Extended Beta Regression in **R**: Shaken, Stirred, Mixed, and Partitioned

Bettina Grün
Johannes Kepler
Universität Linz

Ioannis Kosmidis
University College London

Achim Zeileis
Universität Innsbruck

Abstract

Beta regression – an increasingly popular approach for modeling rates and proportions – is extended in various directions: (a) bias correction/reduction of the maximum likelihood estimator, (b) beta regression tree models by means of recursive partitioning, (c) latent class beta regression by means of finite mixture models. All three extensions may be of importance for enhancing the beta regression toolbox in practice to provide more reliable inference and capture both observed and unobserved/latent heterogeneity in the data. Using the analogy of Smithson and Verkuilen (2006), these extensions make beta regression not only "a better lemon squeezer" (compared to classical least squares regression) but a full-fledged modern juicer offering lemon-based drinks: shaken and stirred (bias correction and reduction), mixed (finite mixture model), or partitioned (tree model). All three extensions are provided in the R package **betareg** (at least 2.4-0), building on generic algorithms and implementations for bias correction/reduction, model-based recursive partioning, and finite mixture models, respectively. Specifically, the new functions `betatree()` and `betamix()` reuse the object-oriented flexible implementation from the R packages **party** and **flexmix**, respectively.

Keywords: beta regression, bias correction, bias reduction, recursive partitioning, finite mixture, R.

1. Introduction

A brief review of beta regression

Beta regression is a model for continuous response variables y which assume values in the open unit interval $(0, 1)$. Such response variables may stem from rates, proportions, concentrations, etc. A regression model where the mean as well as the precision is modelled through covariates was introduced by Ferrari and Cribari-Neto (2004) along with the extensions by Smithson and Verkuilen (2006) and Simas, Barreto-Souza, and Rocha (2010). This model is also referred to as "double index regression model" because it contains two regression parts: one for the mean and one for the precision. Ferrari and Cribari-Neto (2004) employed an alternative parametrization of the beta distribution characterizing more easily the mean and the variance. In this parameterization the beta distribution has the density

$$f(y; \mu, \phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, \quad 0 < y < 1, \quad 0 < \mu < 1, \quad \phi > 0, \quad (1)$$

where $\Gamma(\cdot)$ is the gamma function. A beta-distributed variable Y then has mean $\mathsf{E}(Y) = \mu$ and variance $\mathsf{VAR}(Y) = \mu(1 - \mu)/(1 + \phi)$ so that ϕ can be seen as a precision parameter.

The double index beta regression model is specified in the following way. Given observations on n independent beta-distributed random variables Y_i ($i = 1, \ldots, n$), the corresponding parameters μ_i and ϕ_i are linked to linear predictors η_i and ζ_i as follows

$$g_1(\mu_i) = \eta_i = x_i^\top \beta\,, \tag{2}$$

$$g_2(\phi_i) = \zeta_i = z_i^\top \gamma\,, \tag{3}$$

where x_i and z_i are p- and q-dimensional vectors of covariates observed along with Y_i ($i = 1, \ldots, n$), and $\beta = (\beta_1, \ldots, \beta_p)^\top$, $\gamma = (\gamma_1, \ldots, \gamma_q)^\top$ are the vectors of coefficients associated with the means and the precisions, respectively. The functions $g_1(\cdot)$ and $g_2(\cdot)$ are monotonic link functions, preferably with the property of mapping the range of μ_i $(0, 1)$ and ϕ_i $(0, \infty)$, respectively, to the real line. Suitable candidates for $g_1(\cdot)$ are the logit, probit and generally any inverse of a cumulative distribution function, and for $g_2(\cdot)$ the log function. Another common choice for $g_2(\cdot)$ is the identity function which, however, can lead to invalid results when some $\zeta_i < 0$.

Typically, the coefficients β and γ are estimated by maximum likelihood (ML) and inference is based on the usual central limit theorem with its associated asymptotic tests, e.g., likelihood ratio, Wald, score/Lagrange multiplier (LM).

Implementation in R

The R package **betareg** (Cribari-Neto and Zeileis 2010) provides ML estimation of beta regressions in its main model fitting function `betareg()`. The interface as well as the fitted model objects are designed to be similar to those from `glm()`. The model specification is via a `formula` plus `data`. Because two types of covariates need to be distinguished a two-part formula is allowed based on functionality provided by the **Formula** package (Zeileis and Croissant 2010). For example, `y ~ x1 + x2 + x3 | z1 + z2` would assign the covariates `x1`, `x2`, and `x3` to the mean submodel (2) and `z1` and `z2` to the precision submodel (3), respectively. Function `betareg()` internally uses function `optim()` as a general purpose optimizer to maximize the log-likelihood. The fitted model has methods for several extractor functions, e.g., `coef()`, `vcov()`, `residuals()`, `logLik()`. Base methods for the returned fitted model are `summary()`, `AIC()`, `confint()`. Further methods are available for functions from **lmtest** (Zeileis and Hothorn 2002) and **car** (Fox and Weisberg 2011), e.g., `lrtest()`, `waldtest()`, `coeftest()`, and `linearHypothesis()`. Multiple testing is possible via package **multcomp** (Hothorn, Bretz, and Westfall 2008) and structural change tests can be performed using package **strucchange** (Zeileis, Leisch, Hornik, and Kleiber 2002).

Extensions

Although the **betareg** package as published by Cribari-Neto and Zeileis (2010) provides a rather complete beta regression toolbox based on classical ML inference, further techniques may be required in practice. First, it has been shown that ML inference may be severely biased in the context of beta regression (Kosmidis and Firth 2010), possibly leading to overly optimistic inferences in the sense of underestimating the standard errors of the estimators. Second, it is not always easy to capture all heterogeneity in the data through the two linear predictors, especially when there are latent unobserved groups/clusters of observations.

To address the first issue of potentially biased inference, the results of Kosmidis and Firth (2010) are extended to the case with mean and precision covariates and the corresponding methods are implemented in the `betareg()` function starting from version 2.4-0 of **betareg**. The software optionally allows for bias-corrected or bias-reduced estimation by adopting the unifying iteration developed in Kosmidis and Firth (2010).

To address the second issue of heterogeneity between groups/clusters of observations, two generic strategies, model-based recursive partitioning (Zeileis, Hothorn, and Hornik 2008) and finite mixture models (see e.g., McLachlan and Peel 2000; Frühwirth-Schnatter 2006), are applied to beta regressions. The idea for both techniques is to capture situations in which the regression relationships vary across groups in the population. If one can identify variables which are related to such groups, one may be able to include them directly in the regression relationships. However, (a) this may lead to rather complex and hard to interpret models, and (b) unnecessary complexity is introduced if the differences are only present in a subset of the combined groups induced by several variables. Model-based recursive partitioning avoids such drawbacks. Furthermore, if groups cannot be directly related to observed variables, the heterogeneity can be accounted for by using finite mixture models. Therefore, extensions of the **betareg** package are introduced where model heterogeneity is taken into account when covariates that characterize the groups are available, and when the heterogeneity is due to latent variables. The new function `betatree()` provides model-based recursive partitioning of beta regressions leveraging tools from the **party** package (Hothorn, Hornik, Strobl, and Zeileis 2011), and the function `betamix()` provides beta regression mixture models (or latent class beta regression) reusing the generic functionality from the **flexmix** package (Leisch and Grün 2012).

2. Bias correction and reduction in beta regressions

2.1. Preamble

Kosmidis and Firth (2010) show that bias correction (BC) or bias reduction (BR) of the ML estimator in parametric models may be achieved via a unifying *quasi* Fisher scoring algorithm. They illustrate the applicability of their algorithm in a beta regression setting with a common precision parameter ϕ for all subjects, also revealing some errors in previous literature for the reduction of bias in beta regression models – specifically mistakes in Ospina, Cribari-Neto, and Vasconcellos (2006) and Simas *et al.* (2010) — that led to misleading negative conclusions about the effect of BC/BR on inferences for beta regression models. In Kosmidis and Firth (2010), it is shown that BC/BR for beta regression models can be desirable because the ML estimator of ϕ may demonstrate substantial upwards bias, which in turn may lead to underestimation of asymptotic standard errors and hence over-optimistic Wald-type inferences (e.g., confidence intervals with coverage far below the nominal levels).

The results in Kosmidis and Firth (2010) are extended here to cover not only the case of constant ϕ but also a regression part for the precision parameters as shown in Equation 3.

2.2. Generic framework

Denote by 0_k a vector of k zeros and by $S(\theta)$ the vector of the log-likelihood derivatives for a

parametric model with parameter θ. Firth (1993) showed that the solution $\tilde{\theta}$ of the equation

$$S(\tilde{\theta}) + A(\tilde{\theta}) = 0_{p+q}, \tag{4}$$

has smaller asymptotic bias than the ML estimator, if the t-th component of the vector $A(\theta)$ has the form

$$A_t(\theta) = \frac{1}{2} \operatorname{tr} \left[\{F(\theta)\}^{-1} \{P_t(\theta) + Q_t(\theta)\} \right] \quad (t = 1, \ldots, p+q),$$

with $F(\theta)$ the expected information matrix and

$$P_t(\theta) = \mathsf{E}\{S(\theta)S^\top(\theta)S_t(\theta)\} \quad (t = 1, \ldots, p+q), \tag{5}$$
$$Q_t(\theta) = -\mathsf{E}\{I(\theta)S_t(\theta)\} \quad (t = 1, \ldots, p+q), \tag{6}$$

where $S_t(\theta)$ denotes the t-th component of $S(\theta)$ $(t = 1, \ldots, p+q)$ and $I(\theta)$ is the observed information matrix (minus the matrix of second derivatives of the log-likelihood with respect to θ).

The quasi Fisher scoring iteration that has been developed in Kosmidis and Firth (2010) attempts to solve Equation 4. Specifically, at the j-th step of the iterative procedure, the current value $\theta^{(j)}$ of the parameter vector is updated to $\theta^{(j+1)}$ by

$$\theta^{(j+1)} = \theta^{(j)} + \left\{ F\left(\theta^{(j)}\right) \right\}^{-1} S\left(\theta^{(j)}\right) - b\left(\theta^{(j)}\right), \tag{7}$$

where $b(\theta) = -\{F(\theta)\}^{-1}A(\theta)$ is the vector of the first term in the expansion of the bias of the ML estimator.

If the summand $b\left(\theta^{(j)}\right)$ is ignored, then iteration (7) becomes the usual Fisher scoring iteration that can be used to solve the ML score equations $S(\hat{\theta}) = 0_{p+q}$.

Furthermore, if the starting value $\theta^{(0)}$ is the ML estimator $\hat{\theta}$, then $\theta^{(1)}$ is the bias-corrected estimator $\theta^\dagger$ of θ defined as

$$\theta^\dagger = \hat{\theta} - b(\hat{\theta}),$$

which also has smaller asymptotic bias compared to the ML estimator (Efron 1975).

Hence, the quasi Fisher scoring iteration provides a unified framework for implementing all three types of estimators – ML, BR, and BC – by merely deciding whether the summand $b\left(\theta^{(j)}\right)$ is absent or present in the right hand side of (7), and whether more than one iteration should be allowed in the latter case.

2.3. Bias correction and bias reduction for beta regressions

Denote the vector of the $p+q$ model parameters in a beta regression model by $\theta = (\beta^\top, \gamma^\top)^\top$, and let X and Z be the $n \times p$ and $n \times q$ model matrices with i-th row x_i and z_i, respectively $(i = 1, \ldots, n)$. The ingredients required for setting the iteration described in Section 2.2 are closed-form expressions for the vector of log-likelihood derivatives $S(\theta)$, the expected information matrix $F(\theta)$ and the two higher-order joint null cumulants of log-likelihood derivatives $P_t(\theta)$ and $Q_t(\theta)$ shown in (5) and (6). Based on these, all matrix multiplications and inversions can be performed numerically during the iterative procedure.

The fact that all the aforementioned quantities depend on X and Z, and that $S(\theta)$ and $I(\theta)$ depend additionally on the random variables Y_i $(i = 1, \ldots, n)$ has been concealed here merely

for notational simplicity. The same convention is used for the derivations below, additionally concealing the dependence on θ unless otherwise stated.

The log-likelihood for the beta regression model in (1) is $\ell(\theta) = \sum_{i=1}^{n} \ell_i(\theta)$ with

$$\ell_i(\theta) = \phi_i\mu_i(T_i - U_i) + \phi_i U_i + \log\Gamma(\phi_i) - \log\Gamma(\phi_i\mu_i) - \log\Gamma(\phi_i(1 - \mu_i)) \tag{8}$$

where μ_i and ϕ_i are defined by inverting (2) and (3), respectively, and where $T_i = \log Y_i$ and $U_i = \log(1 - Y_i)$ are the sufficient statistics for the beta distribution with natural parameters $\phi_i\mu_i$ and $\phi_i(1 - \mu_i)$ $(i = 1, \ldots, n)$, respectively.

Direct differentiation of the log-likelihood function reveals that the vector of log-likelihood derivatives has the form

$$S(\theta) = \nabla_\theta \ell(\theta) = \begin{bmatrix} X^\top \Phi D_1 \left(\bar{T} - \bar{U}\right) \\ Z^\top D_2 \left\{ M \left(\bar{T} - \bar{U}\right) + \bar{U} \right\} \end{bmatrix}, \tag{9}$$

with $\Phi = \text{diag}\{\phi_1, \ldots, \phi_n\}$, $M = \text{diag}\{\mu_1, \ldots, \mu_n\}$, $D_1 = \text{diag}\{d_{1,1}, \ldots, d_{1,n}\}$, and $D_2 = \text{diag}\{d_{2,1}, \ldots, d_{2,n}\}$, where $d_{1,i} = \partial\mu_i/\partial\eta_i$ and $d_{2,i} = \partial\phi_i/\partial\zeta_i$. Furthermore, $\bar{T} = (\bar{T}_1, \ldots, \bar{T}_n)^\top$ and $\bar{U} = (\bar{U}_1, \ldots, \bar{U}_n)^\top$ are the vectors of centered sufficient statistics, with

$$\bar{T}_i = T_i - \mathsf{E}(T_i),$$
$$\bar{U}_i = U_i - \mathsf{E}(U_i),$$

where $\mathsf{E}(T_i) = \psi^{(0)}(\phi\mu_i) - \psi^{(0)}(\phi_i)$ and $\mathsf{E}(U_i) = \psi^{(0)}(\phi(1 - \mu_i)) + \psi^{(0)}(\phi_i)$, with $\psi^{(r)}(k) = \partial^{r+1}\log\Gamma(k)/\partial k^{r+1}$ the polygamma function of degree r $(r = 0, 1, \ldots; i = 1, \ldots, n)$.

Differentiating $\ell(\theta)$ one more time reveals that the observed information on θ is

$$I(\theta) = F(\theta) - \begin{bmatrix} X^\top \Phi D_1' \text{diag}\{\bar{T} - \bar{U}\}X & X^\top D_1 \text{diag}\{\bar{T} - \bar{U}\}D_2 Z \\ Z^\top D_2 \text{diag}\{\bar{T} - \bar{U}\}D_1 X & Z^\top D_2' \left(M \text{diag}\{\bar{T} - \bar{U}\} + \text{diag}\{\bar{U}\}\right)Z \end{bmatrix}, \tag{10}$$

where

$$F(\theta) = \begin{bmatrix} X^\top D_1 \Phi K_2 \Phi D_1 X & X^\top D_1 \Phi \left(MK_2 - \Psi_1\right)D_2 Z \\ Z^\top D_2 \left(MK_2 - \Psi_1\right)\Phi D_1 X & Z^\top D_2 \left\{M^2 K_2 + (1_n - 2M)\Psi_1 - \Omega_1\right\}D_2 Z \end{bmatrix}, \tag{11}$$

is the expected information on θ, because the second summand in the right hand side of (10) depends linearly on the centered sufficient statistics and hence has expectation zero. Here, 1_n is the $n \times n$ identity matrix, $D_1' = \text{diag}\{d_{1,1}', \ldots, d_{1,n}'\}$ with $d_{1,i}' = \partial^2\mu_i/\partial\eta_i^2$ and $D_2' = \text{diag}\{d_{2,1}', \ldots, d_{2,n}'\}$ with $d_{2,i}' = \partial^2\phi_i/\partial\zeta_i^2$ $(i = 1, \ldots, n)$. Furthermore, $K_2 = \text{diag}\{\kappa_{2,1}, \ldots, \kappa_{2,n}\}$, where $\kappa_{2,i} = \text{VAR}\left(\bar{T}_i - \bar{U}_i\right) = \psi^{(1)}(\phi_i\mu_i) + \psi^{(1)}(\phi_i(1 - \mu_i))$ for $i = 1, \ldots, n$ and

$$\Psi_r = \text{diag}\left\{\psi^{(r)}(\phi_1(1 - \mu_1)), \ldots, \psi^{(r)}(\phi_n(1 - \mu_n))\right\},$$
$$\Omega_r = \text{diag}\left\{\psi^{(r)}(\phi_1), \ldots, \psi^{(r)}(\phi_n)\right\} \quad (r = 0, 1, \ldots).$$

Some tedious but straightforward algebra, along with direct use of the results in Kosmidis and Firth (2010) for the joint cumulants of $\bar{T}_i$ and $\bar{U}_i$ $(i = 1, \ldots, n)$, gives

$$P_t(\theta) + Q_t(\theta) = \begin{bmatrix} V_{\beta\beta,t} & V_{\beta\gamma,t} \\ V_{\beta\gamma,t}^\top & V_{\gamma\gamma,t} \end{bmatrix} \quad (t = 1, \ldots, p), \tag{12}$$

$$P_{p+s}(\theta) + Q_{p+s}(\theta) = \begin{bmatrix} W_{\beta\beta,s} & W_{\beta\gamma,s} \\ W_{\beta\gamma,s}^\top & W_{\gamma\gamma,s} \end{bmatrix} \quad (s = 1, \ldots, q), \tag{13}$$

where

$$
\begin{aligned}
V_{\beta\beta,t} &= X^\top \Phi^2 D_1 \left(\Phi D_1^2 K_3 + D_1' K_2 \right) X_t^{\mathrm{D}} X \,, \\
V_{\beta\gamma,t} &= X^\top \Phi D_1^2 D_2 \left\{ \Phi \left(M K_3 + \Psi_2 \right) + K_2 \right\} X_t^{\mathrm{D}} Z \,, \\
V_{\gamma\gamma,t} &= Z^\top \Phi D_1 \left\{ D_2^2 \left(M^2 K_3 + 2 M \Psi_2 - \Psi_2 \right) + D_2' \left(M K_2 - \Psi_1 \right) \right\} X_t^{\mathrm{D}} Z
\end{aligned}
$$

and

$$
\begin{aligned}
W_{\beta\beta,s} &= X^\top \Phi D_2 \left\{ \Phi D_1^2 \left(M K_3 + \Psi_2 \right) + D_1' \left(M K_2 - \Psi_1 \right) \right\} Z_s^{\mathrm{D}} X \,, \\
W_{\beta\gamma,s} &= X^\top D_1 D_2^2 \left\{ \Phi \left(M^2 K_3 + 2 M \Psi_2 - \Psi_2 \right) + M K_2 - \Psi_1 \right\} Z_s^{\mathrm{D}} Z \,, \\
W_{\gamma\gamma,s} &= Z^\top D_2^3 \left\{ M^3 K_3 + \left(3 M^2 - 3 M + 1_n \right) \Psi_2 - \Omega_2 \right\} Z_s^{\mathrm{D}} Z \\
&\quad + Z^\top D_2 D_2' \left\{ M^2 K_2 + \Psi_1 - 2 M \Psi_1 - \Omega_1 \right\} Z_s^{\mathrm{D}} Z \,,
\end{aligned}
$$

where $K_3 = \mathrm{diag}\left\{ \kappa_{3,1}, \ldots, \kappa_{3,n} \right\}$, with $\kappa_{3,i} = \mathsf{E}\left\{ \left(\bar{T}_i - \bar{U}_i \right)^3 \right\} = \psi^{(2)}(\phi_i \mu_i) - \psi^{(2)}(\phi_i(1 - \mu_i))$ $(i = 1, \ldots, n)$. Furthermore, C_t^{D} denotes the diagonal matrix with non-zero components the elements of t-th column of a matrix C.

2.4. Implementation in betareg

Support for both bias correction and bias reduction has been added in the principal model fitting function `betareg()` starting from **betareg** 2.4-0. The interface of `betareg()` is essentially the same as described in Cribari-Neto and Zeileis (2010), with merely the addition of a `type` argument that specifies the type of estimator that should be used.

```
betareg(formula, data, subset, na.action, weights, offset,
   link = "logit", link.phi = NULL, type = c("ML", "BC", "BR"),
   control = betareg.control(...), model = TRUE, y = TRUE, x = FALSE, ...)
```

The arguments in the first line (`formula`, `data`, ...) pertain to the data and model specification using a formula that potentially may have two parts pertaining to the mean and the precision submodels, respectively. The arguments `link` and `link.phi` specify the link functions $g_1(\cdot)$ and $g_2(\cdot)$, respectively. The argument `type` controls which of the maximum likelihood (`type = "ML"`), bias-corrected (`type = "BC"`), or bias-reduced (`type = "BR"`) estimates are computed. Finally, `control` is a list of control arguments and `model`, `y`, and `x` control whether the respective data components are included in the fitted model object. For more details on all arguments except `type` see Cribari-Neto and Zeileis (2010).

While the interface of `betareg()` is almost the same as in previous versions, the internal code has been substantially enhanced. Specifically, the optimization via `optim()` is now (optionally) enhanced by an additional Fisher scoring iteration. As in previous versions, the initial optimization of the likelihood is carried out via `optim()`, by default with `method = "BFGS"`, using analytic gradients. In recent versions, this is followed by a Fisher scoring iteration with both analytic gradients and expected information that either neglects or includes the summand $b\left(\theta^{(j)}\right)$ in iteration (7). Thus, the iteration is either used to further improve the

numerical maximization of the likelihood (for `type = "ML"` or `type = "BC"`) or to carry out the bias reduction (for `type = "BR"`) as detailed in Section 2.2. To control the details of the (quasi) Fisher scoring, `betareg.control()` takes two additional arguments `fsmaxit = 200` and `fstol = 1e-8` controlling the maximal number of iterations and convergence tolerance, respectively. If the number of iterations is set to zero (`fsmaxit = 0`), no Fisher scoring is carried out (allowed only for `type = "ML"` and `"BC"`) and thus results from previous versions of **betareg** can be exactly replicated.

3. Beta regression trees

Model-based recursive partitioning (MOB, Zeileis *et al.* 2008) builds on the more widely known method of classification and regression trees (CART, Breiman, Friedman, Olshen, and Stone 1984). As for CART, the idea is to split the sample recursively with respect to available variables (called "partitioning" variables in what follows) in order to capture differences in the response variable. While CART tries to capture differences in the distribution of the response variable (in particular with respect to location) directly, the aim of model-based recursive partitioning is more broadly to capture differences in parameters describing the distribution of the response. In particular, model-based recursive partitioning allows to incorporate regressor variables in a parametric model for the response variable.

Here, we adapt the general MOB framework to the model-based partitioning of beta regressions, called "beta regression trees" for short. The aim is to capture differences in the distribution that are not yet adequately described by the regressor variables through a forward search. Basically, the approach proceeds by (a) fitting a beta regression model, (b) assessing whether its parameters are stable across all partitioning variables, (c) splitting the sample along the partitioning variable associated with the highest parameter instability, (d) repeating these steps until some stopping criterion is met. Thus, interactions and nonlinearities can be incorporated by locally maximizing the likelihood of a partitioned model. More precisely and denoting c_{ij} the j-th partitioning variable ($j = 1, \ldots, l$) for observation i, the steps of the MOB algorithm adapted to beta regression are as follows.

1. Fit a beta regression model with parameters β and γ by maximizing the log-likelihood for all observations y_i in the current sample.

2. Assess whether the parameters β and γ are stable across each partitioning variable c_{ij}.

3. If there is significant parameter instability with respect to at least one of the partitioning variables c_{ij}, split the sample along the variable $j*$ with the strongest association: Choose the breakpoint with highest improvement in the fitted log-likelihood.

4. Repeat steps 1–3 recursively in the resulting subsamples until there is no significant instability any more or the sample size is too small.

The MOB framework of Zeileis *et al.* (2008) is generic in that it requires only the specification of a model with additive objective function for which a central limit theorem holds. Under the usual regularity conditions, the latter requirement is valid for beta regressions. The main building blocks that the MOB algorithm requires are the contributions to the additive objective function (in steps 1 and 3) and to the associated score function (in step 2). For

beta regressions, the objective is the log-likelihood $\ell(\theta)$ and its contributions $\ell_i(\theta)$ are given in (8). By (9) and using the notation in Section 2, the corresponding score (or gradient) contributions have the form

$$
S_i(\theta) = \begin{bmatrix} \mu_i \phi_i d_{1,i} \left(\bar{T}_i - \bar{U}_i \right) x_{i1} \\ \vdots \\ \mu_i \phi_i d_{1,i} \left(\bar{T}_i - \bar{U}_i \right) x_{ip} \\ d_{2,i} \left\{ \mu_i \left(\bar{T}_i - \bar{U}_i \right) + \bar{U}_i \right\} z_{i1} \\ \vdots \\ d_{2,i} \left\{ \mu_i \left(\bar{T}_i - \bar{U}_i \right) + \bar{U}_i \right\} z_{iq} \end{bmatrix} \qquad (i = 1, \dots, n). \tag{14}
$$

The above contributions are employed for testing whether there are significant departures from zero across the partitioning variables. More specifically, MOB uses generalized M-fluctuation tests for parameter instability (Zeileis 2006a; Zeileis and Hornik 2007): fluctuations in numeric variables are assessed with a supLM type test (Andrews 1993) and fluctuations in categorical variables are assessed with a χ^2-type test (Hjort and Koning 2002). For further details and references, see Zeileis *et al.* (2008)[1].

Beta regression trees are implemented in the **betareg** package in function `betatree()` taking the following arguments:

```
betatree(formula, partition, data, subset, na.action,
   link = "logit", link.phi = "log", control = betareg.control(), ...)
```

Essentially, almost all arguments work as for the basic `betareg()` function. The main difference is that a `partition` formula (without left hand side), such as `~ c1 + c2 + c3` has to be provided to specify the vector of partitioning variables $c_i = (c_{i1}, \dots, c_{il})^\top$. As an alternative, `partition` may be omitted when `formula` has three parts on the right hand side, such as `y ~ x1 + x2 | z1 | c1 + c2 + c3`, specifying mean regressors x_i, presicion regressors z_i, and partitioning variables c_i, respectively.

The `betatree()` function takes all arguments and carries out all data preprocessing and then calls the function `mob()` from the **party** package (Hothorn, Hornik, and Zeileis 2006; Hothorn *et al.* 2011). The latter can perform all steps of the MOB algorithm in an object-oriented manner, provided that a suitable model fitting function (optimizing the log-likelihood) is specified and that extractor functions are available for the optimized log-likelihood (8) and the score function (9) at the estimated parameters. For model fitting `betareg.fit()` is employed (through a suitable convenience interface called `betaReg()`) and for extractions the `logLik()` and `estfun()` methods (see also Zeileis 2006b) are leveraged. To control the details of the MOB algorithm – such as the significance level and the minimal subsample size in step 4 – the `...` argument is passed to `mob()`. (Note that this is somewhat different from `betareg()` where `...` is passed to `betareg.control()`.)

---

[1]An example of M-fluctuation tests for parameter instability (also known as structural change) in beta regressions is also discussed in Zeileis (2006a) and replicated in Cribari-Neto and Zeileis (2010). However, this uses a double-maximum type test statistic, not a supLM or χ^2 statistic.

4. Finite mixtures of beta regressions

Finite mixtures are suitable models if the data is assumed to be from different groups, but the group memberships are not observed. If mixture models are fitted one aims at determining the parameters of each group as well as the group sizes. Furthermore, the model can be used to estimate from which group each observation is. In the case of finite mixtures of beta regression models the latent groups can be assumed to differ in their mean and/or in their precision. Furthermore, the group sizes can depend on further covariates.

The mixture model with K components which correspond to K groups is given by

$$h(y; x, z, c, \theta) = \sum_{k=1}^{K} \pi(k; c, \alpha) f(y; g_1^{-1}(x^\top \beta_k), g_2^{-1}(z^\top \gamma_k)), \tag{15}$$

where $h(\cdot; \cdot)$ is the mixture density and $f(y; \mu, \phi)$ is the density of the beta distribution using the mean-precision parameterization shown in (1). Furthermore the component weights $\pi(k; \cdot)$ are nonnegative for all k and sum to one. In what follows the component weights are assumed to be determined from a vector of covariates c by

$$\pi(k; c, \alpha) = \frac{\exp\{c^\top \alpha_k\}}{\sum_{u=1}^{K} \exp\{c^\top \alpha_u\}} \tag{16}$$

with $\alpha_1 \equiv 0$. Without covariates and just a constant ($c = 1$), this reduces to prior probabilities that are fixed across all observations.

Smithson and Segale (2009) and Smithson, Merkle, and Verkuilen (2011) consider finite mixtures of beta regression models to analyze priming effects in judgments of imprecise probabilities. Smithson and Segale (2009) fit mixture models where they investigate if priming has an effect on the size of the latent groups, i.e., they include the information on priming as a predictor variable c. Smithson *et al.* (2011) assume that for at least one component distribution the location parameter is a-priori known due to so-called "anchors". For example, for partition priming, an anchor would be assumed at location $1/K$ if the respondents are primed to believe that there are K possible events. The component distribution for this anchor can be either assumed to follow a beta distribution with known parameters for the mean and the precision or a uniform distribution with known support.

Package **flexmix** (Leisch 2004; Grün and Leisch 2008) implements a general framework for estimating finite mixture models using the EM algorithm. The EM algorithm is an iterative method for ML estimation in a missing data setting. The missing data for mixture models is the information to which component an observation belongs. The EM algorithm exploits the fact that the complete-data log-likelihood for the data and the missing information is easier to maximize. In general for mixture models the posterior probabilities of an observation to be from each component given the current parameter estimates are determined in the E-step. The M-step then consists of maximizing the complete-data log-likelihood where the missing component memberships are replaced by the current posterior probabilities. This implies that different mixture models only require the implementation of a suitable M-step driver. Function `betareg.fit()` provides functionality for weighted ML estimation of beta regression models and hence allows the easy implementation of the M-step.

The function `betamix()` allows to fit finite mixtures of beta regression models using the package **betareg**. It has the following arguments:

```
betamix(formula, data, k, fixed, subset, na.action,
  link = "logit", link.phi = "log", control = betareg.control(...),
  FLXconcomitant = NULL, extra_components,
  verbose = FALSE, ID, nstart = 3, FLXcontrol = list(), cluster = NULL,
  which = "BIC", ...)
```

- Arguments `formula`, `data`, `subset`, `na.action`, `link`, `link.phi` and `control` are the same as for `betareg()`.

  Additionally the formula can also consist of three parts on the right hand side when specifying a concomitant variable model (see below for the `FLXconcomitant` argument).

- Arguments `cluster`, `FLXconcomitant` and `FLXcontrol` are the same as for function `flexmix()` (in the latter two cases without prefix FLX).

  Currently functionality to fit a multinomal logit model for the concomitant variable model is provided by `FLXPmultinom()` with a formula interface for specifying the concomitant variables. To fit a multinomial logit model for the variables `c1` and `c2` use `FLXconcomitant = FLXPmultinom(~ c1 + c2)`. Alternatively, yielding equivalent output, the main model formula can be specified via a three-part formula on the right hand side, e.g., `y ~ x | 1 | c1 + c2` (if there are no covariates for the precision model).

- Argument `k`, `verbose`, `nstart` and `which` are used to specify the repeated runs of the EM algorithm using function `stepFlexmix()`, where `k` is the (vector of) number(s) of mixture components, `nstart` the number of random starting values used, and `which` determines which number of components is kept if `k` is a vector.

- Because the formula for specifying the beta regression model is already a two-part formula, a potential grouping variable is specified via argument `ID` as opposed to when using `flexmix()`.

- Further arguments for the component specific model are `fixed` and `extra_components`. The argument `fixed` can be used to specify the covariates for which parameters are the same over components. This is done via a formula interface. The argument `extra_components` is a list of `"extraComponent"` objects which specify the distribution of the component that needs to be completely specified (via the `type` argument). The parameter values of that distribution are specified through `coef` and `delta`.

  ```
  extraComponent(type = c("uniform", "betareg"), coef, delta,
    link = "logit", link.phi = "log")
  ```

5. Illustrative application

To illustrate the methods introduced above, we consider the analysis of reading accuracy data for nondyslexic and dyslexic Australian children (Smithson and Verkuilen 2006). The data consists of 44 observations of children with ages between eight years and five months and twelve years and three months. For each child, the variables `accuracy` (the score on a reading accuracy test), `iq` (the score on a nonverbal intelligent quotient test, converted to z score),
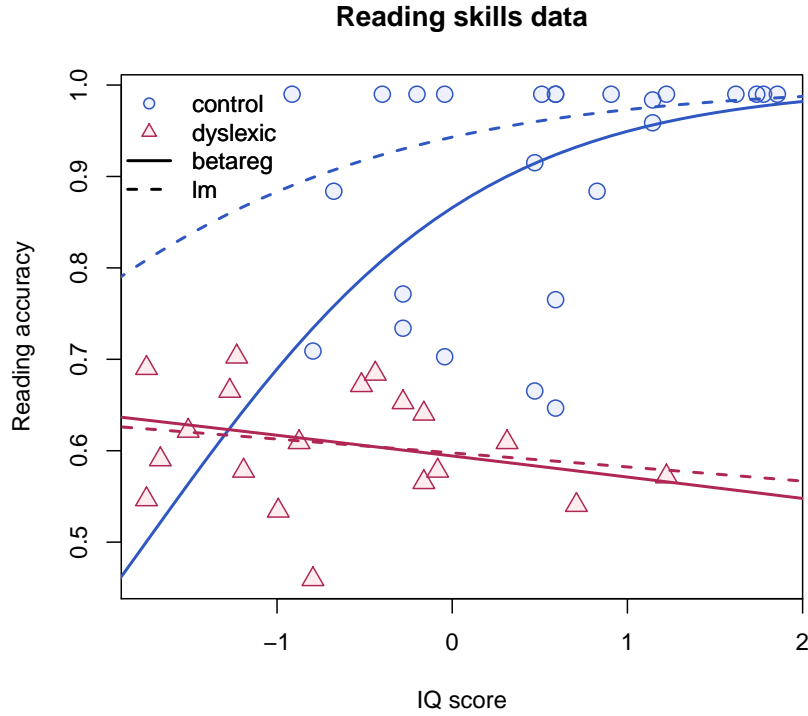
Figure 1: Reading skills data from Smithson and Verkuilen (2006): Linearly transformed reading accuracy by IQ score and dyslexia status (control, blue vs. dyslexic, red). Fitted curves correspond to beta regression (solid) and OLS regression with logit-transformed dependent variable (dashed).

and a binary variable on whether the child is dyslexic were recorded. The 19 dyslexic children have a mean reading accuracy of 0.606 and a mean IQ score of -0.653. The 25 nondyslexic children have a mean reading accuracy of 0.900 and a mean IQ score of 0.497.

Smithson and Verkuilen (2006) investigated whether dyslexic children have a different score on the reading accuracy test when corrected for IQ score. Smithson and Verkuilen (2006) fit a beta regression where the means are linked via the logistic link to main and interaction effects for `iq` and `dyslexic`, and where the precision parameters are linked with a log-link to main effects for the same variables. The fitted model and its comparison to the results of an OLS regression using the logit-transformed `accuracy` as response are given in Cribari-Neto and Zeileis (2010). Figure 1, shows a visualization of the fitted models to briefly highlight the most important findings: In the control group (nondyslexic children), reading skill increases clearly with the IQ score while the variance decreases. In the dyslexic group, reading skills are generally lower and almost unaffected by IQ score.

In what follows, the data is reanalyzed using the methods from Sections 2–4. Initially, the effect of bias to ML inference is assessed. Subsequently, it is illustrated how the differences with respect to dyslexia could have been discovered in a data-driven way. While in the original study dyslexia has, of course, been of prime interest in the model, the data set is used here to illustrate how (a) the two dyslexia groups are automatically selected by recursive partitioning

		Maximum likelihood	Bias correction	Bias reduction
Mean	(Intercept)	1.019	0.990	0.985
		(0.145)	(0.150)	(0.150)
	dyslexia	−0.638	−0.610	−0.603
		(0.145)	(0.150)	(0.150)
	iq	0.690	0.700	0.707
		(0.127)	(0.133)	(0.133)
	dyslexia:iq	−0.776	−0.786	−0.784
		(0.127)	(0.133)	(0.133)
Precision	(Intercept)	3.040	2.811	2.721
		(0.258)	(0.257)	(0.256)
	dyslexia	1.768	1.705	1.634
		(0.258)	(0.257)	(0.256)
	iq	1.437	1.370	1.281
		(0.257)	(0.257)	(0.257)
	dyslexia:iq	−0.611	−0.668	−0.759
		(0.257)	(0.257)	(0.257)
Log-likelihood		66.734	66.334	66.134

Table 1: Comparison of coefficients and standard errors (in parentheses) in the interaction model for reading skills. The ML estimator from `rs_ml`, the BC estimator from `rs_bc`, and the BR estimator from `rs_br` all give very similar results for the mean submodel. In the precision submodel, main effects are slightly damped and the interaction effect is slightly amplified when using BC/BR.

if dyslexia is just one of many covariables and how (b) mixture modeling recovers the dyslexia groups if that covariable is not available at all.

5.1. Bias correction and reduction

To investigate whether the results of Smithson and Verkuilen (2006) may have been affected by severe bias in the ML estimator, all three flavors of estimators are obtained and compared for the model with interactions (both in the mean and precision submodels).

```
R> data("ReadingSkills", package = "betareg")
R> rs_f <- accuracy ~ dyslexia * iq | dyslexia * iq
R> rs_ml <- betareg(rs_f, data = ReadingSkills, type = "ML")
R> rs_bc <- betareg(rs_f, data = ReadingSkills, type = "BC")
R> rs_br <- betareg(rs_f, data = ReadingSkills, type = "BR")
```

The resulting coefficient estimates, standard errors, and log-likelihoods can be displayed using the `summary()` method and are reported in Table 1. All three estimators give very similar results for the mean submodel. In the precision submodel, main effects are slightly dampened and the interaction effect is slightly amplified when using BC/BR. Figure 2 shows the scatter plots of the logarithm of the estimated precision parameters based on the ML, BC, and BR estimates. It is apparent that the logarithms of the estimated precision parameters based on the bias-corrected and bias-reduced estimates are mildly shrunk towards zero. This is a
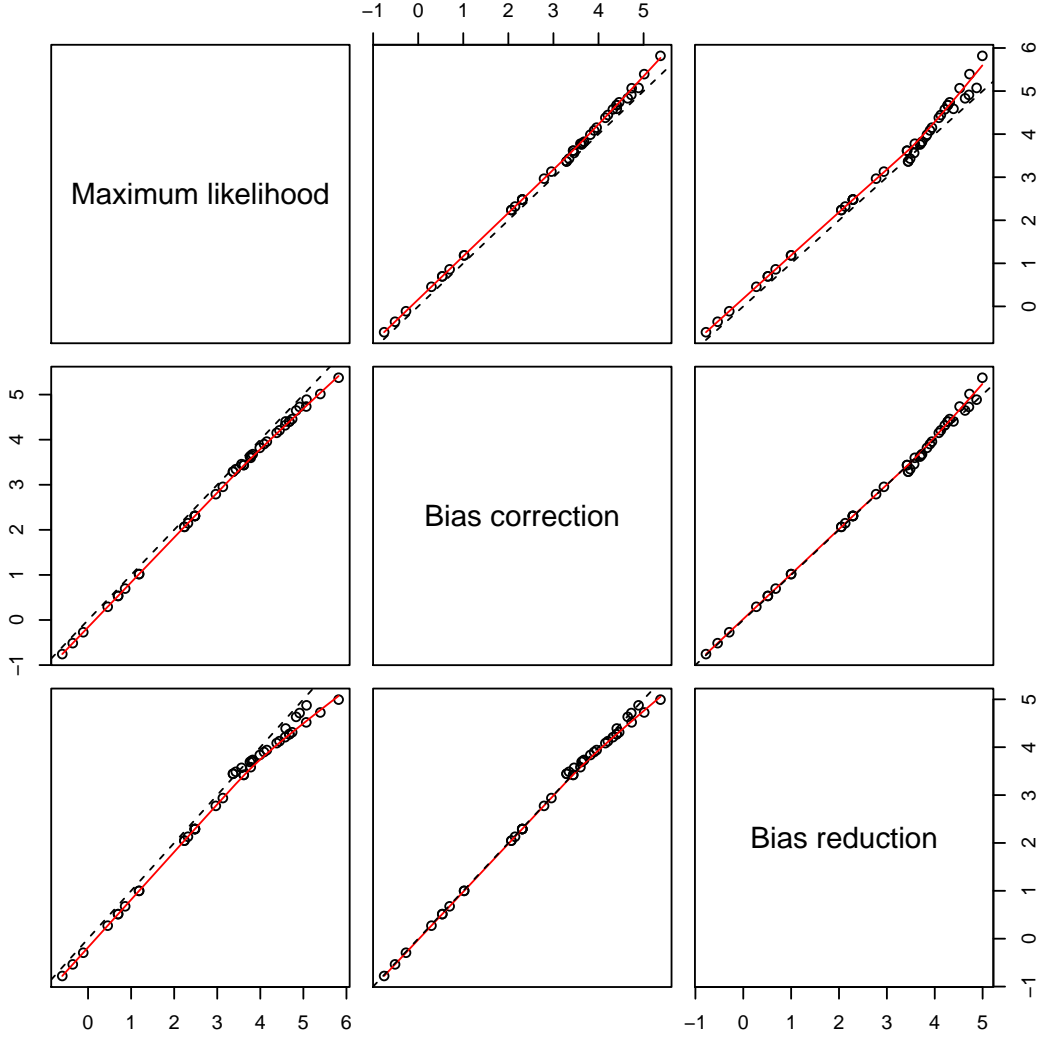
Figure 2: Scatterplots of the logarithm of the estimated precision parameters $\log(\phi_i)$ based on the maximum likelihood, bias-corrected and bias-reduced estimates. The dashed black line is the main diagonal, the solid red line is a scatterplot smoother.

similar but much milder effect compared to the one described in Kosmidis and Firth (2010). The reason that the effect is milder in this particular example relates to the fact that bias for the precision parameters is corrected/reduced on the log-scale where the ML estimator has a more symmetric distribution than on the original scale.

To emphasize that BC/BR may potentially be crucial for empirical analyses, Appendix A replicates the results of Kosmidis and Firth (2010) for a beta regression where substantial upwards bias was detected for the ML estimator of the precision parameter, which in turn causes underestimated asymptotic standard errors; note the direct dependence of the expected information matrix on the precision parameters in (11).

For the reading accuracy data, the similarity of the results in Table 1 between the three different estimation methods and Figure 2 is reassuring and illustrates that analysis based on the ML estimator would not be influenced by bias-related issues. Furthermore, the effect of BC/BR becomes even smaller when the model without interaction in the precision submodel (as chosen by Smithson and Verkuilen 2006) is considered.

5.2. Beta regression tree

For illustrating the use of model-based recursive partitioning methods we assume the following situation: A researcher wants to assess whether the relationship between reading `accuracy` and nonverbal `iq` score is different for some subgroups in the data. Covariates potentially describing these subgroups are available but no prior knowledge how exactly the subgroups can be described by these covariates. For investigating the ability of the tree to select suitable variables for partitioning, `dyslexia` is considered as a partitioning variable along with three additional randomly generated noise variables. One noise variable is drawn from a standard normal distribution, one from a uniform distribution and the third is a categorical variable which takes two different values with equal probability.

```
R> set.seed(1071)
R> n <- nrow(ReadingSkills)
R> ReadingSkills$x1 <- rnorm(n)
R> ReadingSkills$x2 <- runif(n)
R> ReadingSkills$x3 <- factor(sample(0:1, n, replace = TRUE))
```

The model-based tree is fitted using `betatree()`. The first argument is a formula which specifies the model to be partitioned: We have a beta regression where both the mean and the precision of `accuracy` depend on `iq`. The second argument is a formula for the symbolic description of the partitioning variables and both formulas are evaluated using `data`. Additional control arguments for the recursive partitioning method used in `mob_control()` can be specified via the `...` argument. In this case the minimum number of observations in a node is given by `minsplit = 10`.

```
R> rs_tree <- betatree(accuracy ~ iq | iq, ~ dyslexia + x1 + x2 + x3,
+     data = ReadingSkills, minsplit = 10)
```

Alternatively the model could be specified using a three-part formula where the third part is the symbolic description of the partitioning variables.

```
R> rs_tree <- betatree(accuracy ~ iq | iq | dyslexia + x1 + x2 + x3,
+     data = ReadingSkills, minsplit = 10)
```

The returned object is of class `"betatree"` which contains an object of class `"mob"`. All methods for `"mob"` can be reused, e.g., the `print()` method and the `plot()` method (see Figure 3).

```
R> plot(rs_tree)
```

Figure 3 indicates that the data was only split into two subsamples. None of the three noise variables was selected in order to perform a split, but only variable `dyslexia`. This indicates
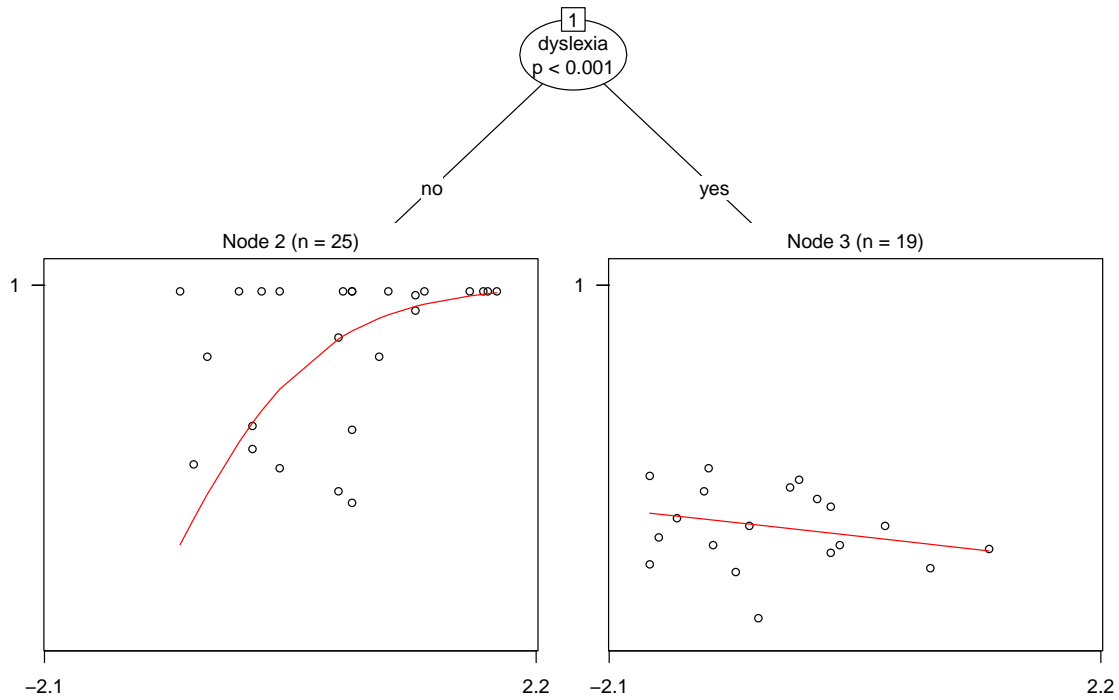
Figure 3: Partitioned beta regression model for the `ReadingSkills` data.

that the relationship between the IQ score and the reading accuracy does not depend on the noise variables as expected. By contrast, the relationship between these two variables differ for dyslexic and nondyslexic children. The beta regressions fitted to each of the two groups of children are illustrated in the two leaf nodes. Note that the fitted models use the IQ score as predictor for the mean and the precision. Hence the results are equivalent to the ML results from Table 1 (where sum contrasts are employed for `dyslexia`). Function `coef()` allows to inspect the parameters of the fitted models, by default in the terminal nodes (nodes 2 and 3).

```
R> coef(rs_tree)
```

```
  (Intercept)          iq (phi)_(Intercept)  (phi)_iq
2   1.6565251  1.46570751          1.272597 2.0478578
3   0.3809322 -0.08622808          4.807662 0.8260329
```

If the fitted object is printed the output indicates after the number of the node, which part of the data according to the split is contained (e.g., `dyslexia == {no}`) or the weights of the observations in the terminal nodes indicated by stars. In the terminal nodes also the estimated parameters of the beta regression models are provided.

```
R> rs_tree
```

```
1) dyslexia == {no}; criterion = 0.999, statistic = 22.687
  2)*  weights = 25
Terminal node model
betaReg fit with coefficients:
     (Intercept)                 iq  (phi)_(Intercept)
           1.657              1.466              1.273
        (phi)_iq
           2.048

1) dyslexia == {yes}
  3)*  weights = 19
Terminal node model
betaReg fit with coefficients:
     (Intercept)                 iq  (phi)_(Intercept)
         0.38093           -0.08623            4.80766
        (phi)_iq
         0.82603
```

The output above confirms that in the nondyslexic group there is a positive association of both mean accuracy and the precision with IQ score. In the dyslexic group, the mean accuracy is generally lower with almost no dependence on IQ score while precision is higher and slightly decreasing with IQ score. Some further details could be revealed by considering for example `summary(rs_tree, node = 3)` that provides the usual regression model summary (unadjusted for recursive partitioning) for the model associated with node 3.

To gain further insight into the recursive construction of the beta regression tree, we use the results of the parameter instability tests in all three nodes. The test statistics together with the corresponding p values can be obtained using function `sctest()` (for structural change test). This indicates which partitioning variables in each node exhibited significant instability and the reason for performing no further split, i.e., either because all parameter instability tests were insignificant (see node 2) or because the node size is too small for a further split (see node 3).

```
R> sctest(rs_tree)


$`1`
              dyslexia        x1        x2        x3
statistic 2.268741e+01 8.5250961 5.569861 3.6273300
p.value   5.847856e-04 0.9094631 0.998710 0.9142006

$`2`
        dyslexia        x1        x2        x3
statistic        0 6.4116324 4.5170190 8.2019125
p.value         NA 0.8412097 0.9751644 0.2325712

$`3`
          [,1]
```

```
statistic     0
p.value       1
```

In node 1 only dyslexia shows significant instability while the noise variables are all insignificant. In node 2, dyslexia cannot be used for splitting anymore and all other variables are still insignificant and thus the partitioning stops. With only 19 observations, node 3 is considered too small to warrant further splitting given that `minsplit = 10` requires that each node contains at least 10 observations and hence no tests are carried out.

5.3. Latent class beta regression

For illustrating the use of finite mixture models we assume the following situation: A researcher wants to assess whether the relationship between reading `accuracy` and nonverbal `iq` score is different for some subgroups in the data without having further covariates potentially describing the groups available. In particular, we assume that the information whether the children are dyslexic or not is not available. Modelling the relationship between reading accuracy and IQ score is now complicated by the fact that latent groups exist in the data where this relationship is different.

The group of nondyslexic children is challenging as some of them essentially have a perfect reading accuracy while for others accuracy is strongly increasing with the IQ score. In a model with observed `dyslexia`, this can be captured by different variances in the two groups. However, issues arise when `dyslexia` is unobserved and a mixture model is employed to infer the groups. Specifically, the subgroup with perfect reading score will typically be selected as one component of the mixture whose variance converges to zero leading to an unbounded likelihood. To address this issue we fit a finite mixture model with three components, where one component is used to capture those children who have a perfect reading accuracy test score. Following Smithson *et al.* (2011) this additional component is assumed to follow a uniform distribution on the interval `coef ± delta`.

```
R> rs_mix <- betamix(accuracy ~ iq, data = ReadingSkills, k = 3,
+    extra_components = extraComponent(type = "uniform",
+      coef = 0.99, delta = 0.01), nstart = 10)
```

The argument `nstart` is set to 10. This implies that the EM algorithm is run 10 times, with each run being randomly initialized. Then only the best solution according to the log-likelihood is returned. In this way, the chance that the global optimum is detected is increased (the EM algorithm is generally only guaranteed to converge to a local optimum and the convergence behaviour depends on the initialization).

The returned fitted model is of class `"betamix"` and has methods for `coef()`, `logLik()`, `print()` and `summary()`. These methods reuse functionality already available for finite mixture models that are directly fitted using `flexmix()` from package **flexmix**. The `print()` method shows the function call and provides information on how many observations are assigned to each of the components based on the values of the posterior probabilities. Furthermore, the convergence status of the EM algorithm is reported, and in the case of convergence, the number of iterations that were performed is shown.

```
R> rs_mix
```

```
Call:
betamix(formula = accuracy ~ iq, data = ReadingSkills,
    k = 3, nstart = 10, extra_components = extraComponent(type = "uniform",
        coef = 0.99, delta = 0.01))

Cluster sizes:
 1  2  3
20 10 14

convergence after 48 iterations
```

The `summary()` method provides more information on the estimated coefficients and their estimated standard errors. For the calculation of the latter, function `optim()` is used for the numerical approximation of the corresponding Hessian matrix.

R> summary(rs_mix)

```
$Comp.1
$Comp.1$mean
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  0.502437   0.083207  6.0384 1.556e-09 ***
iq          -0.047179   0.114318 -0.4127    0.6798
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


$Comp.1$precision
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  4.23880    0.75975  5.5792 2.416e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1



$Comp.2
$Comp.2$mean
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  1.40568    0.26828  5.2396 1.609e-07 ***
iq           0.82554    0.21693  3.8056 0.0001415 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


$Comp.2$precision
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  2.68629    0.45765  5.8697 4.366e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Because only two components are freely estimated and the parameters for the third component were fixed a-priori, the detailed information on the estimated parameters is only provided for
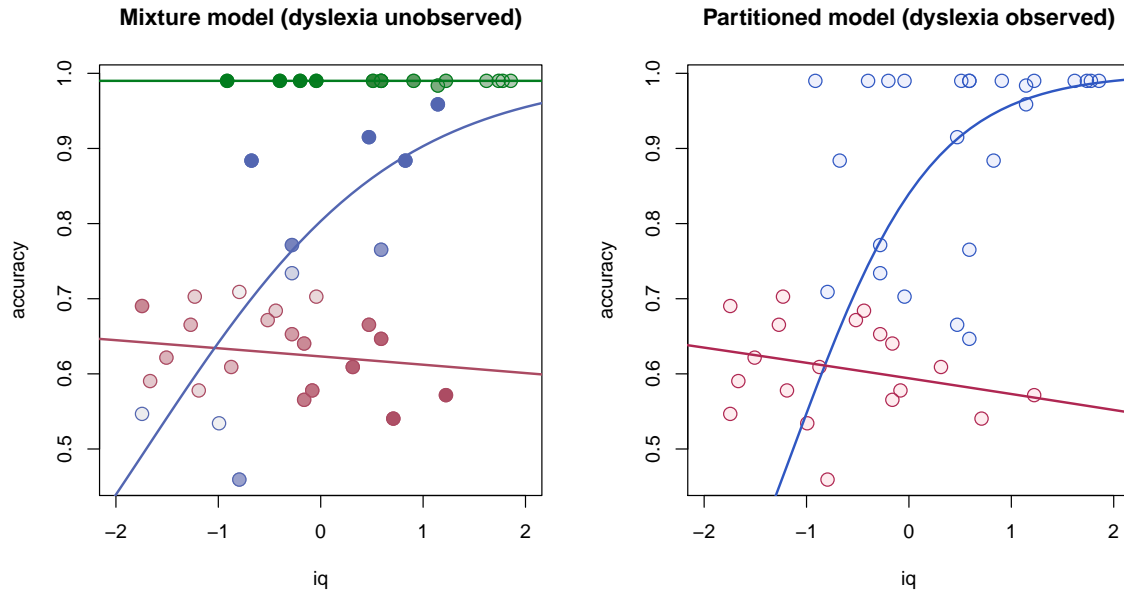
Figure 4: Fitted regression lines for the mixture model with three components and the observations shaded according to their posterior probabilities (left). Fitted regression lines for the partitioned beta regression model with shading according to the observed `dyslexic` variable where nondyslexic and dyslexic children are in blue and red, respectively (right).

components 1 and 2. The regression part for the mean indicates that in the first component the IQ score does not significantly affect the achieved accuracy, while there is a positive significant effect of the IQ score on accuracy in the second component.

A cross-tabulation of the cluster assignments of the mixture model with the variable `dyslexia` indicates that no dyslexic children are assigned to the third component. Furthermore, children assigned to the first component have a high probability (80%) of being dyslexic.

```
R> table(clusters(rs_mix), ReadingSkills$dyslexia)

    no yes
  1  4  16
  2  7   3
  3 14   0
```

The fitted mean regression lines for each of the three components are provided in Figure 4 (left). The observations are shaded according to the magnitude of the corresponding posterior probabilities. The stronger the shading of an observation is in red, the higher the posterior probability for this observation being from the first component is. Blue shading corresponds to the second component and green to the third. For comparison purposes, the right plot in Figure 4 shows the mean regression lines for the dyslexic and nondyslexic children as obtained by recursive partitioning – or equivalently for the model where an interaction with the variable `dyslexic` is specified in the regressions for mean and precision.

The fitted regression lines for the dyslexic children (red) and the latent group capturing the dyslexic children (component 1) are very similar. In contrast, the group of nondyslexic children is modelled differently. With observed dyslexia, the heterogeneity in the control group is captured by differences in the precision submodel, i.e., in the variance. However, for unobserved dyslexia, it is more natural to capture the increased heterogeneity in the control group using two components, one of which would correspond to perfect reading accuracy irrespective of the IQ score.

6. Conclusions

The new extensions of the package **betareg** allow to move beyond classical ML inference when fitting beta regression models. Bias correction and bias reduction of the ML estimates can be useful alternatives when the ML inferences turn out to be unreliable, and actually their ready availability in the package allows users to check how sensitive inferences (standard errors, confidence intervals and Wald tests, in particular) can be to the bias of the ML estimator. Recursive partitioning methods and finite mixture models enable the user to investigate heterogeneity – both observed and unobserved – in the regression model fitted to the whole sample.

For users already familiar with previous versions of the **betareg** package, obtaining the bias-corrected/reduced estimators is straightforward; the user only needs to appropriately specify the `type` argument (which defaults to `"ML"`). For the implementation of the aforementioned extensions some changes and additions in the fitting function `betareg.fit()` were necessary. Specifically, the optimization of the likelihood is now followed by a Fisher scoring iteration. Furthermore, if the bias-reduced or bias-corrected estimates are requested, that Fisher scoring iteration is accordingly modified using a bias adjustment.

To fit beta regression trees and finite mixtures of beta regressions the new functions `betatree()` and `betamix()` are available in package **betareg**. These functions borrow functionality from the packages **party** and **flexmix**. The interface of the two new functions has been designed to be as similar as possible to `betareg()`, in order to facilitate their use by users that are already familiar with the `betareg()` function. For modelling heterogeneity `betareg.fit()` is reused to fit the models in the nodes when beta regression trees are constructed, and in the M-step when finite mixture models are fitted. For this task, only a small amount of additional code — some S4 glue — was necessary to inherit the functionality provided by the **party** and **flexmix** packages to the package **betareg**.

Overall, the increased flexibility of the extended package **betareg** enables users to conveniently check model suitability and appropriateness of the resultant inferences. With the extended package users can easily compare the results from a beta regression model fitted using ML estimation to those using bias correction/reduction, and draw conclusions incorporating observed or unobserved heterogeneity in their models.

Acknowledgments

References

Andrews DWK (1993). "Tests for Parameter Instability and Structural Change with Unknown Change Point." *Econometrica*, **61**, 821–856.

Breiman L, Friedman J, Olshen RA, Stone CJ (1984). *Classification and Regression Trees.* Chapman and Hall, New York.

Cribari-Neto F, Zeileis A (2010). "Beta Regression in R." *Journal of Statistical Software*, **34**(2), 1–24. URL http://www.jstatsoft.org/v34/i02/.

Efron B (1975). "Defining the Curvature of a Statistical Problem (with Applications to Second Order Efficiency)." *The Annals of Statistics*, **3**(6), 1189–1217.

Ferrari SLP, Cribari-Neto F (2004). "Beta Regression for Modelling Rates and Proportions." *Journal of Applied Statistics*, **31**(7), 799–815.

Firth D (1993). "Bias Reduction of Maximum Likelihood Estimates." *Biometrika*, **80**(1), 27–38.

Fox J, Weisberg S (2011). *An R Companion to Applied Regression.* 2nd edition. Sage Publications, Thousand Oaks, CA.

Frühwirth-Schnatter S (2006). *Finite Mixture and Markov Switching Models.* Springer-Verlag, New York.

Grün B, Leisch F (2008). "FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters." *Journal of Statistical Software*, **28**(4), 1–35. URL http://www.jstatsoft.org/v28/i04/.

Hjort NL, Koning A (2002). "Tests for Constancy of Model Parameters Over Time." *Nonparametric Statistics*, **14**, 113–132.

Hothorn T, Bretz F, Westfall P (2008). "Simultaneous Inference in General Parametric Models." *Biometrical Journal*, **50**(3), 346–363.

Hothorn T, Hornik K, Strobl C, Zeileis A (2011). **party**: *A Laboratory for Recursive Partytioning.* R package version 0.9-99996, URL http://CRAN.R-project.org/package=party.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.

Kosmidis I, Firth D (2010). "A Generic Algorithm for Reducing Bias in Parametric Estimation." *Electronic Journal of Statistics*, **4**, 1097–1112.

Leisch F (2004). "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**(8), 1–18. URL http://www.jstatsoft.org/v11/i08/.

Leisch F, Grün B (2012). **flexmix**: *Flexible Mixture Modeling.* R package version 2.3-6, URL http://CRAN.R-project.org/package=flexmix.

McLachlan GJ, Peel D (2000). *Finite Mixture Models*. John Wiley & Sons, New York.

Ospina R, Cribari-Neto F, Vasconcellos KLP (2006). "Improved Point and Interval Estimation for a Beta Regression Model." *Computational Statistics & Data Analysis*, **51**(2), 960–981.

Simas AB, Barreto-Souza W, Rocha AV (2010). "Improved Estimators for a General Class of Beta Regression Models." *Computational Statistics & Data Analysis*, **54**(2), 348–366.

Smithson M, Merkle EC, Verkuilen J (2011). "Beta Regression Finite Mixture Models of Polarization and Priming." *Journal of Educational and Behavioral Statistics*, **36**(6), 804–831.

Smithson M, Segale C (2009). "Partition Priming in Judgments of Imprecise Probabilities." *Journal of Statistical Theory and Practice*, **3**(1), 169–181.

Smithson M, Verkuilen J (2006). "A Better Lemon Squeezer? Maximum-Likelihood Regression with Beta-Distributed Dependent Variables." *Psychological Methods*, **11**(1), 54–71.

Zeileis A (2006a). "Implementing a Class of Structural Change Tests: An Econometric Computing Approach." *Computational Statistics & Data Analysis*, **50**(11), 2987–3008.

Zeileis A (2006b). "Object-Oriented Computation of Sandwich Estimators." *Journal of Statistical Software*, **16**(9), 1–16. URL http://www.jstatsoft.org/v16/i09/.

Zeileis A, Croissant Y (2010). "Extended Model Formulas in R: Multiple Parts and Multiple Responses." *Journal of Statistical Software*, **34**(1), 1–13. URL http://www.jstatsoft.org/v34/i01/.

Zeileis A, Hornik K (2007). "Generalized M-Fluctuation Tests for Parameter Instability." *Statistica Neerlandica*, **61**(4), 488–508.

Zeileis A, Hothorn T (2002). "Diagnostic Checking in Regression Relationships." R *News*, **2**(3), 7–10. URL http://CRAN.R-project.org/doc/Rnews/.

Zeileis A, Hothorn T, Hornik K (2008). "Model-Based Recursive Partitioning." *Journal of Computational and Graphical Statistics*, **17**(2), 492–514.

Zeileis A, Leisch F, Hornik K, Kleiber C (2002). "**strucchange**: An R Package for Testing for Structural Change in Linear Regression Models." *Journal of Statistical Software*, **7**(2), 1–38. URL http://www.jstatsoft.org/v07/i02/.

A. Bias correction/reduction for gasoline yield data

To illustrate how upward bias in the ML estimator of the precision parameter in beta regressions can severely affect inference, results from Kosmidis and Firth (2010) are replicated. All three flavors of estimators (ML, BC, and BR) are computed for the fixed-precision beta regression model considered in Ferrari and Cribari-Neto (2004) (also replicated in Cribari-Neto and Zeileis 2010):

```
R> data("GasolineYield", package = "betareg")
R> gy <- lapply(c("ML", "BC", "BR"), function(x)
+    betareg(yield ~ batch + temp, data = GasolineYield, type = x))
```

The estimate of the precision parameter shrinks considerably when bias correction/reduction is used, indicating a large upwards bias for the ML estimator of ϕ.

```
R> sapply(gy, coef, model = "precision")

   (phi)    (phi)    (phi)
440.2784 261.2061 261.0378
```

while the log-likelihood does not change much

```
R> sapply(gy, logLik)

[1] 84.79756 82.94707 82.94499
```

This results in much larger standard errors (and hence smaller test statistics and larger p values) for all coefficients in the mean part of the model. Table 2 replicates Kosmidis and Firth (2010, Table 1). The picture does also not change much when a log-link is used in the precision model, see below and Table 3 replicating Kosmidis and Firth (2010, Table 3).

	Maximum likelihood		Bias correction		Bias reduction	
β_1	-6.15957	(0.18232)	-6.14837	(0.23595)	-6.14171	(0.23588)
β_2	1.72773	(0.10123)	1.72484	(0.13107)	1.72325	(0.13106)
β_3	1.32260	(0.11790)	1.32009	(0.15260)	1.31860	(0.15257)
β_4	1.57231	(0.11610)	1.56928	(0.15030)	1.56734	(0.15028)
β_5	1.05971	(0.10236)	1.05788	(0.13251)	1.05677	(0.13249)
β_6	1.13375	(0.10352)	1.13165	(0.13404)	1.13024	(0.13403)
β_7	1.04016	(0.10604)	1.03829	(0.13729)	1.03714	(0.13727)
β_8	0.54369	(0.10913)	0.54309	(0.14119)	0.54242	(0.14116)
β_9	0.49590	(0.10893)	0.49518	(0.14099)	0.49446	(0.14096)
β_{10}	0.38579	(0.11859)	0.38502	(0.15353)	0.38459	(0.15351)
β_{11}	0.01097	(0.00041)	0.01094	(0.00053)	0.01093	(0.00053)
ϕ	440.27839	(110.02562)	261.20610	(65.25866)	261.03777	(65.21640)

Table 2: ML, BC and BR estimates and corresponding estimated standard errors (in parentheses) for a logit-linked beta regression model for the gasoline yield data. The precision parameter ϕ is assumed to be equal across the observations.

	Maximum likelihood		Bias correction		Bias reduction	
β_1	-6.15957	(0.18232)	-6.14837	(0.21944)	-6.14259	(0.22998)
β_2	1.72773	(0.10123)	1.72484	(0.12189)	1.72347	(0.12777)
β_3	1.32260	(0.11790)	1.32009	(0.14193)	1.31880	(0.14875)
β_4	1.57231	(0.11610)	1.56928	(0.13978)	1.56758	(0.14651)
β_5	1.05971	(0.10236)	1.05788	(0.12323)	1.05691	(0.12917)
β_6	1.13375	(0.10352)	1.13165	(0.12465)	1.13041	(0.13067)
β_7	1.04016	(0.10604)	1.03829	(0.12767)	1.03729	(0.13383)
β_8	0.54369	(0.10913)	0.54309	(0.13133)	0.54248	(0.13763)
β_9	0.49590	(0.10893)	0.49518	(0.13112)	0.49453	(0.13743)
β_{10}	0.38579	(0.11859)	0.38502	(0.14278)	0.38465	(0.14966)
β_{11}	0.01097	(0.00041)	0.01094	(0.00050)	0.01093	(0.00052)
$\log\phi$	6.08741	(0.24990)	5.71191	(0.24986)	5.61608	(0.24984)

Table 3: ML, BC and BR estimates and corresponding estimated standard errors (in parentheses) for a logit-linked beta regression model for the gasoline yield data. Precision is estimated on the log-scale and is assumed to be equal across the observations.

```
R> data("GasolineYield", package = "betareg")
R> gy2 <- lapply(c("ML", "BC", "BR"), function(x)
+   betareg(yield ~ batch + temp | 1, data = GasolineYield, type = x))
R> sapply(gy2, logLik)

[1] 84.79756 83.79707 83.26777
```

Affiliation:

Bettina Grün
Institut für Angewandte Statistik
Johannes Kepler Universität Linz
Altenbergerstraße 69
4040 Linz, Austria
E-mail: Bettina.Gruen@jku.at
URL: http://ifas.jku.at/gruen/

Ioannis Kosmidis
Department of Statistical Science
University College London
Gower Street
London WC1E 6BT, United Kingdom
E-mail: ioannis@stats.ucl.ac.uk
URL: http://www.ucl.ac.uk/~ucakiko/

Achim Zeileis
Department of Statistics

Universität Innsbruck
Universitätsstr. 15
6020 Innsbruck, Austria
E-mail: `Achim.Zeileis@R-project.org`
URL: `http://eeecon.uibk.ac.at/~zeileis/`