Bayesian Discretised Beta Regression for Analysis of Ratings Data: The R Package DBR

Mansour T.A. Sharabiani
School of Public Health
Imperial College London, UK

Alireza S. Mahani
Davison Kempner Capital Management
New York, USA

Cathy M. Price
Solent NHS Trust
Southampton, UK

Alex Bottle
School of Public Health
Imperial College London, UK

Abstract

The question of whether to treat ratings data - often generated from survey responses - as ordinal or numeric has received considerable attention over the years. Theoretical arguments notwithstanding, when the number of response levels is high, practitioners often seek a numeric interpretation of the response variable and the effect of predictors on 'mean' response. While linear regression is a convenient option, its implicit assumptions of unbounded response, strict linearity, and homoscedasticity are unrealistic, when applied to ratings responses. In this paper, we introduce the R package **DBR** for Discretised Beta Regression analysis of ratings data. **DBR** implements an adaptation of beta regression with several enhancements. First, it handles the forward and backward mapping between the observed range of responses and the standard range of the beta distribution (from zero to one). Secondly, it accounts for the discrete nature of observations by using cumulative-density terms in constructing the likelihood function. Thirdly, **DBR** represents extreme-value count inflation, often seen in survey responses, both in estimation and prediction steps. Finally, by adopting a Bayesian framework using Markov Chain Monte Carlo sampling for estimation, **DBR** benefits from robust estimation, credible-interval calculation and prediction functionalities. To facilitate model interpretation, and in addition to returning the standard coefficients in the internal beta regression model, a second function summarises and visualises the nonlinear impact of each predictor on the observed (ratings) outcome. We hope that the **DBR** R package offers a viable alternative - based on realistic assumptions about the data generation process - for numeric analysis of ratings data by practitioners.

Keywords: Discretised Beta Regression, Ordinal Regression, Likert, Bayesian, Markov Chain Monte Carlo.

1. Introduction

When analysing survey-response data, a key decision is whether the data should be treated as nominal, ordinal or numeric. When there is no natural order in responses, the data should clearly be treated as nominal. An example would be the type of lung cancer detected in a patient. Choice models such as multinomial logit (Hasan, Wang, and Mahani 2016) and probit are suitable for regression analysis of nominal response variables. If responses present a natural order but do not carry a clear numeric interpretation (ordinal data), one can use ordered logit and probit regression models (Goodrich, Gabry, Ali, and Brilleman 2018). An example would be a patient's degree of happiness in sending their child to school after a prolonged period of remote learning.

The third type of survey response - referred to as ratings data - is similar to ordinal data, but contains more levels. Examples of rating scales - used to elicit rating responses - are numeric, Likert (average of multiple numeric responses), fully-anchored (numbers with associated, descriptive text) and adjectival (e.g. 'poor', 'ok', 'good' and 'excellent') (Harpe 2015). When it comes to ratings data, there has been considerable debate about whether the responses should be treated as ordinal or numeric (Harpe 2015; Liddell and Kruschke 2018; Jamieson 2004; Norman 2010; Kuzon, Urbanchek, and McCabe 1996; Armstrong 1981; Knapp 1990; Pell 2005; Carifio and Perla 2007, 2008). Numeric treatment of ratings data allows for easier interpretation of regression coefficients, but has been shown to lead to inconsistent results (Liddell and Kruschke 2018) when there are few levels. When dealing with many levels, the numeric treatment has the advantage of consuming significantly fewer degrees of freedom compared with ordinal regression, but the underlying assumptions of strict linearity, unboundedness and homoscedasticity remain at odds with the nature of ratings data.

In this paper, we present the open-source **DBR** R package for Discretised Beta Regression (DBR) analysis of ratings data. **DBR** offers a middle ground between linear regression - built on a strict equidistant interpretation of the response scale - and ordinal regression with full flexibility in partitioning a latent variable into sub-regions that are mapped to the observed, discrete levels.

DBR is an adaptation of beta regression, following the specification of Ferrari and Cribari-Neto (2004); Zeileis, Cribari-Neto, Grün, and Kos-midis (2010). It is similar to ordinal regression, especially the ordered probit model, in that it maps a continuous, latent variable to the observed discrete response by partitioning the range of the latent variable. However, DBR has two important differences from ordered probit regression: 1- the underlying distribution is assumed to be beta (with proper shift and scale factors applied) rather than normal, 2- cutoff points in DBR are assumed to be halfway points between the observed values, rather being estimated from data. (However, see the discussion of left and right buffers in Section 2.4) This ensures that the number of model parameters does not scale linearly with the number of levels in the ratings scale, thus reducing the risk of overfitting compared with ordinal regression. Overall, this setup allows DBR to create a numeric yet realistic interpretation of ratings data. (Note that ordinal regression in R can be done using the `polr` function in the **MASS** package (Venables and Ripley 2002).)

DBR is similar to beta-binomial regression, which has also been recommended for the analysis of ratings data (Najera-Zuloaga, Lee, and Arostegui 2018). There are differences, however: first, rather than directly mapping responses to a discrete distribution (binomial or beta-binomial), DBR follows the latent-variable approach in ordinal regression, which is more in line with our intuition about the process of response selection by survey respondents. Secondly, the `DBR` software accounts for extreme-value count inflation using cumulative-density terms in the log-likelihood function. (Note that beta-binomial regression in R is available via the **aod** package (Lesnoff, M., Lancelot, and R. 2012), or the **PROreg** package (Najera-Zuloaga, Lee, and Arostegui 2020).)

The mathematical framework of DBR is similar to that of Inflated Discrete Beta Regression (IDBR) in Taverne and Lambert (2014), with three differences. First, DBR uses flexible endpoints for cutting the beta distribution, while IDBR uses mass density in discrete space. Secondly, DBR only applies regression (on covariates) to the mean response, while IDBR also models the impact of covariates on the dispersion parameter and inflated response. Thirdly, DBR uses a Gibbs wrapper around the univariate slice sampler (Neal 2003) as implemented in the **MfUSampler** R package (Mahani and Sharabiani 2017), while IDBR uses Metropolis sampling. Finally, it must also be noted that - to our knowledge - there is no open-source software available for IDBR.

The rest of this paper is organised as follows. In Section 2, we present the mathematical underpinnings of **DBR**. In Section 3, we illustrate the key features of **DBR** including model training, prediction, diagnostics and interpretation. Section 4 offers discussion and concluding remarks. System setup is captured in Appendix A.

2. The DBR Mathematical Framework

We begin this section with a brief review of beta regression. This is followed by changes made to beta regression in DBR for adapting it to rating responses, namely the forward and backward transformations, discretisation correction, and inflated extreme values. We end this section with a brief overview of the Bayesian estimation framework used in **DBR**.

2.1. Overview of Beta Regression

The probability density function (PDF) for beta distribution is given by:

$$f(y; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\,\Gamma(\beta)}\, y^{\alpha-1}\,(1-y)^{\beta-1}, \tag{1a}$$

$$\Gamma(z) \equiv \int_0^\infty u^{z-1} e^{-u} du. \tag{1b}$$

where the random variable y is restricted to the interval $[0, 1]$, $\alpha, \beta > 0$ are the so-called shape parameters of the distribution, and $\Gamma(.)$ is the Gamma function, which is a generalisation of the factorial function to real (and complex) numbers. For beta regression, we follow Ferrari and Cribari-Neto (2004); Zeileis *et al.* (2010) by using the alternative, mean-precision parameterisation of beta distribution:

$$f(y; \mu, \phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\,\phi)\,\Gamma\left((1-\mu)\,\phi\right)}\, y^{\mu\,\phi-1}\,(1-y)^{(1-\mu)\,\phi-1} \tag{2}$$

where the parameters μ (mean) and ϕ (precision) are linked to the shape parameters as follows:

$$\begin{cases} \mu & = \frac{\alpha}{\alpha+\beta} \qquad (3a) \\ \phi & = \alpha + \beta \qquad (3b) \end{cases}$$

and reversely:

$$\begin{cases} \alpha & = \mu\,\phi \qquad (4a) \\ \beta & = (1-\mu)\,\phi \qquad (4b) \end{cases}$$

We also require that $0 < \mu < 1$ and $\phi > 0$. The first and second moments of the distribution can be expressed in terms of the mean and precision parameters:

$$\mathrm{E}[y] = \mu \tag{5a}$$

$$\mathrm{VAR}[y] = \frac{\mu\,(1-\mu)}{1+\phi} \tag{5b}$$

We can see from the above that the model is heteroscedastic, i.e., the response variance is reduced (approaching zero) - for a fixed precision parameter - as the mean approaches either end of the $(0,1)$ range. This dispersion-compression at extreme ends of the response range is consistent with our expectation.

With the above mean-precision specification in hand, we can set up beta regression by assuming that the mean parameter - via a link function - is a linear function of model predictors, $\mathbf{x}$:

$$g(\mu) = \mathbf{x}^\top \beta, \tag{6}$$

where $g(.)$ could be a suitable function that maps $(0,1)$ to real line, e.g., the logit function, $g(u) = \log(u/(1-u))$. Further flexibility can be achieved by making the precision parameter a function of predictors, also via a suitable link function such as log. Note that the nonlinear link function causes the first derivative of the mean response with respect to any explanatory variable (or predictor), x_k, to be non-constant, i.e.:

$$\frac{\partial \mathrm{E}[y]}{\partial x_k} = \beta_k \, \frac{dg^{-1}(z)}{dz}|_{z=\mathbf{x}^\top \beta} \tag{7}$$

2.2. Forward and Backward Transformation of the Response Variable

In most likelihood-based approaches to parameter estimation, we must evaluate the 'logarithm' of the likelihood function, which in the case of DBR involves the PDF of the beta distribution (Eq. 2). Note that setting $y = 0$ or $y = 1$ causes the PDF to become zero, and hence its logarithm to become infinite. For this reason, we should only allow an open-ended interval for y, i.e., we require $y \in (0,1)$. Therefore, the first step towards adapting beta regression for DBR is to map the raw data to the $(0,1)$ range.

Consider K unique response values, sorted in increasing order: $y_1 < \cdots < y_K$. A naive transformation could be:

$$z_k = \frac{y_k - y_1}{y_K - y_1}. \tag{8}$$

But the above would map to $[0,1]$, rather than to $(0,1)$. Instead, we introduce left (b_l) and right (b_r) buffers:

$$b_l \equiv (y_2 - y_1)/2 \tag{9a}$$

$$b_r \equiv (y_K - y_{K-1})/2 \tag{9b}$$

We have essentially extended the 'latent' range of the data to $y_1 - b_l$ on the left, and $y_K + b_r$ on the right. This leads to the revised linear transformation:

$$y \longrightarrow z = u(y) \equiv \frac{y - \delta}{r}. \tag{10}$$

where $y \in \{y_1, \ldots, y_K\}$, and we have defined

$$\delta \equiv y_1 - b_l \tag{11a}$$

$$r \equiv y_K - y_1 + b_l + b_r \tag{11b}$$

It can be easily verified that the above transformation would map the data to the following range:

$$b_l/(y_K - y_1 + b_l + b_r) \leq u(y) \leq (y_K - y_1 + b_l)/(y_K - y_1 + b_l + b_r) \tag{12}$$

The above is what is needed for model training (i.e., regression). For prediction, we differentiate between two modes: 1) we can generate 'samples' from the posterior predictive density, or 2) we can provide a 'point' estimate that represents the expected average of the samples in (1).

In sample prediction, we must apply a reverse transformation to map the beta-distribution deviates to the observed range of outcome, followed by a discretisation step, in which we report y_k that is closest to the sample drawn from beta distribution according to the mean and dispersion parameters provided by the regression model. Referring to this transformation as $u_s^{-1}(.)$, we formally define it as

$$z \longrightarrow y = u_s^{-1}(z) \equiv y_k \; s.t. \; |r\,\hat{x} + \delta - y_k| \leq |r\,\hat{x} + \delta - y_{k'}|, \; \forall k' \in \{1, \ldots, K\}. \tag{13}$$

(Ties are theoretically possible given finite resolution of floating-point math on digital computers, but rare cases can be handled by choosing the smallest of the (at most two) k's.) The point prediction is calculated as the average of many samples, generated per above.

2.3. Discretisation Correction

The discretisation process must be reflected in the likelihood function for estimation. In other words, if we observe the value z_k, we cannot be certain that the latent sample drawn from the beta distribution - before discretisation - was z_k, but only that it was between $\frac{z_{k-1}+z_k}{2}$ and $\frac{z_k+z_{k+1}}{2}$, when $1 < k < K$. When $k = 1$, the left boundary is 0, and when $k = K$, the right boundary is 1. We summarise the above by introducing boundary functions $z_l(.)$ and $z_r(.)$:

$$z_l(y_k) = \begin{cases} 0 & k = 1 \\ \frac{u(y_{k-1})+u(y_k)}{2} & 1 < k \leq K \end{cases} \tag{14}$$

and

$$z_r(y_k) = \begin{cases} \frac{u(y_k)+u(y_{k+1})}{2} & 1 \leq k < K \\ 1 & k = K \end{cases} \tag{15}$$

Given the above, we assert that the contribution of a data point with response y_k to the likelihood is

$$P(y = y_k) = F\left(z_r(y_k)\right) - F\left(z_l(y_k)\right) \tag{16}$$

where $F(.)$ is the cumulative density function for beta distribution (Eq. 1a or 2), defined as:

$$F(x) = \int_0^x f(u)\, du. \tag{17}$$

2.4. Handling Extreme Responses

Extreme response to survey questions is one of several known types of bias in survey data Furnham (1986). For example, in some Likert scales, the observed proportion of 0's and 10's for a 0-10 scale may be higher than 1's and 9's, respectively. Researchers have discussed reasons for, impact of, and ways to handle, this bias (Meisenberg and Williams 2008; Greenleaf 1992).

Study-design considerations aside, one method for analysis of extreme responses is a mixture model, similar to zero-inflated Poisson distribution (Lambert 1992). In the case of beta distribution, we can modify Eq.16 as follows

$$P(y = y_k) = (1 - \pi_l - \pi_r)\ \{F\left(z_r(y_k)\right) - F\left(z_l(y_k)\right)\} + \begin{cases} \pi_l & k = 1 \\ 0 & 1 < k < K \\ \pi_r & k = K \end{cases} \tag{18}$$

The new parameters π_l, π_r are both probabilities, and thus must be between 0 and 1. In a regression context, they can be both made to be linear functions of predictors, via a suitable link function.

We take a different approach in DBR, however, and utilise the existing framework for handling discretisation by allowing the left and right buffers, b_l, b_r to be estimated from the data, rather than being fixed according to Eqs. 9a and 9b.

Besides boundary values, extreme responses can also be observed for midpoint/neutral points on a Likert scale. While the inflation/mixture-density approach of Eq. 18 can be deployed for this case as well, we refrain from including it in our implementation of DBR due to the increase in parameter count and hence risk of overfitting. Including a neutral point on the Likert scale may encourage the respondent to take an easy way out, thus providing more noise than information. Hence some practitioners have argued in favor of removing the neutral options, e.g., by using an even number of levels instead of an odd number (Allen and Seaman 2007).

2.5. Bayesian Estimation

In **DBR**, we opt for a Bayesian framework, using Markov Chain Monte Carlo sampling for posterior density estimation. This offers a few benefits over Maximum-Likelihood (ML) methods. First, the uncertainty of parameter estimates - expressed in terms of 'credible intervals' - does not reply on asymptotic arguments that may break down for small-data problems. Secondly, the MCMC sampling offers better chances to escape local maxima, compared with greedy optimisation algorithms used in Maximum-Likelihood (ML) methods. Finally, Bayesian frameworks allow users to express 'prior' beliefs in parameter values.

The conditional probability of observed responses is given by:

$$P(\mathbf{y}|\mathbf{X}; \phi, \mathbf{fi}, b_l, b_r) = \prod_{n=1}^{N} \left\{ F\left(z_r(y_{k[n]}; b_l, b_r); g^{-1}(\mathbf{fi}^\top \mathbf{x_n}), \phi\right) - F\left(z_l(y_{k[n]}; b_l, b_r); g^{-1}(\mathbf{fi}^\top \mathbf{x_n}), \phi\right)\right\} \tag{19}$$

In the above, $z_l(y; b_l, b_r)$ and $z_r(y; b_l, b_r)$ are functions that map each observed response to its left and right intervals over the $(0, 1)$ scale that is the domain of the beta distribution, $g^{-1}(\mathbf{fi}^\top \mathbf{x})$ is the ‘mean function’, i.e., function that calculates the mean of the beta distribution by forming the linear predictor $\mathbf{fi}^T \mathbf{x}$, followed by the logistic function. From the above, we obtain the following log-posterior:

$$
\begin{aligned}
L(\phi, \mathbf{fi}, b_l, b_r) = \log \Big(F \left(z_r(y_{k[n]}; b_l, b_r); g^{-1}(\mathbf{fi}^\top \mathbf{x_n}), \phi \right) \\
- F \left(z_l(y_{k[n]}; b_l, b_r); g^{-1}(\mathbf{fi}^\top \mathbf{x_n}), \phi \right) \Big) + \Phi(\phi) + \mathbf{B}(\mathbf{fi}) + B_l(b_l) + B_r(b_r)
\end{aligned}
\tag{20}
$$

where $\Phi(\phi)$, $\mathbf{B}(\mathbf{fi})$, $B_l(b_l)$ and $B_r(b_r)$ are the log-prior functions specified for the precision parameter of beta distribution (ϕ), coefficients for the mean parameters (β) and the left and right buffers (b_l, b_r), respectively. For results shown in this work, we use non-informative, flat priors for all parameters (with conservative boundaries). When using left and/or right buffers (see Section 2.2), the maximum allowed values for these buffers can be adjusted by the user (see Section 3).

For MCMC-based parameter estimation, we use our **MfUSampler** R package (Mahani and Sharabiani 2017). This software relies on a Gibbs wrapper around the univariate slice sampler (Neal 2003). As said earlier, MCMC has the inherent advantage of being able to escape local optima and finding the true global optimum, which is highly desirable for complex functions such as seen in Eq. 20. (However, this is not guaranteed to happen in every problem.) In addition, the fact that slice sampler is derivative-free provides further convenience for the user as it removes the need to supply the log-posterior derivatives (e.g., the gradient vector and the Hessian matrix).

3. Using the DBR Package

We begin this section with a brief introduction to the ’pain interference’ data used in the examples. This is followed by an illustration and discussion of **DBR** features for model training (`dbr`), MCMC diagnostics (`plot`), model prediction (`predict`), and model interpretation (`coef` and `summary`). Table 1 offers a list of public functions in **DBR** and a brief description of each one.

Function Name	**Description**
`dbr`	Model estimation (Bayesian, MCMC)
`plot`	MCMC diagnostics (trace plots, density plots, log-posterior)
`predict`	Model prediction (point, sample)
`coef`	Coefficients of internal beta regression
`summary`	Context-dependent pseudo-coefficients (plots and tables)
`coda_wrapper`	Access MCMC diagnostic functions in **coda**, e.g., autocorrelation, effective size, convergence tests
`dbr.control`	Setting configuration parameters of `dbr`

Table 1: Public functions in **DBR**. Note that `plot`, `predict`, `coef`, and `summary` are generic `S3` methods that have been implemented for the `dbr` class.

3.1. The Pain Interference Data

This data is based on a survey of nearly 10,000 patients in UK health clinics, conducted during 2010-2014, to assess the quality of care they received. After data filtering and consolidation, we have 1,318 observations of three variables: (pain) `severity`, (pain) `interference`, and `age`:

```
R> library("DBR")
R> data("pain")
R> summary(pain)
```

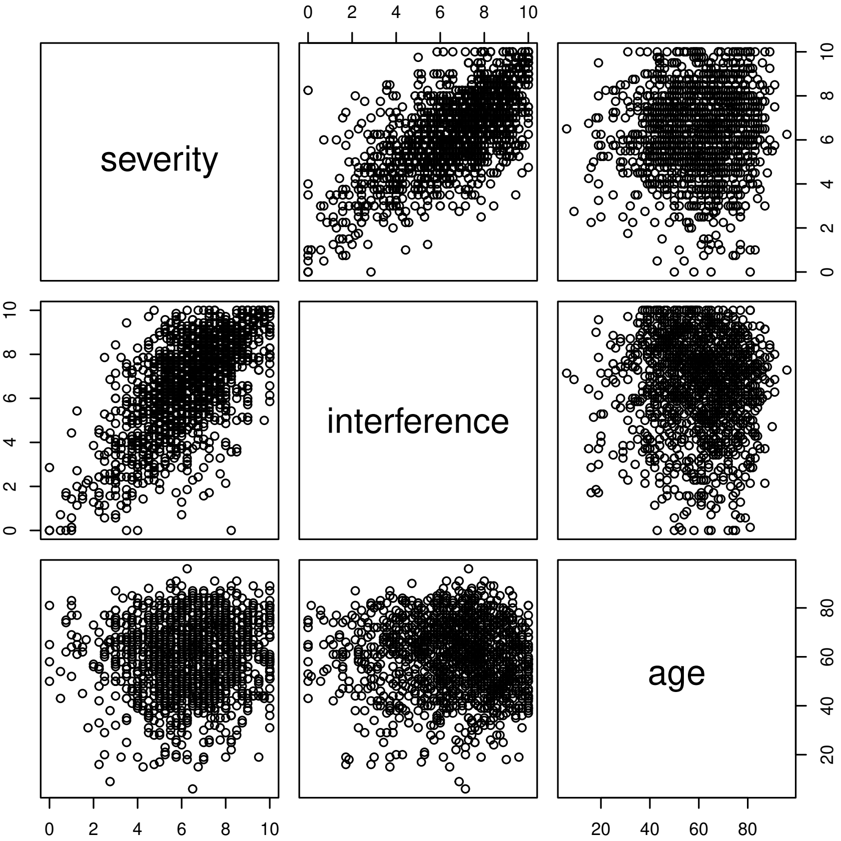
```
    severity        interference          age      
 Min.   : 0.000   Min.   : 0.000   Min.   : 6.0  
 1st Qu.: 5.000   1st Qu.: 5.286   1st Qu.:50.0  
 Median : 6.500   Median : 7.000   Median :61.0  
 Mean   : 6.318   Mean   : 6.584   Mean   :59.6  
 3rd Qu.: 7.750   3rd Qu.: 8.286   3rd Qu.:71.0  
 Max.   :10.000   Max.   :10.000   Max.   :96.0  
```

Description of variables:

- *Severity*: Average of 4 responses, each on a 0-10 scale (11 levels). They measure patients' perception of pain severity - over the 7 days leading up to the survey - at (1) its worst, (2) at its least, (3) on average, and (4) 'right now'.
- *Interference*: Average of 7 scores, each on a 0-10 scale (11 levels). These questions measure - over the 7 days leading up to the survey - the level of interference of pain in patient's life along the following dimensions: (1) general activity, (2) mood, (3) walking ability, (4) normal work (outside of home and housework), (5) relations with other people, (6) sleep and (7) enjoyment of life.
- *Age*: Age of respondents, in years.

The severity and interference scores are examples of ratings data. While numeric in appearance, they are bounded (between 0 and 10) and have discrete values. For example, the interference score can only assume values such as 0, 1/7, 2/7, ..., $10-2/7$, $10-1/7$, 10 (since it is formed as average of 7 numbers, each an integer between 0 and 10). As a brief exploration of the data, we review the pairwise scatterplots:

```
R> plot(pain)
```



We observe a clear positive correlation between pain severity and pain interference scores, confirmed via a Spearman correlation test:

```
R> cor.test(pain$severity, pain$interference, method = "spearman")

        Spearman's rank correlation rho

data:  pain$severity and pain$interference
S = 136306075, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.6427926
```

But the impact of age on pain interference is less clear. The Spearman test below indicates a statistically-significant negative correlation between age and pain interference:

```
R> cor.test(pain$age, pain$interference, method = "spearman")

        Spearman's rank correlation rho
```

```
data:  pain$age and pain$interference
S = 417536805, p-value = 0.0006158
alternative hypothesis: true rho is not equal to 0
sample estimates:
        rho
-0.09420837
```

In the examples that follow, we will predict pain interference (response variable) from pain severity and age of respondents (predictors).

3.2. Model Training

The main function in the **DBR** package is `dbr`, which is used for model training. It follows the standard conventions in R by using the first two slots for model formula, and a data frame containing the training data, with column names matching the terms specified in the formula. Taking advantage of default values for other function arguments, the simplest call to `dbr` would be:

```
R> est_dbr_default <- dbr(
+    formula = interference ~ severity + age
+    , data = pain
+  )
```

One function argument in `dbr` is `yunique`, which contains the expected unique values of the response variable. When left unspecified, this parameter is assumed to be equal to the observed unique values of the response variable in `data`, as specified in the model `formula`. While in many cases this is the desired behavior, in others it may not be. In our example, we expect the response variable to cover a range of 0 to 10, in increments of 1/7. However, as shown below, a few valid values are not present in the data:

```
R> setdiff(0:70, round(7 * sort(unique(pain$interference))))
```

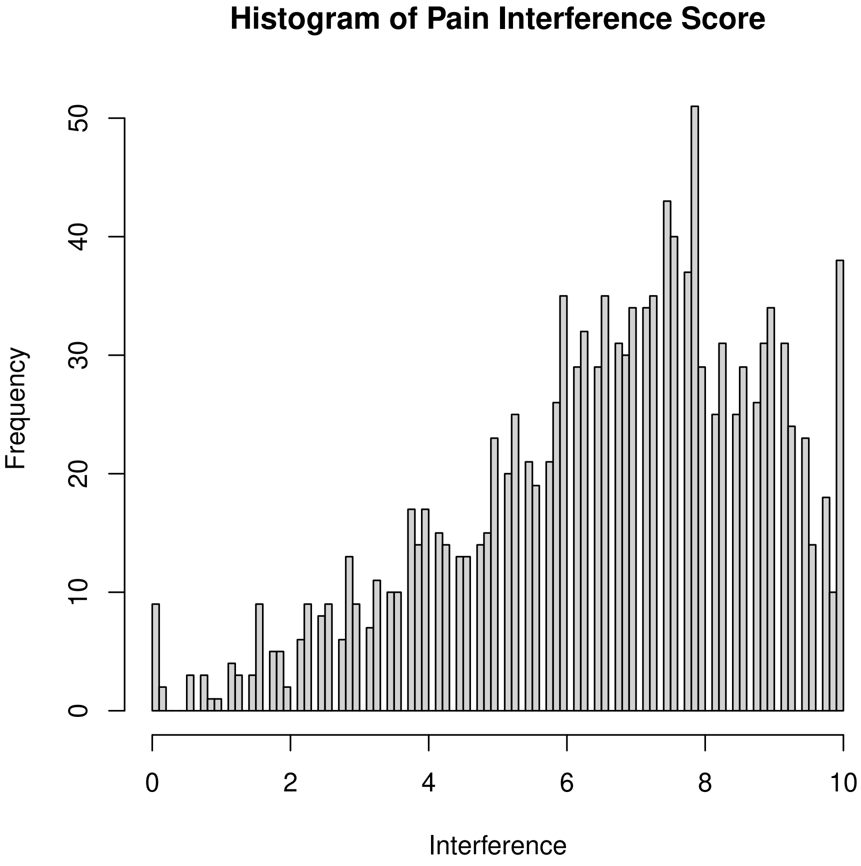
```
[1] 2 3
```

In other words, values of 2/7 and 3/7 have not occurred in the training data. This could distort the DBR algorithm's calculation of cutpoints. To correct this, we will explicitly override the `yunique` argument in the call to `dbr`.

The `control` argument of the `dbr` function includes other parameters for controlling the training process, which can be set by calling the function `dbr.control`. The first two arguments in this function, `nsmp` and `nburnin`, control the MCMC sampling process that is used for Bayesian parameter estimation (see Section 3.3), while the last three parameters - `estimate_left_buffer`, `estimate_right_buffer` and `buffer_max` - control the behavior of the model with respect to the left and right buffers (see Section 2.2).

Examining the histogram of the *interference* score indicates the presence of inflated extreme values at both left and right ends of the spectrum:

```
R> hist(
+    pain$interference, breaks = 100, xlab = "Interference"
+     , main = "Histogram of Pain Interference Score"
+   )
```



Hence, we will set both flags `estimate_left_buffer` and `estimate_right_buffer` to `TRUE`. Given the above, we make the following modified call to `dbr`:

```
R> my.seed <- 0
R> set.seed(my.seed)
R> est_dbr_short <- dbr(
+    formula = interference ~ severity + age
+    , data = pain
+    , yunique = 0:70 / 7
+    , control = dbr.control(
+      estimate_left_buffer = TRUE
+      , estimate_right_buffer = TRUE
+    )
+  )
```

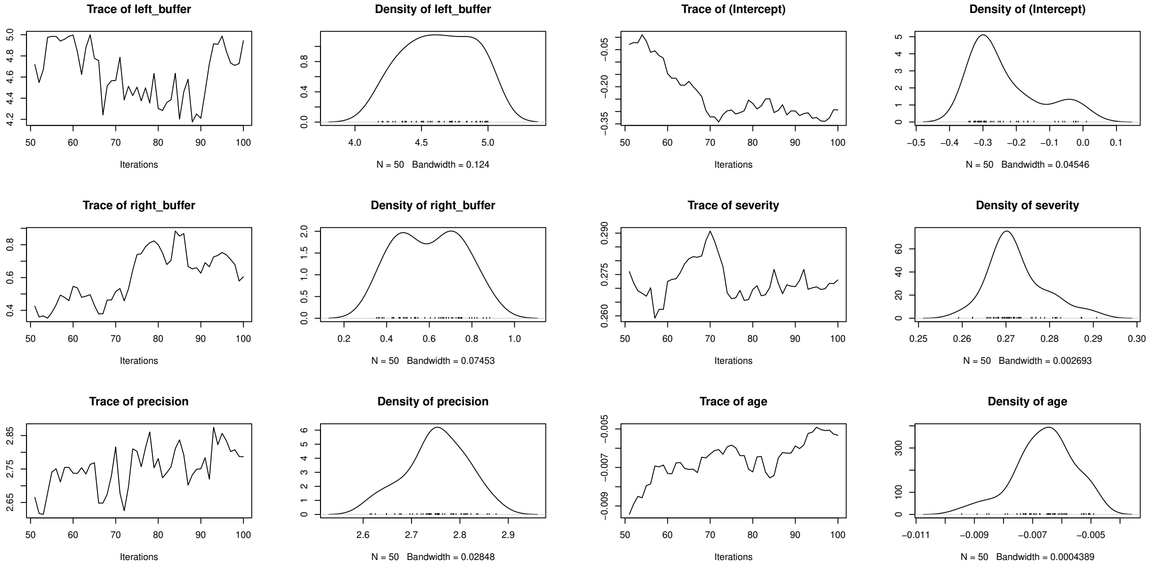
Note that, for reproducibility purposes, we set the random seed number to a predetermined value prior to all function calls involving random number generation.

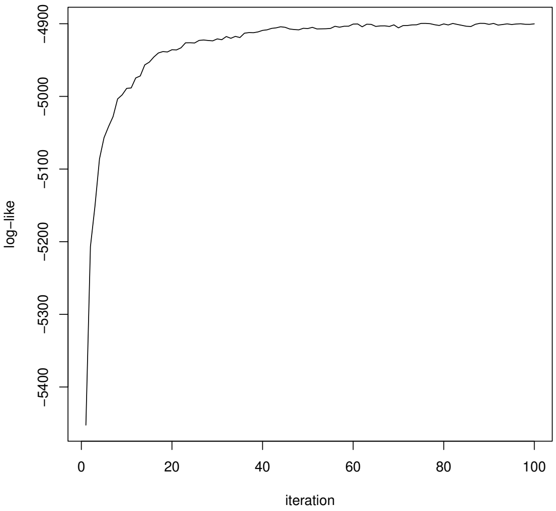
3.3. MCMC Diagnostics

As mentioned in Section 2.5, we use Markov Chain Monte Carlo (MCMC) sampling for parameter estimation in a Bayesian context which, among other benefits, leads to consistent quantification of uncertainty in parameter estimates. It is important to allow sufficient iterations for MCMC chains to converge - i.e., become independent of the initial conditions - before including the samples in parameter estimation. This is controlled by the `nburnin` parameter. We also need to include sufficient samples, after the initial burn-in period, which is controlled by the parameter `nsmp` (representing the total number of samples, including burn-in). The default values for `nsmp` and `nburnin` are 100 and 50 samples, respectively. (In most real-world settings, these numbers are too small.)

To decide if `nburnin` and `nsmp` are large enough or must be increased, we have various diagnostic tools at our disposal in the R package **coda** (Plummer, Best, Cowles, and Vines 2006). The `plot` function in **DBR** provides a good starting point for MCMC diagnostics by producing two outputs: 1) Trace and density plots for each estimated parameter (after discarding `nburnin` initial samples), using a call to the `plot.mcmc` function in the **coda** package, via the **plot** function in the **MfUSampler** package; and 2) plot of log-likelihood by iteration.

```
R> plot(est_dbr_short)
```





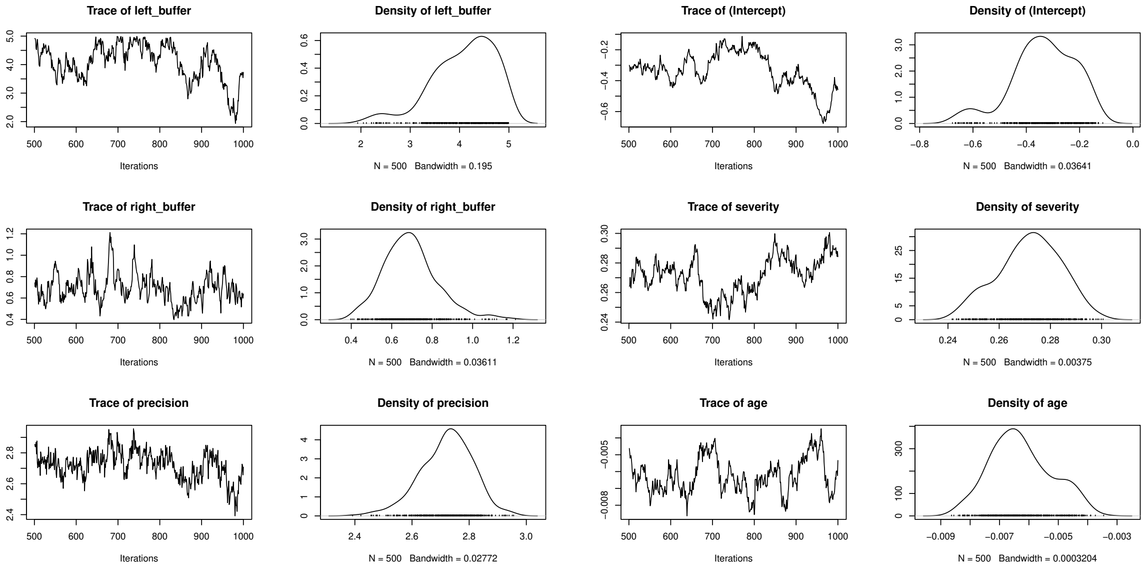
Trace plots show significant drift in parameter values after burn-in, a sign that the chain has not converged. This can also be seen in the log-likelihood plot, where there is a significant drift upwards in average values of the log-likelihood after the 50 initial samples. These two observations suggest that we need to discard more samples as burn-in period to allow for the MCMC chain to converge. Also, the autocorrelation evident in the trace plots suggests that the effective sample size for model parameters is much smaller than the apparent number (50 samples after burn-in). We should therefore collect more samples after burn-in as well.

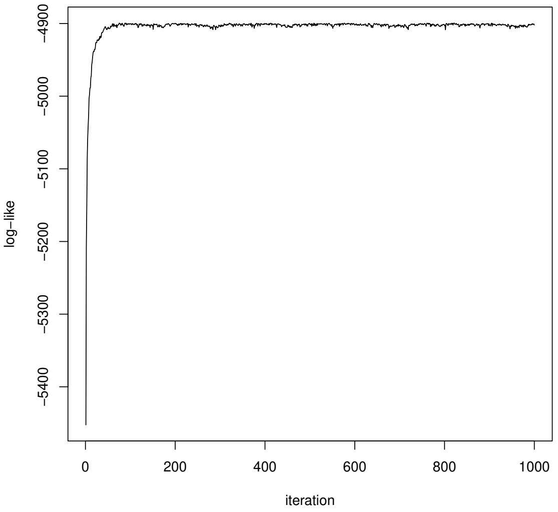
Given the above observations, we increase `nsmp` to 1000 and `nburnin` to 500 samples:

```
R> set.seed(my.seed)
R> est_dbr_long <- dbr(
+    formula = interference ~ severity + age
+    , data = pain
+    , yunique = 0:70 / 7
+    , control = dbr.control(
+      estimate_left_buffer = TRUE
+      , estimate_right_buffer = TRUE
+      , nsmp = 1000, nburnin = 500
+    )
+  )
```

Convergence is much better now:

```
R> plot(est_dbr_long)
```





More formal convergence tests - available in the **coda** package - can be conducted using the `coda_wrapper` function. The first argument should be the estimated DBR object, which is returned bythe `dbr` function. The second argument, `coda_function`, must be a **coda** diagnsotic function, whose first argument is an `mcmc` object. (This object is constructed from the `dbr` object inside the utility function `coda_wrapper`.) Further arguments to the `coda_function` can be passed using the `...` slot. For example, the Geweke convergence test (Geweke 1992) can be performed as follows:

```
R> coda_wrapper(est_dbr_long, coda::geweke.diag, frac1 = 0.15)

Fraction in 1st window = 0.15
Fraction in 2nd window = 0.5

 left_buffer right_buffer    precision  (Intercept)     severity
```

```
   -0.1198     -4.6094     -1.5803      1.5498     -0.9069
       age
   -1.3863
```

For a full list of MCMC diagnostics available in **coda**, see the package documentation at https://cran.r-project.org/web/packages/coda/coda.pdf.

3.4. Model Prediction

Having arrived at a convergence MCMC chain resulting in stable coefficients, we now focus on prediction capabilities of the **DBR** package. The package offers two prediction modes, `point` prediction and `sample` prediction. The `point` prediction returns the expected - or mean - response, which is a continuous value. This is achieved via a call to the `predict` function and by setting the `type` argument to `point`:

```
R> set.seed(my.seed)
R> pred_point <- predict(est_dbr_long, newdata = pain, type = "point")
R> head(pred_point)

[1] 5.270857 4.758857 7.858857 5.012571 6.793429 3.746857

R> hist(pred_point, breaks = 100, col = "grey"
+       , xlab = "Pain Inteference"
+       , main = "Histogram of Point Predictions"
+       )
```

As we can see in the above histogram, the extreme-value inflation is not necessarily reflected in the predicted mean responses. This is because the point prediction averages out the sample-level dispersion from predictions. To see the full dispersion of predictions, we have to switch to `sample` mode:

```
R> set.seed(my.seed)
R> pred_sample <- predict(est_dbr_long, newdata = pain)
R> hist(pred_sample, breaks = 100, col = "grey"
+       , xlab = "Pain Inteference"
+       , main = "Histogram of Sample Predictions"
+       )
```

The histogram of predictions in the `sample` mode resembles the observed data histogram much more closely. This is can be formally tested using the Kolmogorov–Smirnov test:

```
R> ks.test(pred_sample, pain$interference)

        Two-sample Kolmogorov-Smirnov test

data:  pred_sample and pain$interference
D = 0.034731, p-value = 0.08372
alternative hypothesis: two-sided
```

We can see that the null hypothesis i.e., the predictive posterior and the training data have the same probability distribution, cannot be rejected with 95% confidence.

3.5. Model Interpretation

DBR offers two ways to interpret, or explain, the fitted model, using the `coef` and `summary` functions. `coef` returns a table of quantiles for the model coefficients, using the MCMC samples collected during training. The values of the quantiles can be set using the `prob` argument, with default value being `c(0.025, 0.5, 0.975)`, which returns the median as well as the symmetric 95% credible interval for each coefficient, as well as the precision and the left and right buffer parameters.

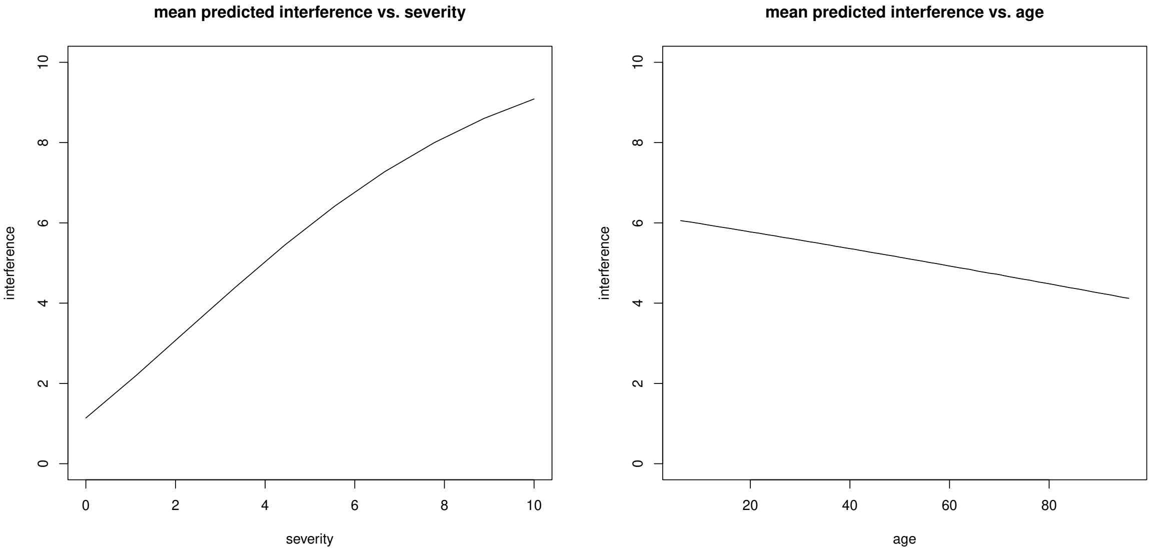
```
R> coef(est_dbr_long, prob = c(0.05, 0.5, 0.95))
```

```
    left_buffer right_buffer precision (Intercept)  severity
5%     2.816840    0.4893287  2.555945  -0.5905511 0.2499705
50%    4.190474    0.6857060  2.726520  -0.3282158 0.2727203
95%    4.908693    0.9364807  2.849629  -0.1645590 0.2903637
             age
5%  -0.007901430
50% -0.006401699
95% -0.004398898
```

Examining the credible intervals indicates if the coefficients are significant, with a Bayesian interpretation. In the above example, we observe that the coefficients for both severity and age are significant at the 95% level.

While this information is helpful, the coefficients in DBR do not have an intuitive interpretation, due to the nonlinearities imposed by the beta distribution and the discretisation process. In other words, each unit change in a given predictor or covariate does not produce a constant change in the mean response in a DBR model. This is why we provide the `summary` function for producing model 'pseudo-coefficients', i.e., plots of the relationship between a predictor and the mean response, as a function of the value of the predictor, and conditioned on a fixed set of values for the remaining predictors in the model (if any):

```
R> summary_dbr_long <- summary(est_dbr_long)
```

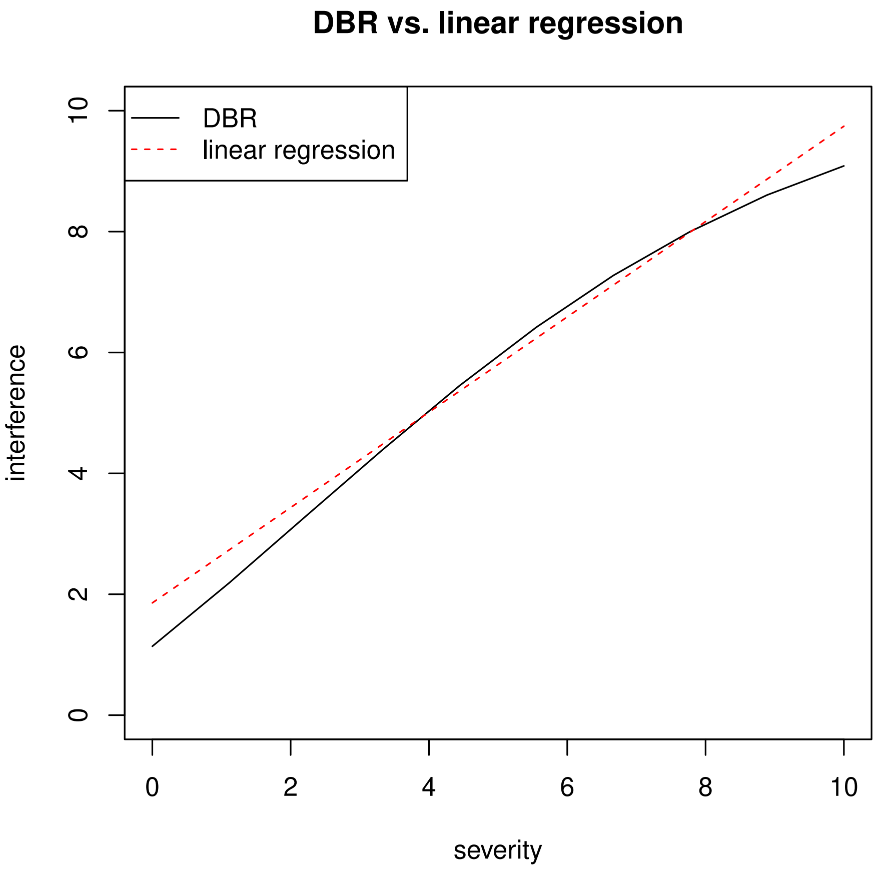


The function argument `context` can be used to provide the baseline values of predictors. If missing, the function defaults to using the first row of the training data. For example, in the above plots, `age` is fixed at 44 years for the `severity` plot (left), and `severity` is fixed at 4.25 for `age` plot (right). (The `make_plot` argument allows the user to switch on/off the plots.)

The above plots reveal that, the dependence of mean predicted interference on `age` is very close to linear, while `severity` has a noticeably nonlinear relationship with mean predicted interference. This makes sense given that the impact of `age` - over the range of its observed values in the training data - on mean response is restricted to a narrower range than `severity`.

The function also returns the dataframes and mean-predicted-response values - used in creating plots - for potential further analysis. For example, it is interesting to compare the predictions of the DBR model against linear regression:

```
R> plot(summary_dbr_long$severity$X$severity, summary_dbr_long$severity$y
+      , xlab = "severity", ylab = "interference"
+      , main = "DBR vs. linear regression", type = "l"
+      , ylim = c(0, 10))
R> lines(
+    summary_dbr_long$severity$X$severity
+    , predict(
+      lm(interference ~ age + severity, pain)
+      , newdata = summary_dbr_long$severity$X
+    )
+    , col = "red", lty = 2
+  )
R> legend("topleft", legend = c("DBR", "linear regression")
+         , pch = rep(-1, 2), col = c("black", "red")
+         , lty = c(1, 2))
```



As expected, the dependence of mean predicted interference score on severity score for linear regression is a straight line, while the DBR model predicts a nonlinear relationship. In particular, we see a declining slope as the severity score approaches its maximum value of 10. We also observe a lower predicted interference for zero severity, compared to the prediction from the linear regression model, which is closer to the ideal outcome of zero predicted interference for zero severity (no pain).

4. Discussion

We have presented the **DBR** R package - available on the Comprehensive R Archive Network (CRAN) at `https://cran.r-project.org/web/packages/DBR/index.html` - for analysis of ratings data using the discretised beta regression (DBR) analytical framework. DBR allows for quantifying the impact of predictors on the response while relaxing the unrealistic assumptions of fixed slope and variance, which are present in homoscedastic linear regression. Major software components and use cases were presented using an example from pain-survey data. For more details on the functions available in the **DBR** package, including their arguments and output, please see the package documentation.

The Bayesian framework, along with Markov Chain Monte Carlo (MCMC) sampling for estimation, offers several advantages (Kruschke and Liddell 2018; Liddell and Kruschke 2018), including robust credible-interval calculation without resorting to unrealistic assumptions about the asymptotic behavior of the log-likelihood function. While MCMC can be time-consuming for large datasets, and/or when many burn-in and sampling iterations are needed for accurate estimation of model parameters, there are several techniques proposed in the literature for speeding it up, such as in Mahani and Sharabiani (2015).

One future improvement is to take full advantage of the Bayesian framework by allowing for users of the DBR R package to supply or select informative priors for regression parameters. Another direction of future work is to add support in the software for inflated midpoint values using the mixture framework described in Section 2.4. Embedding DBR in composite settings such as multi-level and mixture models is another potential extension, which the Bayesian framework adopted for DBR would naturally support.

References

Allen IE, Seaman CA (2007). "Likert scales and data analyses." *Quality progress*, **40**(7), 64–65.

Armstrong GD (1981). "Parametric statistics and ordinal data: A pervasive misconception." *Nursing Research*, **30**(1), 60–62.

Carifio J, Perla R (2008). "Resolving the 50-year debate around using and misusing Likert scales." *Medical education*, **42**(12), 1150–1152.

Carifio J, Perla RJ (2007). "Ten common misunderstandings, misconceptions, persistent myths and urban legends about Likert scales and Likert response formats and their antidotes." *Journal of social sciences*, **3**(3), 106–116.

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

Furnham A (1986). "Response bias, social desirability and dissimulation." *Personality and individual differences*, **7**(3), 385–400.

Geweke J (1992). "Evaluating the accuracy of sampling-based approaches to the calculations of posterior moments." *Bayesian statistics*, **4**, 641–649.

Goodrich B, Gabry J, Ali I, Brilleman S (2018). "rstanarm: Bayesian applied regression modeling via Stan." *R package version*, **2**(4), 1758.

Greenleaf EA (1992). "Measuring extreme response style." *Public Opinion Quarterly*, **56**(3), 328–351.

Harpe SE (2015). "How to analyze Likert and other rating scale data." *Currents in Pharmacy Teaching and Learning*, **7**(6), 836–850.

Hasan A, Wang Z, Mahani AS (2016). "Fast Estimation of Multinomial Logit Models: R Package mnlogit." *Journal of Statistical Software*, **75**(3), 1–24. doi:10.18637/jss.v075.i03.

Jamieson S (2004). "Likert scales: How to (ab) use them?" *Medical education*, **38**(12), 1217–1218.

Knapp TR (1990). "Treating ordinal scales as interval scales: an attempt to resolve the controversy." *Nursing research*, **39**(2), 121–123.

Kruschke JK, Liddell TM (2018). "The Bayesian New Statistics: Hypothesis testing, estimation, meta-analysis, and power analysis from a Bayesian perspective." *Psychonomic Bulletin & Review*, **25**(1), 178–206.

Kuzon W, Urbanchek M, McCabe S (1996). "The seven deadly sins of statistical analysis." *Annals of plastic surgery*, **37**, 265–272.

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

Lesnoff, M, Lancelot, R (2012). *aod: Analysis of Overdispersed Data.* R package version 1.3.2, URL `https://cran.r-project.org/package=aod`.

Liddell TM, Kruschke JK (2018). "Analyzing ordinal data with metric models: What could possibly go wrong?" *Journal of Experimental Social Psychology*, **79**, 328–348.

Mahani AS, Sharabiani MT (2015). "SIMD parallel MCMC sampling with applications for big-data Bayesian analytics." *Computational Statistics & Data Analysis*, **88**, 75–99.

Mahani AS, Sharabiani MTA (2017). "Multivariate-From-Univariate MCMC Sampler: The R Package MfUSampler." *Journal of Statistical Software, Code Snippets*, **78**(1), 1–22. `doi:10.18637/jss.v078.c01`.

Meisenberg G, Williams A (2008). "Are acquiescent and extreme response styles related to low intelligence and education?" *Personality and individual differences*, **44**(7), 1539–1550.

Najera-Zuloaga J, Lee DJ, Arostegui I (2018). "Comparison of beta-binomial regression model approaches to analyze health-related quality of life data." *Statistical methods in medical research*, **27**(10), 2989–3009.

Najera-Zuloaga J, Lee DJ, Arostegui I (2020). "PROreg: Patient Reported Outcomes Regression Analysis." R package version 1.1, URL `https://CRAN.R-project.org/package=PROreg`.

Neal RM (2003). "Slice sampling." *Annals of statistics*, pp. 705–741.

Norman G (2010). "Likert scales, levels of measurement and the "laws" of statistics." *Advances in health sciences education*, **15**(5), 625–632.

Pell G (2005). "Use and misuse of Likert scales."

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL `https://journal.r-project.org/archive/`.

Taverne C, Lambert P (2014). "Inflated discrete beta regression models for Likert and discrete rating scale outcomes." *arXiv preprint arXiv:1405.4637*.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Fourth edition. Springer, New York. ISBN 0-387-95457-0, URL https://www.stats.ox.ac.uk/pub/MASS4/.

Zeileis A, Cribari-Neto F, Grün B, Kos-midis I (2010). "Beta regression in R." *Journal of statistical software*, **34**(2), 1–24.

A. Setup

Below is the corresponding R session information.

```
R> sessionInfo()

R version 4.1.1 (2021-08-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.3 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0

locale:
 [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=C.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
 [7] LC_PAPER=C.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C           LC_TELEPHONE=C
[11] LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
[1] DBR_1.4.0

loaded via a namespace (and not attached):
[1] compiler_4.1.1   ars_0.6          tools_4.1.1
[4] HI_0.4           coda_0.19-4      grid_4.1.1
[7] MfUSampler_1.0.6 lattice_0.20-44
```

Affiliation:

Mansour T.A. Sharabiani
School of Public Health
Imperial College London

UK
E-mail: mt5605@ic.ac.uk