which are available in R packages (e.g., `bayesplot`, `coda` Plummer et al. 2006, and `rstan`, among others). To facilitate this essential step of analysis, our package offers two comprehensive functions that make convergence assessment straightforward.

The first function, `convergence.diag()`, returns some of the most widely used convergence diagnostics (see Gelman et al. (2013) for methodological details).

By default, it also prints key diagnostics concerning the HMC into the console, among which are the number of divergent iterations and the number of transitions after warm-up that exceed the maximum tree depth. The only required argument is `model`, an object of class `flexreg` that is the result of `flexreg()` or `flexreg_binom()`. Optionally, the `diagnostics` argument allows specifying a character vector of diagnostic names (`"Rhat"`, `"geweke"`, `"heidel"`, `"raftery"`, and/or `"gelman"`); however, the default (and our recommendation) is to compute all tests (`diagnostics = "all"`). It is also possible to specify a character vector of parameter names within the `pars` argument; if `pars` is omitted, all parameters in the regression models are evaluated. Finally, the `additional.args` argument enables customization of diagnostic-specific options (see the function documentation for details, omitted here for brevity).

For example, the following code checks the Rhat and Geweke diagnostics for the regression coefficients on the mean model of the FBB regression.

```
# Compute some convergence diagnostics for beta parameters:
> convergence.diag(FBB,  diagnostics = c("Rhat", "geweke"), pars = "beta")

# Divergences:
#   0 of 10000 iterations ended with a divergence.
#
# Tree depth:
#   0 of 10000 iterations saturated the maximum tree depth of 10.
#
# Energy:
#   E-BFMI indicated no pathological behavior.
# Convergence Diagnostics:
#
# [[1]]
# [[1]]$geweke
# [[1]]$geweke[[1]]
#
# Fraction in 1st window = 0.1
# Fraction in 2nd window = 0.5
#
# beta[1] beta[2] beta[3]
# 0.3871  1.3035 -1.0162
#
# [[1]]$geweke[[2]]
#
# Fraction in 1st window = 0.1
# Fraction in 2nd window = 0.5
#
# beta[1]  beta[2]  beta[3]
# 0.70324 -0.07574 -0.18964
#
#
# [[1]]$Rhat
#  beta[1]  beta[2]  beta[3]
# 1.000218 1.000424 1.000469
```

The second function, `convergence.plot()`, produces some convergence plots from the Monte Carlo draws. Similarly to the previous function, the only required argument is `model`, which must be an object of class `flexreg`. The function either saves the plots in a multi-page (one plot per page).pdf file, whose name can be specified in the argument `file`, or displays them directly in the graphics window if `file is NULL`.

Users can select a subset of available plot types using the `plotfun` argument, which accepts a character vector (possible options are `density`, `trace`, `intervals`, `rate`, `rhat`, and `acf`). The `pars` argument works identically as in `convergence.diag()` to select a subset of model parameters. Finally, some additional arguments are available to customize the graphical representations and to specify the name, width, and height of the.pdf file. Figure 3 shows the convergence plots produced by the `convergence.plot()` function for the parameter ϕ of the FBB regression model.

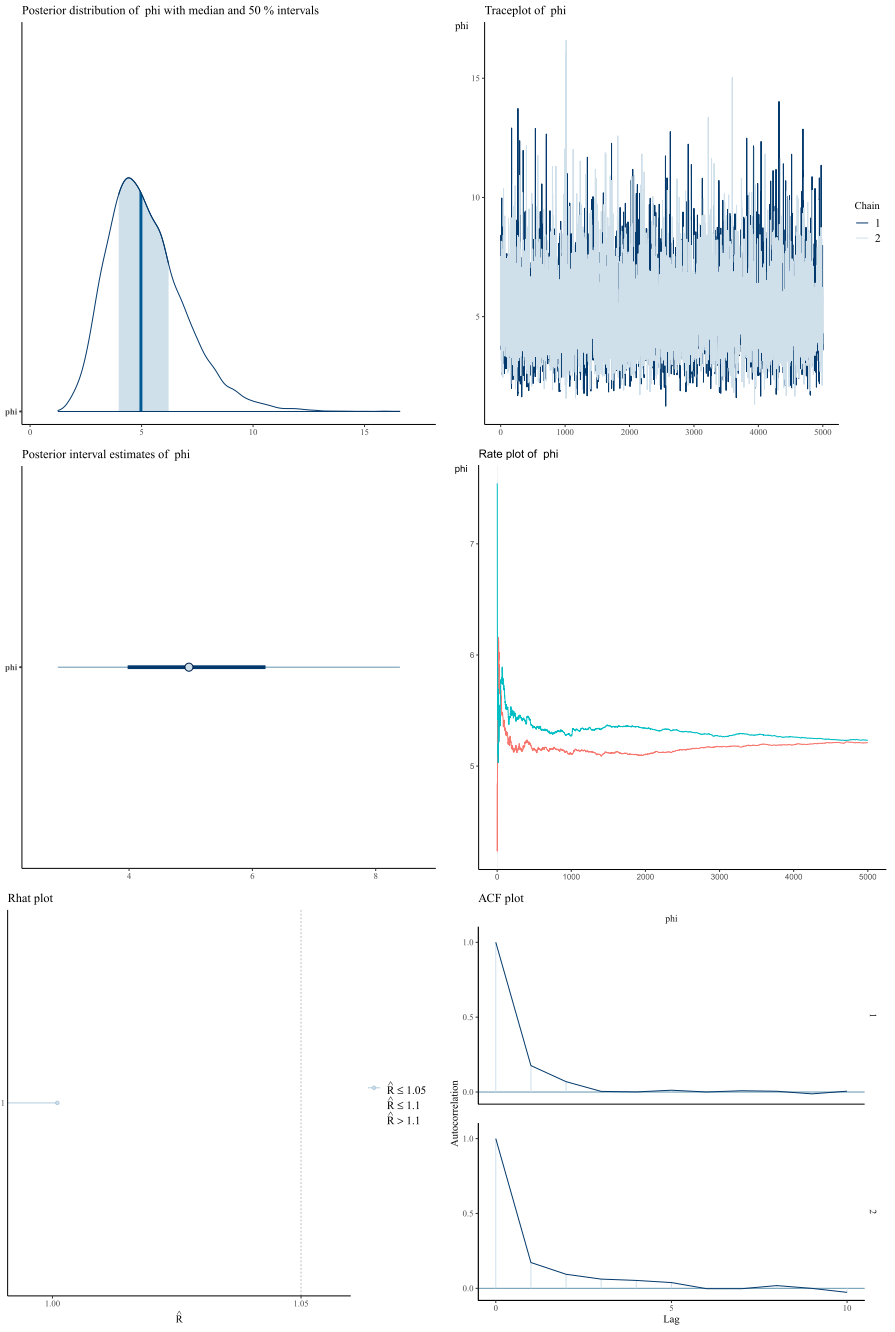

Fig. 3 Plots generated with the following code `convergence.plot(FBB, file = "ConvergenceOutput.pdf", pars = "phi")`

```
# Producing a .pdf file containing convergence plots for the parameter
# phi of the FBB regression model:
> convergence.plot(FBB, file = "ConvergenceOutput.pdf", pars = "phi")
```

3.4 Functions to explore objects of class flexreg

The `FlexReg` package provides a comprehensive suite of functions and methods for regression model analysis, including tools for assessing goodness of fit, performing posterior predictive checks, generating model summaries, and visualizing regression curves. To ensure consistency with common R modeling practices, `FlexReg` also supports methods such as `coef()`, `model.matrix()`, and `model.frame()`, which are aligned with those used with `lm()` and `glm()` objects. These functions serve to extract model coefficients, construct the design matrix, and retrieve the data used for model fitting, respectively. In the following subsections, the other available functions will be described in detail.

3.4.1 Predictions and residuals

Two fundamental S3 methods for the analysis of linear models, `predict()` and `residuals()`, have been adapted to the context of bounded regression models within a Bayesian framework. The `predict()` method applied to an object of class `flexreg` returns a `data.frame` whose columns contain various types of predictions depending on the argument `type`. The available options are:

- `"response"`: marginal fitted means of response/relative response;
- `"link"`: linear predictors for the mean;
- `"precision"`: fitted precisions;
- `"overdispersion"`: fitted overdispersions;
- `"variance"`: fitted variances.

Based on the simulated Markov Chains, $\boldsymbol{\eta}^{(s)}, s = 1, \ldots, S$, the predicted values are computed for each $\boldsymbol{\eta}^{(s)}$ (i.e., for each element of the chain) and for each observation. To summarize the results, the function returns an aggregation of these values according to the argument `estimate`, which can be either `"mean"` (default), `"median"`, or `"quantile"` (of level `q`, to be specified by the user).

If `type = "response"`, the predicted values are equal to $\widehat{\mu}_i^{(s)} = g_1^{-1}\left(\mathrm{x}_i\boldsymbol{\beta}^{(s)}\right)$ (see Eq. (7)).

For discrete responses, the pseudo-counts $n_i\widehat{\mu}_i$ are also included, as shown in the example below.

```
# Predict method for the FBB regression model. The output includes the
# response (response) and the pseudo-counts (response.binom)

> predict(FBB, type = "response")
#    response.binom  response
# 1        18.34842 0.1433470
# 2        19.31514 0.1508995
# 3        21.29406 0.1663598
# ...
```

Instead, for continuous data with augmentation, a predicted response that also accounts for the augmentation is returned, which is equal to

$$\widehat{\mu}_{i,aug}^{(s)} = \widehat{q}_{1i}^{(s)} + \widehat{q}_{2i}^{(s)} \widehat{\mu}_i^{(s)}, \tag{11}$$

where $\widehat{q}_{0i}^{(s)} = g_3^{-1}\left(\mathrm{x}_{i0}^{\intercal}\boldsymbol{\omega}_0^{(s)}\right)$, $\widehat{q}_{1i}^{(s)} = g_4^{-1}\left(\mathrm{x}_{i1}^{\intercal}\boldsymbol{\omega}_1^{(s)}\right)$ (see Eq. (7)), and $\widehat{q}_{2i}^{(s)} = 1 - \widehat{q}_{0i}^{(s)} - \widehat{q}_{1i}^{(s)}$. Moreover, the probabilities of augmentation in zero and/or one are also returned.

```
# Predict function for the augmented FB model with augmentation in zero.
# The output includes the augmented response (response.aug),
# the response (response), and the probability of augmentation in zero (q0).

> predict(FB.aug0, type = "response")
#    response.aug  response        q0
# 1     0.1062247 0.2440154 0.5645555
# 2     0.1172087 0.2680012 0.5625489
# 3     0.1405476 0.3183990 0.5585058
# ...
```

If `type = "link"`, the function computes the linear predictors for the mean $\mathrm{x}_i^{\intercal}\boldsymbol{\beta}^{(s)}$.

When `type = "precision"`, an option available only for continuous models fitted with the `flexreg()` function, the method computes the predicted values for the precision parameter ϕ, that is $\widehat{\phi}_i^{(s)} = \phi^{(s)}$ if the precision parameter has not been regressed onto covariates, and $\widehat{\phi}_i^{(s)} = g_2^{-1}\left(\mathrm{z}_i^{\intercal}\boldsymbol{\psi}^{(s)}\right)$ otherwise (see Eq. (7)).

For BB and FBB regression models, which handle discrete responses with overdispersion, the option `type = "overdispersion"` is available. In this case, the `predict()` function returns $\widehat{\theta}_i^{(s)} = \theta^{(s)}$ if the overdispersion parameter is not regressed onto covariates, and $\widehat{\theta}_i^{(s)} = g_2^{*-1}\left(\mathrm{z}_i^{\intercal}\boldsymbol{\psi}^{(s)}\right)$ otherwise (see Eq. (10)). Finally, setting `type = "variance"` allows to obtain the predicted variances. Their expression depends on the chosen model (see Sect. 2), while the predictions are obtained by plug-in methods. By way of example, in the Beta model, the s-th value of the fitted variance for observation i is equal to

$$\widehat{\mathrm{Var}}^{(s)}(Y_i) = \frac{\widehat{\mu}_i^{(s)}(1-\widehat{\mu}_i^{(s)})}{\widehat{\phi}_i^{(s)}+1}.$$

Please note that for continuous responses with augmentation term(s), the expression of the variance to be plugged in is

$$(1-q_{0i}-q_{1i})\mathrm{Var}\,(Y_i|0<Y_i<1)+q_{1i}^2+(1-q_{0i}-q_{1i})\mu_i^2-(q_{1i}+(1-q_{0i}-q_{1i})\mu_i)^2.$$

Moreover, it is possible to set the argument `cluster = TRUE`, which works only for `"FB"`, `"VIB"`, and `"FBB"` regression models and for `type` either `"response"` or `"variance"`. If `type = "response"` and `cluster = TRUE`, the function returns the predicted component-specific means $\widehat{\lambda}_{1i}$ and $\widehat{\lambda}_{2i}$ (see Eq. (4)). For `"VIB"` models these values are omitted, since they coincide with the conditional mean. Otherwise, if `type = "variance"` and `cluster = TRUE`, the function returns the component-specific variances together with the overall variance.

Finally, in line with the standard `predict()` methods for linear and generalized linear models, the argument `newdata` can be specified as a `data.frame` containing the variables to be used for prediction. This `data.frame` must include all covariates appearing in the regression models specified in `formula`, `zero.formula`, and `one.formula`. For regression models with bounded discrete responses, the additional argument `n.new` is required to provide the total number of trials for prediction. The vector supplied to `n.new` must have length equal to `nrow(newdata)`. If `newdata` is omitted or set to `NULL` (the default), the fitted values are used to get the predictions.

```
# Prediction of overdispersion for FBB regression model:
> predict(FBB, type = "overdispersion")
#   overdispersion
# 1      0.1732488

> newdata <- data.frame("females_std" = c(-2.5, -1, 3))

# Prediction of response for FBB regression model for new observations:
> predict(FBB, type="response", newdata = newdata,
          n.new = rep(128,3), cluster = TRUE)
#   response.binom    response         l1           l2
# 1       1.231451 0.009620709 0.02173535 0.0002496757
# 2      14.480770 0.113131015 0.24851008 0.0023281547
# 3       9.628205 0.075220350 0.16526231 0.0015618964

# Prediction of variance for FB regression model with zero augmentation
# for new observations:
> predict(FB.aug0, type="variance", newdata = newdata, cluster = TRUE)
#      variance     cluster1     cluster2
# 1 0.000150784 0.0006702281 0.0002126801
# 2 0.016222059 0.0251786621 0.0160520892
# 3 0.010483704 0.0156675597 0.0088495713
```

The `residuals()` method applied to a `flexreg` object computes raw or standardized residuals. Similarly to `predict()` method, all computations in the `residuals()` function are performed for each element of the simulated Markov Chains and then returned in an aggregated way according to the argument `estimate`, which can be set to `"mean"` (default), `"median"`, or `"quantile"` with a numeric `q` denoting the quantile level. The raw and standardized residuals for observation i in the s-th element of the chain are defined as

$$r_i^{(s)} = \tilde{y}_i - \widehat{\mu}_i^{(s)} \quad \text{and} \quad \check{r}_i^{(s)} = \frac{r_i^{(s)}}{\sqrt{\widehat{\text{Var}}^{(s)}(Y_i)}},$$

respectively, where $\tilde{y}_i$ is equal to y_i for continuous data and to y_i/n_i for binomial data, $i = 1, \ldots, N$. Quantities $\widehat{\mu}_i^{(s)}$ and $\widehat{\text{Var}}^{(s)}(Y_i)$, correspond to the predicted mean and variance, which are computed internally by means of the method `predict()` with `type = "response"` and `type = "variance"`, respectively. Please note that for continuous regression models with augmentation, raw residuals are computed as $r_i^{(s)} = y_i - \widehat{\mu}_{i,aug}^{(s)}$ where $\widehat{\mu}_{i,aug}^{(s)}$ is the predicted response as described in (11). Moreover, if the model is of type `FB` or `FBB` and `cluster = TRUE`, in addition to the raw or standardized residuals, the `residuals()` method returns component-specific residuals that are computed with respect to the cluster means as follows:

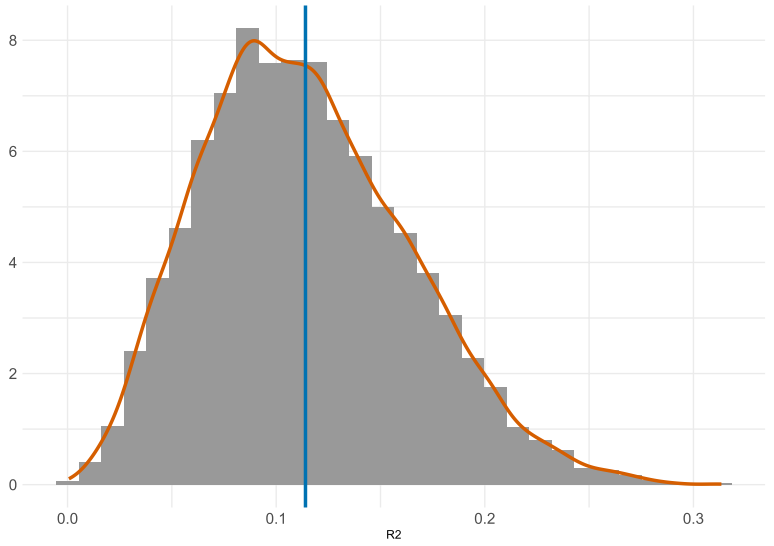

Fig. 4 Histogram and density plot (orange line) of the Bayesian R^2 for the FB regression model with zero augmentation, the output of `R2_bayes()` function. The vertical blue line represents the posterior mean of R^2

$$r_{ji}^{(s)} = \tilde{y}_i - \widehat{\lambda}_{ji}^{(s)} \quad \text{and} \quad \breve{r}_{ji}^{(s)} = \frac{r_{ij}^{(s)}}{\sqrt{\widehat{\text{Var}}_j^{(s)}(Y_i)}}, \quad i = 1, \ldots, N, \quad j = 1, 2,$$

where $\widehat{\text{Var}}_j^{(s)}(Y_i)$ for $j = 1, 2$ are the cluster variances, while $\widehat{\lambda}_{1i}^{(s)}$ and $\widehat{\lambda}_{2i}^{(s)}$ are the predicted component-specific means and are computed by plugging in from (4):

$$\widehat{\lambda}_{1i}^{(s)} = \widehat{\mu}_i^{(s)} + (1 - p_i^{(s)})\tilde{w}_i^{(s)} \quad \text{and} \quad \widehat{\lambda}_{2i}^{(s)} = \widehat{\mu}_i^{(s)} - p_i^{(s)}\tilde{w}_i^{(s)}.$$

The `residuals()` method returns a `data.frame` whose columns contain the raw or standardized residuals and, possibly, the cluster residuals. Moreover, when `cluster = TRUE`, the final output of the function also comprises a column named `label` assuming values 1 or 2, for each observation, depending on which is the smallest corresponding cluster residual (either raw or standardized). Interestingly, this output can be used as a naive clustering approach based on the regression model, taking advantage of the FB-type mixture structure.

```
# Computing the raw residuals (default) for the FBB regression model:
> residuals(FBB, cluster = TRUE)
#             raw    cluster1     cluster2 label
# 1   0.231652968  0.06095324  0.372091433     1
# 2  -0.143086999 -0.32259545  0.004759813     2
# 3   0.341452668  0.14393083  0.504465774     1
# ...

# Computing the standardized residuals for the FB regression model
# with zero augmentation:
> residuals(FB.aug0, type = "standardized", cluster = TRUE)
#    standardized    cluster1   cluster2 label
# 1    0.76662390  0.23520483  1.2622797     1
# 2   -1.42820515 -1.90969181 -1.2095547     2
# 3    0.92178643  0.51697735  1.5759146     1
# ...

# Computing the standardized residuals for the FBB regression model:
> residuals(FBB, type = "standardized", cluster = TRUE)
#     standardized      cluster1      cluster2 label
# 1   0.0112517785  0.0024521609  0.1269051572     1
# 2  -0.0067713608 -0.0127938014  0.0015845040     2
# 3   0.0153784245  0.0055703004  0.1603661424     1
# ...
```

3.4.2 Bayesian goodness of fit measures and posterior predictive checks

As for the model comparisons and goodness of fit evaluations, `FlexReg` provides the functions `WAIC()` and `R2_bayes()`, both requiring as their only argument an object (or a list of objects) of class `flexreg`.

The `R2_bayes()` function computes a Bayesian version of R^2, which is defined as the ratio between the variance of the predicted values and the sum of the variance of the predicted values plus the expected residual variance (Gelman et al. 2019). The function returns the posterior distribution of Bayesian R^2, whose values are between 0 and 1; its posterior mean can be interpreted, in agreement with the classical R^2, as the proportion of the variability of the response explained by the regression model (as an example, see Fig. 4). The `WAIC()` function computes the widely applicable information criterion (WAIC) and efficient approximate leave-one-out cross-validation (LOO). This function is a wrapper of functions `waic()` and `loo()` from the `loo` package, specifically adapted to work with objects of class `flexreg`. WAIC values can be used to compare the fit of competing models, with the interpretation being that the lower the value, the better the model (Fig. 5).

```
# Bayesian R^2:
> R2_Bayes <- R2_bayes(FB.aug0)
> mean(R2_Bayes[[1]])
# [1] 0.1140404

# Goodness of fit measures:
> WAIC_measures <- WAIC(FB.aug0)
> WAIC_measures$waic_out
# Computed from 10000 by 70 log-likelihood matrix.
#
#           Estimate  SE
# elpd_waic    -43.9 2.4
# p_waic         6.4 1.1
# waic          87.8 4.8
#
# 3 (4.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.

> WAIC_measures$loo_out
# Computed from 10000 by 70 log-likelihood matrix.
#
#          Estimate  SE
# elpd_loo    -44.0 2.4
# p_loo         6.5 1.2
# looic        88.1 4.9
# ------
# MCSE of elpd_loo is 0.0.
# MCSE and ESS estimates assume independent draws (r_eff=1).
#
# All Pareto k estimates are good (k < 0.7).
# See help('pareto-k-diagnostic') for details.
```

The `posterior_predict()` method takes an object of class `flexreg` and generates values from the posterior predictive distribution, which is the distribution of a future outcome given the observed data (Gelman et al. 2013). The posterior predictive distribution is computed for y in the case of bounded continuous responses and for y/n in the case of bounded discrete responses. The optional argument `newdata` allows specifying a `data.frame` containing variables with which to per-

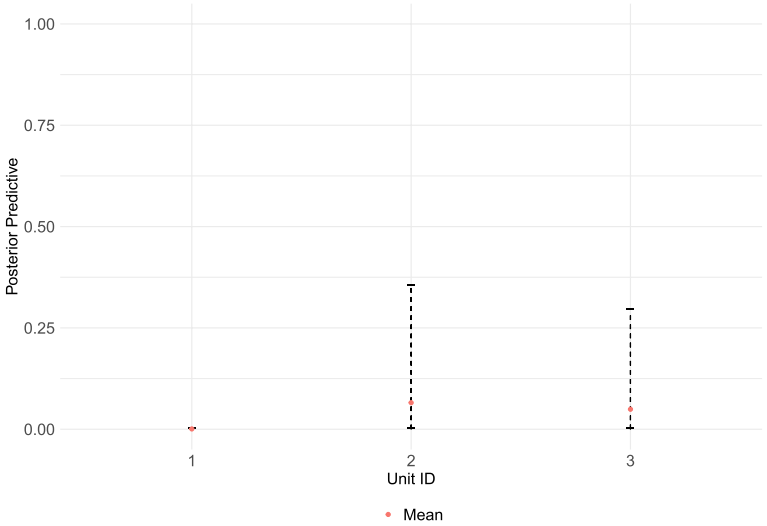

Fig. 5 Posterior predicted means (red points) and 90% predictive intervals for three new observations under the FB regression model with zero augmentation

form the prediction. If `newdata` is omitted, the fitted values are used. The function returns an object of class `flexreg_postpred` that has its own `plot()` and `summary()` methods. Optional arguments for the `plot()` function are `response`, either a numerical vector containing the responses to be added to the plot, or `NULL` if no responses should be displayed. In addition, `prob` is a numeric argument indicating the probability level of the interval for the posterior predictive (the default is 0.9), and `p_mean` is a logical value indicating whether the posterior predictive mean should be plotted or not.

```
# Posterior predictives for the augmented FB regression model:
> set.seed(42)
> postpred_new <- posterior_predict(FB.aug0, newdata = newdata)

> summary(postpred_new)
#   Min 1st Qu.       Median         Mean       3rd Qu.       Max
# 1   0       0 0.000000e+00 0.0007289186 3.396162e-312 0.4855410
# 2   0       0 0.000000e+00 0.0655581130  7.512136e-02 0.9514171
# 3   0       0 2.652743e-05 0.0494104921  3.883679e-02 0.9999821

# Plot of posterior predictive intervals at 90% confidence bound in Fig. 5:
> plot(postpred_new, p_mean = TRUE)
```

3.4.3 Summaries and visualization functions

The `summary()` method applied to an object of class `flexreg` returns some relevant summary statistics, as well as a summary of raw residuals and WAIC measures.

```
# Summary method for object of class flexreg:
> summary(FBB)
# Call: flexreg_binom(formula = y ~ females_std + I(females_std^2),
#                     data = Bacteria, n = "n",
#                     n.chain = 2, n.iter = 10000, seed = 123)
#
# Model name:  FBB
#
# Residuals:
#   Min      1Q  Median      3Q     Max
# -0.3303 -0.1980 -0.1433  0.2320  0.6789
#
# Coefficients (mean model with logit link):
#                   Post. Mean Post. SD    2.5% Post. Median   97.5%
#   (Intercept)         -0.9913   0.2230 -1.4421      -0.9866 -0.5618
#   females_std          0.6892   0.2170  0.2765       0.6825  1.1364
#   I(females_std^2)    -0.4151   0.1271 -0.6717      -0.4118 -0.1706
#
# Overdispersion parameter(s):
#         Post. Mean Post. SD    2.5% Post. Median  97.5%
#   theta     0.1732   0.0481  0.0976       0.1675 0.2835
#   phi       5.2218   1.7454  2.5277       4.9708 9.2466
#
# Additional Parameters:
#     Post. Mean Post. SD   2.5% Post. Median  97.5%
#   p     0.4545   0.0624 0.3335       0.4540 0.5776
#   w     0.9798   0.0161 0.9384       0.9836 0.9990
#
# Waic method:
#   Computed from 10000 by 70 log-likelihood matrix.
#
#           Estimate   SE
# 3 (4.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.

# elpd_waic   -204.2 20.0
# p_waic         5.0  0.9
# waic         408.4 40.1
#
# 1 (1.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.

> summary(FB.aug0)
# Call: flexreg(formula = y.perc ~ females_std + I(females_std^2),
#               zero.formula = ~ females_std, data = Bacteria,
#               hyperparam.omega0 = 10, n.chain = 2,
#               n.iter = 10000, seed = 123)
#
# Model name:  FB with zero augmentation
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -0.7541 -0.4227 -0.2524  0.0218  0.3366
#
# Coefficients (mean model with logit link):
#                     Post. Mean Post. SD    2.5% Post. Median   97.5%
#   (Intercept)          0.5707   0.2846  0.0251       0.5667  1.1414
#   females_std          1.4616   0.3618  0.7591       1.4600  2.1853
#   I(females_std^2)    -0.8533   0.2302 -1.3195      -0.8499 -0.4088
#
# Coefficients (precision model with identity link for phi):
#       Post. Mean Post. SD   2.5% Post. Median   97.5%
#   phi     5.7887   2.0534 3.0564       5.3699 11.1412
#
# Coefficients (zero augmentation model with multinomial logit link):
#               Post. Mean Post. SD    2.5% Post. Median  97.5%
#   (Intercept)     0.1181   0.2430 -0.3559       0.1168 0.5885
#   females_std    -0.1844   0.2524 -0.6897      -0.1797 0.3061
#
# Additional Parameters:
#     Post. Mean Post. SD   2.5% Post. Median  97.5%
#   p     0.5267   0.3086 0.0177       0.5501 0.9860
#   w     0.3019   0.2112 0.0133       0.2710 0.8188
#
# Waic method:
#   Computed from 10000 by 70 log-likelihood matrix.
#
#           Estimate  SE
# elpd_waic    -43.9 2.4
# p_waic         6.4 1.1
# waic          87.8 4.8
#
# 3 (4.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

The `plot()` method for `flexreg` objects returns a graphical representation of the regression curves from a fitted model. The regression curves are plotted against a selected quantitative covariate from the the mean model, while all other covariates are fixed at specified values. The graph also shows the scatterplot of the covariate and the response. The function requires two arguments: `x`, an object of class `flexreg`, and `name.x`, the name of the covariate from the mean model to be plotted on the x-axis. If additional covariates for the mean model are present, their default values can be specified through the argument `additional.cov.default`, a list such as `additional.cov.default = list("x2" = 0, "x3" = -1)`.

By default, the function returns the regression curve for the mean (i.e., the curve representing $\widehat{\mu}$) due to the argument `type = "response"`. For augmented models, it is possible to plot the regression curve for the augmented response (i.e., $\widehat{\mu}_{aug}$), either in addition to or instead of the regression curve for the mean. To do this it is sufficient to set `type = c("response", "response.aug")` or `type = "response.aug"`, respectively. For FB and FBB regression models, it is possible to set the argument `cluster = TRUE` to plot the regression curves of the cluster means, $\widehat{\lambda}_1$ and $\widehat{\lambda}_2$. Lastly, the optional logical argument `smooth` controls the smoothness of the plotted curves. When `smooth = FALSE`, the function displays the observed regression curves, represented as piecewise linear functions that connect the predicted values of the observed values of the covariate specified in `name.x`. Otherwise, when `smooth = TRUE` (the default), a smoothed version of the regression curves is plotted.

Please note that the `plot()` method for `flexreg` class objects returns an object of class `ggplot`, so that the user can customize the resulting plot through standard `ggplot2` functions. The following code produces the plots in Figs. 6 and 7.

```
# The following code returns the plot in Fig.  6:
> plot(FBB, name.x = "females_std", cluster = TRUE) +
        labs(x = "Standardized Females")

# The following code returns the plot on the left-hand side of Fig. 7:
> plot(FB.aug0, name.x = "females_std", smooth = FALSE, cluster = TRUE,
        type = c("response", "response.aug")) +
        labs(x = "Standardized Females")

# The following code returns the plot on the right-hand side of Fig. 7:
> plot(FB.aug0, name.x = "females_std", cluster = TRUE,
        type = c("response", "response.aug")) +
        labs(x = "Standardized Females")
```

4 Conclusions

The growing occurrence of bounded response variables in applied statistical analyses has generated increasing interest in appropriate regression models. Besides some well-established models such as the beta, binomial, and BB regression models, some recent proposals have gained popularity due to their flexibility in handling non-standard features. The `FlexReg` package implements all these models within a uni-

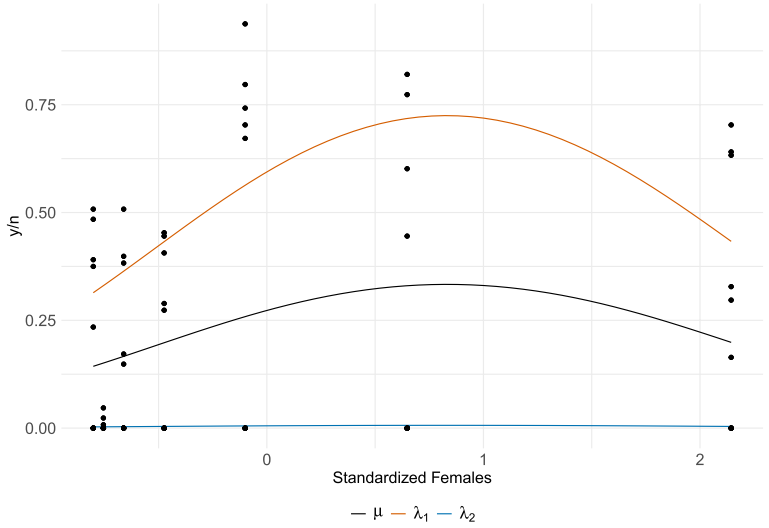

Fig. 6 Graphical representation of the smoothed regression curves of the mean (black line) and cluster means (red and blue lines) from the FBB model

fied framework, allowing for the analysis of both continuous and discrete bounded responses.

From a methodological perspective, the package not only implements the established results concerning the flexible distributions for bounded data and their regression models, but also introduces some novelties, including the definition of quantile and cumulative distribution functions, a more efficient estimation procedure for zero and/or one augmented models, and the specification of several types of predicted and residual values. The package is designed to be accessible to practitioners through carefully chosen default settings, while also offering experienced users the possibility to tailor key modeling components (e.g., link functions, priors' and hyperparameters' specifications) and to perform in-depth analyzes (e.g., convergence diagnostics and model fit assessment).

Future extensions of the `FlexReg` package may be outlined to enrich the current version. In particular, all the regression models considered in this paper could be extended to include a hierarchical (multilevel) structure, allowing for the proper modeling of clustered or hierarchical data, as well as spatial and/or temporal regression structures. Moreover, although it has been shown that the FBBReg model can allocate one of its mixture components to capture excess of zeros (Ascari and Migliorati 2021), the explicit inclusion of zero-inflated versions of both the BBReg and FBBReg models would further broaden the range of models that the `FlexReg` package could accommodate.

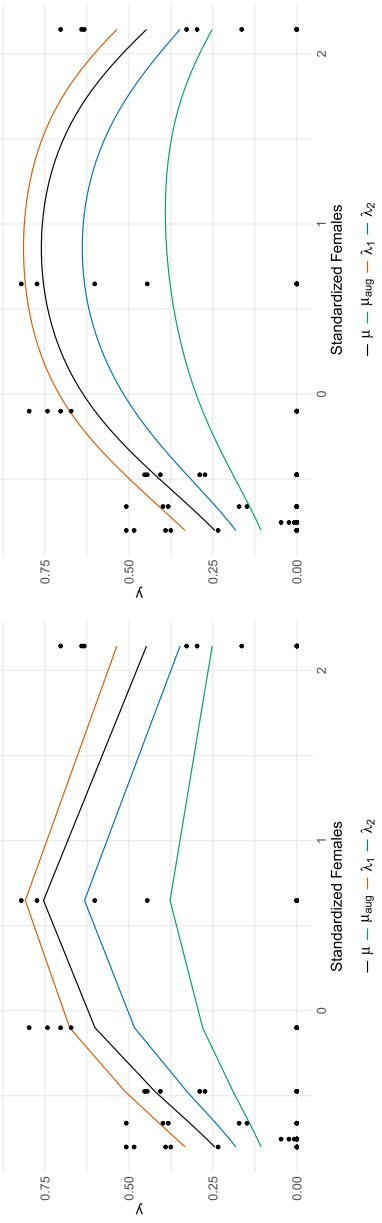

Fig. 7 Graphical representation of the not smoothed (left) and smoothed (right) regression curves of the mean (black line), the augmented response (green line), and cluster means (red and blue lines) from the FB model with zero augmentation

Acknowledgements This research was partially financially supported by the University of Milano-Bicocca.

Funding Open access funding provided by Università degli Studi di Milano - Bicocca within the CRUI-CARE Agreement.

References

Aitchison J (2003) The statistical analysis of compositional data, 2nd edn. The Blackburn Press, London

Ascari R, Migliorati S (2021) A new regression model for overdispersed binomial data accounting for outliers and an excess of zeros. Stat Med 40(17):3895–3914. https://doi.org/10.1002/sim.9005

Ascari R, Migliorati S (2022) A new regression model for count compositions. in: data analysis and related applications, volume 2: Multivariate, Health and Demographic Data Anal. Wiley, https://doi.org/10.1002/9781394165544.ch2

Cribari-Neto F, Zeileis A (2010) Beta regression in R. J Stat Softw 34(2):1–24. https://doi.org/10.18637/jss.v034.i02

Demétrio CGB, Hinde J, Moral RA (2014) Models for Overdispersed Data in Entomology, Springer International Publishing, Cham, pp 219–259. https://doi.org/10.1007/978-3-319-06877-0_9

Di Brisco AM, Migliorati S (2019) A new mixed-effects mixture model for constrained longitudinal data. Stat Med 39(2):129–145. https://doi.org/10.1002/sim.8406

Di Brisco AM, Migliorati S (2020) A spatial mixed-effects regression model for electoral data. Stat Methods Appl 30(2):543–571. https://doi.org/10.1007/s10260-020-00534-6

Di Brisco AM, Migliorati S, Ongaro A (2020) Robustness against outliers: a new variance inflated regression model for proportions. Stat Model 20(3):274–309. https://doi.org/10.1177/1471082X18821213

Ferrari SLP, Cribari-Neto F (2004) Beta regression for modelling rates and proportions. J Appl Stat 31(7):799–815. https://doi.org/10.1080/0266476042000214501

Gabry J, Simpson D, Vehtari A et al (2019) Visualization in Bayesian workflow. J R Stat Soc Ser A Stat Soc 182(2):389–402. https://doi.org/10.1111/rssa.12378

Gabry J, Goodrich B, Lysy M, et al (2023) rstantools: Tools for Developing R Packages Interfacing with 'Stan'. https://CRAN.R-project.org/package=rstantools, r package version 2.3.1.1

Gelman A, Carlin JB, Stern HS et al (2013) Bayesian data analysis. CRC Press

Gelman A, Goodrich B, Gabry J et al (2019) R-squared for Bayesian regression models. Am Stat 73(3):307–309. https://doi.org/10.1080/00031305.2018.1549100

Hinde J, Demétrio CGB (1998) Overdispersion: models and estimation. Comput Stat Data Anal 27:151–170. https://doi.org/10.1016/S0167-9473(98)00007-3

Lesnoff M, Lancelot R (2012) Analysis of Overdispersed Data. https://cran.r-project.org/web/packages/aod/aod.pdf, R package version 1.1.32

Liu F, Kong Y (2015) zoib: an R package for Bayesian inference for beta regression and zero/one inflated beta regression. R J 7:34–51. https://doi.org/10.32614/RJ-2015-019

Migliorati S, Di Brisco AM, Ongaro A (2018) A new regression model for bounded responses. Bayesian Anal 13(3):845–872. https://doi.org/10.1214/17-BA1079

Ospina R, Ferrari SL (2012) A general class of zero-or-one inflated beta regression models. Comput Stat Data Anal 56(6):1609–1623. https://doi.org/10.1016/j.csda.2011.10.005

Plummer M, Best N, Cowles K et al (2006) CODA: convergence diagnosis and output analysis for MCMC. R News 6(1):7–11

Stan Development Team (2022) Stan modeling language users guide and reference manual, version 2.32. https://mc-stan.org
Stan Development Team (2023) RStan: the R interface to Stan. https://mc-stan.org/, R package version 2.21.8
Vehtari A, Gelman A, Gabry J (2017) Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Stat Comput 27:1413–1432. https://doi.org/10.1007/s11222-016-9696-4
Wickham H (2016) ggplot2: elegant graphics for data analysis. Springer-Verlag New York, https://ggplot2.tidyverse.org
Williams DA (1975) The analysis of binary responses from toxicological experiments involving reproduction and teratogenicity. Biometrics 31:949–952. https://doi.org/10.2307/2529820
Yee TW (2025) Vector Generalized Linear and Additive Models: With an Implementation in R. Springer

Authors and Affiliations

Roberto Ascari[1] · Agnese Maria Di Brisco[2] · Sonia Migliorati[1] · Andrea Ongaro[1]

✉ Roberto Ascari
roberto.ascari@unimib.it

Agnese Maria Di Brisco
agnese.dibrisco@uniupo.it

Sonia Migliorati
sonia.migliorati@unimib.it

Andrea Ongaro
andrea.ongaro@unimib.it

[1] Department of Economics, Management and Statistics, University of Milano-Bicocca, Piazza dell'Ateneo Nuovo, 1, 20126 Milan, Italy

[2] Department of Studies for Economics and Business, University of Piemonte Orientale, Via Ettore Perrone 18, 28100 Novara, Italy