Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 11(1): 1675–1700.
DOI: 10.3934/math.2026070
Received: 04 February 2025
Revised: 03 December 2025
Accepted: 08 December 2025
Published: 19 January 2026

Research article

Unbiased random effects estimation in generalized linear mixed models via likelihood-based boosting

Johanna Gerstmeyer[1], **Elisabeth Bergherr**[1] **and Colin Griesbach**[2,*]

[1] Chair of Spatial Data Science and Statistical Learning, Georg-August-Universität Göttingen, Platz der Göttinger Sieben 3, 37073 Göttingen, Germany

[2] Georg-Elias-Mueller Institute of Psychology, Georg-August-Universität Göttingen, Goßlerstraße 14, 37073 Göttingen, Germany

* **Correspondence:** Email: colin.griesbach@uni-goettingen.de.

Abstract: Boosting techniques present a popular alternative to conventional methods for estimating covariate effects in generalized linear mixed models. Additionally to functionality in high-dimensional data setups, they also offer variable selection. The established framework for boosting in GLMMs tends to exhibit problematic behavior in the selection process when cluster-constant covariates are present, which leads to incorrect estimates. We propose an improved algorithm that rectifies this issue by reworking the updating process of the random effects which already proved successful for linear mixed models, i.e. normally distributed outcomes, and provide additional insights regarding the computation of model complexity. We show the improvements in the quality of estimates via various simulations and data examples.

Keywords: mixed models; statistical learning; clustered data; high dimensional data; variable selection
Mathematics Subject Classification: 62J07

1. Overview

Generalized linear mixed models (GLMMs) have become a popular tool for various applications, most notably longitudinal data obtained from repeated measurements as well as clustered data. However, especially for high-dimensional data, the conventional methods for parameter estimation of GLMMs can be problematic in terms of estimation accuracy and computational effort. Consequently, alternative approaches that can also offer variable selection have been proposed.

One such alternative approach is boosting, which will be shortly outlined in the following with reference to [24]. The basic idea of boosting is to iteratively apply so-called weak base-learners and to combine the results in order to obtain a strong learner that achieves a better prediction. A weak learner is defined as yielding a classification rate of slightly above 50%, whereas a strong learner should achieve a nearly perfect classification. The concept behind the algorithm is to weight observations based on the success of the base-learner in the previous iteration, thus increasing the importance of observations that are difficult to classify.

Boosting was first introduced in the field of machine learning, most prominently in form of the AdaBoost algorithm [8, 9], for classification problems. It was then extended to statistical modelling where it can also be applied to regression problems [4, 10]. The statistical boosting algorithms are separated into gradient boosting and likelihood-based boosting approaches. The idea behind gradient boosting is to fit the base-learner to the negative gradient of the loss function of the previous iterations. Likelihood-based boosting uses base-learners that maximize an overall likelihood in each boosting iteration. Component-wise likelihood-based boosting, which is the type of boosting we will consider in all the following, fits a separate candidate model for each covariate $\boldsymbol{x}_j$ of the covariate matrix $\boldsymbol{X}$ in each iteration and only updates the parameter estimate of the candidate with the largest likelihood. Overall, model-based boosting routines adapt well-established boosting algorithms from machine learning to interpretable statistical models by exchanging base procedures like decision trees with an ensemble of statistical models, e.g. linear effects, splines or random effects.

The advantages of boosting methods include the implicitly performed variable selection as well as their stability in the case of high dimensional data, which also allows for settings where the number of covariates exceeds the number of observations. Thus, boosting poses an attractive alternative to the conventional estimation approaches for GLMMs, which has already been done in several ways in the statistical boosting literature. For the popular `mboost` [17], an additional base-learner modelling random effects was proposed in [18] while likelihood-based boosting routines were transferred to mixed models by [36]. The `bGLMM` algorithm as proposed in [35] is an already established implementation of a component-wise likelihood-based boosting method for generalized linear mixed models. It is contained in the `GMMBoost` package [14]. However, [11] discovered that the `bGLMM` algorithm can be problematic when applied to data where cluster-constant covariates are present. They observed that cluster-varying covariates tend to be favored in the selection and updating process with the random intercept updates partly accounting for the effects actually stemming from cluster-constant covariates.

One can clearly illustrate this issue by analyzing the CD4 data set [1]. It contains data from a trial on 467 patients with human immunodeficiency virus (HIV) infection. The data set contains repeated measurements of the number of CD4 cells per milliliter of blood per patient as the response variable. Explanatory variables are time, treatment, gender and a certain drug intolerance. It also contains a variable *noAIDS/AIDS*, which denotes if the patient fulfills an AIDS-defining condition. The acquired immunodeficiency syndrome (AIDS) is caused by the human immune deficiency virus attacking CD4 cells and decreasing their number. As such, the number of CD4 cells per milliliter of blood is used as a diagnostic criterion for AIDS and is an AIDS-defining condition. While this clear dependency makes the inclusion of the variable in the model somewhat trivial, its estimate should evidently show a significant effect.

Analyzing the CD4 data set with `bGLMM` as well as the conventional maximization method `glmmPQL` available in the `MASS` package we obtain the estimates presented in Table 1. One can see that for the cluster-varying covariate *time*, the estimates are similar, but for the other, all cluster-constant, covariates the estimates differ. The `bGLMM` algorithm has kept almost all the cluster-constant covariates at or close to zero and the covariate *noAIDS/AIDS* is clearly underestimated. Despite its obvious significant effect on the number of CD4 cells, `bGLMM` only shows a relatively weak update of the *noAIDS/AIDS* covariate. In comparison, `glmmPQL` estimates a much higher effect.

Table 1. Fixed effects ($\boldsymbol{\beta}$) estimation results for a generalized linear mixed model with random intercepts on the CD4 data set with `glmmPQL` and `bGLMM`.

	time	treatment	gender	**AIDS**	AZT
`glmmPQL`	-0.028	0.127	-0.061	**1.163**	0.094
`bGLMM`	-0.029	0.000	0.000	**0.202**	0.010

Looking at the random intercept estimates, one can see that `bGLMM` instead captures the effects of the *noAIDS/AIDS* covariate within the random intercept updates. Figure 1 shows that the random intercept estimates for subjects with no AIDS-defining condition are visibly raised.

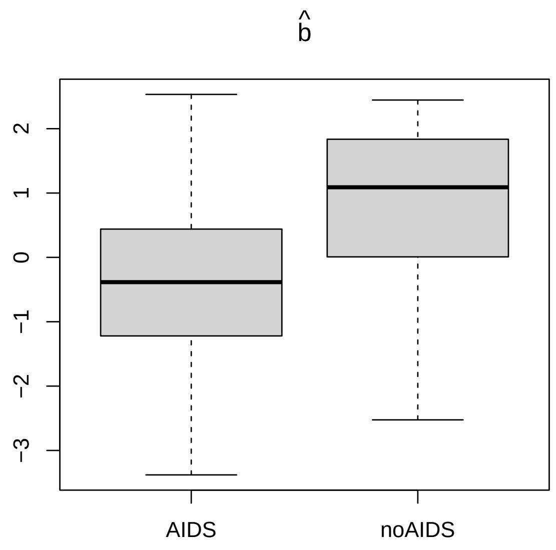


Figure 1. Random intercept estimates by `bGLMM` separated for *noAIDS/AIDS*.

To solve this issue of the treatment of cluster-constant covariates in the boosting process, [11] propose various changes and apply them to a basic linear mixed model (LMM) resulting in the `lbbLMM` (likelihood-based boosting for linear mixed models) algorithm. The general ideas proposed are the following:

- Perform the updates for fixed and for random effects separately in order to ensure a fair selection of the best-performing fixed effect.
- Apply a weak learning rate to the updating scheme of the random effects in order to prevent them from growing too quickly.
- Introduce a correction matrix in order to filter out possible correlations between random effects estimates and observed covariates.

According to [11], the proposed changes show a clear improvement of the performance in the presence of cluster-constant covariates compared to the `bGLMM` algorithm. Since their implementation for LMMs was able to achieve satisfying results, it is sensible to now extend these measures to GLMMs.

The purpose of this work is to establish a revised boosting approach for GLMMs that achieves an overall satisfying performance in problematic settings. We therefore use the foundation of the `bGLMM` algorithm and equip it with the findings from [11] in order to achieve a reliable and well-performing boosting algorithm with unbiased parameter estimates. In addition, we investigate how precisely the information criteria used for tuning the algorithm approximate the model complexity, and finally explore whether simpler methods might also be favorable options. The remainder of this paper is structured as follows:

First, we will give an overview of the model class of GLMMs and the relevant methods and algorithms, including the newly developed `lbbGMM` algorithm. Subsequently, the `lbbGMM` algorithm is evaluated with respect to its overall performance and compared to other established methods using both synthetic and real world data. Finally, we will discuss our findings and their implications as well as limitations.

2. Methods

2.1. Generalized linear mixed models

First, we will formulate the model specification for the class of generalized linear mixed models (GLMMs) as first proposed in [20]. The following outline is based on [5] with some adjustments regarding notation in reference to [35]. We denote $\boldsymbol{y}_i' = (y_{i1}, \dots, y_{in_i})$ collecting $j = 1, \dots, n_i$ observations y_{ij} in $i = 1, \dots, n$ clusters. Furthermore, we denote $\boldsymbol{x}_{ij}' = (x_{ij1}, \dots, x_{ijp})$ and $\boldsymbol{z}_{ij}' = (z_{ij1}, \dots, z_{ijs})$ for p fixed effect and s random effect covariates respectively.

The density of y_{ij}, conditional on the linear predictor η_{ij}, is assumed to belong to an exponential family

$$f(y_{ij}|\boldsymbol{x_{ij}}, \boldsymbol{z_{ij}}, \boldsymbol{b_i}) = \exp\left\{\frac{(y_{ij}\theta_{ij} - \kappa(\theta_{ij}))}{\phi} + c(y_{ij}, \phi)\right\},$$

where $\theta_{ij} = \theta(\mu_{ij})$ denotes the natural parameter, the function $\kappa(\theta_{ij})$ corresponds to the type of exponential family, $c(\cdot)$ is the log normalization constant and ϕ the dispersion parameter.

The means μ_{ij} of the observations y_{ij} are related to the linear predictor η_{ij} through the appropriate link function $g(\cdot)$.

$$g(\mu_{ij}) = \beta_0 + \boldsymbol{x}_{ij}'\boldsymbol{\beta} + \boldsymbol{z}_{ij}'\boldsymbol{b_i} = \eta_{ij}.$$

With the inverse of the link function $h(\cdot) = g^{-1}(\cdot)$ this can alternatively be written as

$$\mu_{ij} = h(\beta_0 + \boldsymbol{x}_{ij}'\boldsymbol{\beta} + \boldsymbol{z}_{ij}'\boldsymbol{b_i}) = h(\eta_{ij}).$$

β_0 denotes the intercept and $\boldsymbol{\beta}' = (\beta_1, \dots, \beta_p)$ the parameter vector for the fixed effects. The random effects are normally distributed with $\boldsymbol{b}_i \sim N(0, \boldsymbol{Q})$ with covariance matrix $\boldsymbol{Q}$.

We arrive at a matrix notation by denoting $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{X}}_1, \dots, \tilde{\boldsymbol{X}}_n]$ with $\tilde{\boldsymbol{X}}_i = [\boldsymbol{1}, \boldsymbol{X}_i]$ and $\boldsymbol{X}_i' = (x_{i1}, \dots, x_{in_i})$ as well as $\tilde{\boldsymbol{Z}} = diag(\tilde{\boldsymbol{Z}}_1, \dots, \tilde{\boldsymbol{Z}}_n)$ with $\boldsymbol{Z}_i' = (z_{i1}, \dots, z_{in_i})$

$$g(\boldsymbol{\mu}) = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}} + \boldsymbol{Z}\boldsymbol{b},$$

where the fixed effects vector is denoted as $\tilde{\boldsymbol{\beta}}' = (\beta_0, \boldsymbol{\beta}')$ and the random effects vector as $\boldsymbol{b}' = (\boldsymbol{b}_1', \dots, \boldsymbol{b}_n')$ with covariance matrix $\boldsymbol{Q_b} = diag(\boldsymbol{Q}, \dots, \boldsymbol{Q})$.

2.2. Estimation approaches

A popular method for estimating GLMMs is the penalized quasi-likelihood (PQL) approach [3,22]. The marginal log-likelihood function corresponding to the density function for the exponential family is

$$l(\boldsymbol{\delta},\boldsymbol{\gamma}) = \sum_{i=1}^{n} \log\left(\int f(\boldsymbol{y}_i|\boldsymbol{\delta},\boldsymbol{\gamma})p(\boldsymbol{b}_i|\boldsymbol{\gamma})d\boldsymbol{b}_i\right),$$

where $\boldsymbol{\delta}' = (\beta_0, \boldsymbol{\beta}, \boldsymbol{b}_i{}')$ and $\boldsymbol{\gamma}' = (\phi, \boldsymbol{\varrho}')$ collects the dispersion parameter and the covariance structure $\boldsymbol{\varrho}$ of $\boldsymbol{Q}$. $p(\boldsymbol{b}_i|\boldsymbol{\gamma})$ denotes the density function of the random effects.

An approximation of this function based on the Laplace method [3] arrives at the penalized log-likelihood function

$$l^{pen}(\boldsymbol{\delta},\boldsymbol{\gamma}) = \sum_{i=1}^{n} \log(f(\boldsymbol{y}_i|\boldsymbol{\delta},\boldsymbol{\gamma})) - \frac{1}{2}\sum_{i=1}^{n} \boldsymbol{b}_i{}'\boldsymbol{Q}^{-1}\boldsymbol{b}_i.$$

Another popular method is the approach based on posterior modes [6]. This approach arrives at an equivalent representation of the penalized log-likelihood above.

Generally, the maximization of this log-likelihood is carried out by using iterative methods, such as Fisher scoring. This is further complicated by the fact that the covariance $\boldsymbol{Q}$ is usually unknown. As a result, the computational effort of the estimation often restricts the model in terms of the number of covariates. Additionally, the estimates can become unstable when many predictor variables are available and no variable selection is performed.

2.3. The `bGLMM` algorithm

As previously mentioned, in our application at hand for GLMMs we consider boosting based on the log-likelihood. The boosting algorithm presented in this section is the `bGLMM` algorithm as proposed by [35]. It performs component-wise boosting, meaning per each iteration step a model that contains the intercept and a linear term, which only includes the r-th fixed effect, is fitted. The corresponding r-th design matrices are denoted by

$$\boldsymbol{X}_{\cdot\cdot r} = [\boldsymbol{1}, \boldsymbol{x}_{\cdot\cdot r}],$$

where $\boldsymbol{x}_{\cdot\cdot r}$ with $r = 1,\dots,p$ denotes the covariate vector for the r-th component and the i-th cluster. This leads to the corresponding predictor

$$\boldsymbol{\eta}_{\cdot\cdot r} = \boldsymbol{X}_{\cdot\cdot r}\tilde{\boldsymbol{\beta}}_r + \boldsymbol{Z}\boldsymbol{b}$$

with $\tilde{\boldsymbol{\beta}}_r{}' = (\beta_0, \beta_r)$ and $\boldsymbol{\delta}_r' = (\beta_0, \beta_r, \boldsymbol{b}')$. In the following we will give an overview of the `bGLMM` algorithm as outlined by [35]: First, the starting values $\hat{\tilde{\boldsymbol{\beta}}}^{(0)}, \hat{\boldsymbol{b}}^{(0)}, \hat{\boldsymbol{Q}}^{(0)}$ are computed by fitting a global intercept model with random effects:

$$g(\mu_{it}) = \beta_0 + \boldsymbol{z}_{it}'\boldsymbol{b}_i.$$

For this the `glmmPQL` function from the `MASS` package is used. Then, for each iteration $l = 1, 2, \dots, l_{max}$ the following is performed: For each fixed effect $r = 1,\dots,p$ the penalized score function $\boldsymbol{s}_r^{pen}(\boldsymbol{\delta})$ and penalized Fisher matrix $\boldsymbol{F}_r^{pen}(\boldsymbol{\delta})$ are computed. They are then used in a simplified Fisher scoring [23]

$$\hat{\boldsymbol{\delta}}_r^{(l)} = (\boldsymbol{F}_r^{pen}(\hat{\boldsymbol{\delta}}^{(l-1)}))^{-1}\boldsymbol{s}_r^{pen}(\hat{\boldsymbol{\delta}}^{(l-1)})$$

to compute an update for the r-th component. The variance-covariance components are replaced by the current estimates $\hat{\boldsymbol{Q}}^{(l-1)}$. From these possible updates the component t that results in the smallest $AIC_r^{(l)}$ or $BIC_r^{(l)}$ is selected. For determining the complexity of the model, the effective degrees of freedom are calculated by using the trace of the Hat-matrix $\boldsymbol{H}$. The vector $(\hat{\boldsymbol{\delta}}_t^{(l)})' = (\hat{\beta}_0^*, \hat{\beta}_t^*, (\hat{\boldsymbol{b}}^*)')$ corresponds to the selected t-th component. Next, the update is performed as

$$\hat{\beta}_0^{(l)} = \hat{\beta}_0^{(l-1)} + \hat{\beta}_0^* \quad , \quad \hat{\boldsymbol{b}}^{(l)} = \hat{\boldsymbol{b}}^{(l-1)} + \hat{\boldsymbol{b}}^*$$

and for $r = 1, \dots, p$ as

$$\hat{\beta}_r^{(l)} = \begin{cases} \hat{\beta}_r^{(l-1)} & \text{, if} \quad r \neq t \\ \hat{\beta}_r^{(l-1)} + \nu \hat{\beta}_r^* & \text{, if} \quad r = t \end{cases}$$

with $0 < \nu \leq 1$ as the learning rate. The resulting $\hat{\boldsymbol{\delta}}^{(l)}$ is then used to update the predictor

$$\hat{\boldsymbol{\eta}}^{(l)} = \boldsymbol{A}\hat{\boldsymbol{\delta}}^{(l)}, \quad \text{where} \quad \boldsymbol{A} = [\tilde{\boldsymbol{X}}, \boldsymbol{Z}].$$

The estimates $\hat{\boldsymbol{Q}}_b$ are obtained either via a REML-type estimate or by employing an approximate EM-algorithm using posterior mode estimates and posterior curvatures. After the maximum number of iterations as specified by l_{max} has been completed, the optimal number of iterations l_{opt} is determined by calculating the information criterion for each iteration. The degrees of freedom are again obtained via the Hat-matrix $\boldsymbol{H}$. The iteration that leads to the smallest $AIC^{(l)}$ or $BIC^{(l)}$ is selected as optimal and with it its corresponding $\hat{\boldsymbol{\delta}}^{(l_{opt})}$, $\hat{\boldsymbol{Q}}^{(l_{opt})}$ and $\hat{\boldsymbol{\mu}}^{(l_{opt})}$.

2.4. *lbbGMM algorithm*

Extending the previously outlined measures proposed by [11] from LMMs to GLMMs leads to the new `lbbGMM` (likelihood-based boosting for generalized linear mixed models) algorithm. It generally follows the same procedure as the `bGLMM` algorithm outlined in the previous section, while also integrating the new changes as introduced by the `lbbLMM` algorithm. In the following, we will present an overview of the `lbbGMM` algorithm and discuss some notable aspects in more detail. The algorithm and all corresponding analyses are conducted using `R` [29].

2.4.1. Algorithm outline

Algorithm `lbbGMM`

- **Initialize** $\hat{\tilde{\boldsymbol{\beta}}}^{(0)}, \hat{\tilde{\boldsymbol{b}}}^{(0)}, \hat{\boldsymbol{Q}}^{(0)}$ with starting values and apply the correction matrix (see 2.4.2) to the random effects vector.
- **for** $l = 1$ to $l_{\max}$ **do**
  Fixed effects update:
  - Calculate the penalized score function and Fisher matrix for the r-th fixed effect with corresponding $\boldsymbol{\beta}_r = (\hat{\beta}_0^{(l-1)}, \hat{\beta}_r^{(l-1)})'$ as

$$\boldsymbol{s}_r^{pen}(\boldsymbol{\beta}_r) = \frac{\partial l^{pen}}{\partial \boldsymbol{\beta}_r} \quad \text{and} \quad \boldsymbol{F}_r^{pen}(\boldsymbol{\beta}_r) = -\mathbb{E}\left[\frac{\partial^2 l^{pen}}{\partial \boldsymbol{\beta}_r \partial \boldsymbol{\beta}_r'}\right].$$

- Use Fisher scoring to obtain p possible updates

$$\hat{\boldsymbol{u}}_r^{(l)} = (\boldsymbol{F}_r^{pen}(\hat{\boldsymbol{\beta}}_r))^{-1}\boldsymbol{s}_r^{pen}(\hat{\boldsymbol{\beta}}_r).$$

- Select the best performing component as the one minimizing the $AIC_r^{(l)}$ or $BIC_r^{(l)}$.
- Perform the update of the fixed effect coefficients and the intercept based on the selection result $\hat{\boldsymbol{u}}_*^{(l)} = (\hat{u}_0^{(l)}, \hat{u}_*^{(l)})$:

$$\hat{\beta}_0^{(l)} = \hat{\beta}_0^{(l-1)} + \nu\hat{u}_0^{(l)}$$

$$\hat{\beta}_r^{(l)} = \begin{cases} \hat{\beta}_r^{(l-1)} & \text{, if} \quad r \neq * \\ \hat{\beta}_r^{(l-1)} + \nu\hat{u}_*^{(l)} & \text{, if} \quad r = * \end{cases}$$

- Calculate the penalized score function and Fisher matrix for the r-th fixed effect with corresponding $\boldsymbol{\beta}_r = (\hat{\beta}_0^{(l-1)}, \hat{\beta}_r^{(l-1)})'$ as

$$\boldsymbol{s}_r^{pen}(\boldsymbol{\beta}_r) = \frac{\partial l^{pen}}{\partial \boldsymbol{\beta}_r} \quad \text{and} \quad \boldsymbol{F}_r^{pen}(\boldsymbol{\beta}_r) = -\mathbb{E}\left[\frac{\partial^2 l^{pen}}{\partial \boldsymbol{\beta}_r \partial \boldsymbol{\beta}_r'}\right].$$

- Use Fisher scoring to obtain p possible updates

$$\hat{\boldsymbol{u}}_r^{(l)} = (\boldsymbol{F}_r^{pen}(\hat{\boldsymbol{\beta}}_r))^{-1}\boldsymbol{s}_r^{pen}(\hat{\boldsymbol{\beta}}_r).$$

- Select the best performing component as the one minimizing the $AIC_r^{(l)}$ or $BIC_r^{(l)}$.
- Perform the update of the fixed effect coefficients and the intercept based on the selection result $\hat{\boldsymbol{u}}_*^{(l)} = (\hat{u}_0^{(l)}, \hat{u}_*^{(l)})$:

$$\hat{\beta}_0^{(l)} = \hat{\beta}_0^{(l-1)} + \nu\hat{u}_0^{(l)}$$

$$\hat{\beta}_r^{(l)} = \begin{cases} \hat{\beta}_r^{(l-1)} & \text{, if} \quad r \neq * \\ \hat{\beta}_r^{(l-1)} + \nu\hat{u}_*^{(l)} & \text{, if} \quad r = * \end{cases}$$

Random effects update:

- Calculate the penalized score function and Fisher matrix for the random effects as

$$\boldsymbol{s}_{ran}^{pen}(\boldsymbol{b}) = \frac{\partial l^{pen}}{\partial \boldsymbol{b}} \quad \text{and} \quad \boldsymbol{F}_{ran}^{pen}(\boldsymbol{b}) = -\mathbb{E}\left[\frac{\partial^2 l^{pen}}{\partial \boldsymbol{b} \partial \boldsymbol{b}'}\right].$$

- Perform a simultaneous weak update of the random coefficients by applying a learning rate $0 < \nu_b \leq 1$ and the correction matrix $\boldsymbol{C}$ (see 2.4.2) to the Fisher scoring:

$$\hat{\boldsymbol{b}}^{(l)} = \hat{\boldsymbol{b}}^{(l-1)} + \nu_b \boldsymbol{C}(\boldsymbol{F}_{ran}^{pen}(\hat{\boldsymbol{b}}^{(l-1)}))^{-1}\boldsymbol{s}_{ran}^{pen}(\hat{\boldsymbol{b}}^{(l-1)}).$$

Variance-covariance-components update:

- Update the variance-covariance components $\boldsymbol{Q}$.

– Compute the model error used for obtaining $\boldsymbol{\Sigma}^{(l)}$.

end for

- **Tune** the algorithm after l_{max} total iterations by determining the optimal number of iterations l_{opt} via some pre-chosen information criteria and return $\hat{\boldsymbol{\beta}}^{(l_{opt})}, \hat{\boldsymbol{b}}^{(l_{opt})}, \hat{\boldsymbol{Q}}^{(l_{opt})}$ as final estimates.

2.4.2. Computational details

In this subsection, single aspects of the `lbbGMM` algorithm proposed in Section 2.4.1 are highlighted in more detail.

Starting values. We initialize $\hat{\tilde{\boldsymbol{\beta}}}^{(0)}, \tilde{\boldsymbol{b}}^{(0)}, \hat{\boldsymbol{Q}}^{(0)}$ by estimating a global intercept model with random effects via the `MASS` package

$$g(\mu_{ij}) = \beta_0 + \boldsymbol{z}'_{ij}\boldsymbol{b}_i.$$

For the random effects the correction matrix $\boldsymbol{C}$ is additionally applied, so that

$$\hat{\boldsymbol{b}}^{(0)} = \boldsymbol{C}\hat{\tilde{\boldsymbol{b}}}^{(0)}.$$

Fixed effects boosting process. The fixed and random effects updates are decoupled from one another, with the fixed effect update being performed first. The penalized score function and Fisher matrix for the r-th fixed effect with corresponding $\boldsymbol{\beta}_r = (\hat{\beta}_0^{(l-1)}, \hat{\beta}_r^{(l-1)})'$ are calculated as follows [6]:

$$\begin{aligned} \boldsymbol{s}_r^{pen}(\boldsymbol{\beta}_r) &= \frac{\partial l^{pen}}{\partial \boldsymbol{\beta}_r} \\ &= \tilde{\boldsymbol{X}}'_r \boldsymbol{D}\boldsymbol{\Sigma}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{F}_r^{pen}(\boldsymbol{\beta}_r) &= -\mathbb{E}\left[\frac{\partial^2 l^{pen}}{\partial \boldsymbol{\beta}_r \partial \boldsymbol{\beta}'_r}\right] \\ &= \tilde{\boldsymbol{X}}'_r \boldsymbol{W} \tilde{\boldsymbol{X}}_r \quad , \end{aligned}$$

with $\boldsymbol{W} = \boldsymbol{D}\boldsymbol{\Sigma}^{-1}\boldsymbol{D}'$ where $\boldsymbol{D} = diag(\boldsymbol{D}_1, \ldots, \boldsymbol{D}_n)$ with $\boldsymbol{D}_i = \dfrac{\partial h(\boldsymbol{\eta}_i)}{\partial \boldsymbol{\eta}}$, and $\boldsymbol{\Sigma} = diag(\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_n)$ with $\boldsymbol{\Sigma}_i = diag(\hat{\sigma}^2)$.

Using Fisher scoring this results in p possible updates, where the best performing component is decided as the one minimizing an information criterion such as Akaike's information criterion (AIC) [2] or the Bayesian information criterion (BIC) [30]. For our purposes they are given by:

$$AIC_r^{(l)} = -2\sum_{i=1}^{n} \log(f(\boldsymbol{y}_i|\hat{\boldsymbol{\mu}}_{i\cdot r}^{(l)})) + 2\mathrm{df}_r^{(l)},$$

$$BIC_r^{(l)} = -2\sum_{i=1}^{n} \log(f(\boldsymbol{y}_i|\hat{\boldsymbol{\mu}}_{i\cdot r}^{(l)})) + \log(n)\mathrm{df}_r^{(l)}.$$

Where for the case of Poisson data $\log(f(\boldsymbol{y}_i|\hat{\boldsymbol{\mu}}_{i\cdot r}^{(l)}))$ takes the form of

$$\log(f(\boldsymbol{y}_i|\hat{\boldsymbol{\mu}}_{i\cdot r}^{(l)})) = \sum_{j=1}^{n_i} y_{ij}\log(\hat{\mu}_{ijr}^{(l)}) - \hat{\mu}_{ijr}^{(l)}$$

and for binary responses

$$\log(f(\boldsymbol{y}_i|\hat{\boldsymbol{\mu}}_{i\cdot r}^{(l)})) = \sum_{j=1}^{n_i} y_{ij}\log(\hat{\mu}_{ijr}^{(l)}) + (1-y_{ij})\log(1-\hat{\mu}_{ijr}^{(l)}).$$

C. Griesbach et al. [11] obtained the respective degrees of freedom (df) by using an approximation consisting of the sum of the variance-covariance parameters and the non-zero components of $\boldsymbol{\beta}$. The alternate approach of using the trace of the Hat-matrix $\boldsymbol{H}$ to obtain the df, similarly to the `bGLMM` algorithm, is outlined in detail later on.

Based on the selection result we obtain $\hat{\boldsymbol{u}}_*^{(l)} = (\hat{u}_0^{(l)}, \hat{u}_*^{(l)})$ and perform the update of the fixed effect coefficients and the intercept:

$$\hat{\beta}_0^{(l)} = \hat{\beta}_0^{(l-1)} + \nu\hat{u}_0^{(l)},\ \ \hat{\beta}_r^{(l)} = \begin{cases}\hat{\beta}_r^{(l-1)} & \text{, if} \quad r \neq * \\ \hat{\beta}_r^{(l-1)} + \nu\hat{u}_*^{(l)} & \text{, if} \quad r = *\end{cases}$$

Random Effects Update. Following this, the process for updating the random effects is performed. We calculate the penalized score function and Fisher matrix for the random effects as follows:

$$\boldsymbol{s}_{ran}^{pen}(\boldsymbol{b}) = \frac{\partial l^{pen}}{\partial \boldsymbol{b}} = \boldsymbol{Z}'\boldsymbol{D}\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}) - \boldsymbol{Q_b}^{-1}\boldsymbol{b}$$

and

$$\boldsymbol{F}_{ran}^{pen}(\boldsymbol{b}) = -\mathbb{E}\left[\frac{\partial^2 l^{pen}}{\partial\boldsymbol{b}\partial\boldsymbol{b}'}\right] = \boldsymbol{Z}'\boldsymbol{W}\boldsymbol{Z} + \boldsymbol{Q_b}^{-1}.$$

By applying a learning rate $0 < \nu_b \leq 1$ to the Fisher scoring, we then arrive at a simultaneous weak update of the random coefficients. In this step, we also apply the aforementioned correction matrix $\boldsymbol{C}$ that was proposed by [11] and is discussed in more detail in the next section.

Correction Matrix. The derivation of the correction matrix $\boldsymbol{C}$ does not require any changes to extend from LMMs to GLMMs. Therefore, we will provide an overview of its original derivation as presented by [11]. For the s-th random effect $\tilde{\boldsymbol{b}} = (b_{s1}, \ldots, b_{sn})'$ with $s = 1, \ldots, q$ its correction matrix $\boldsymbol{C}_s$ is constructed as

$$\boldsymbol{C}_s = \boldsymbol{X}_{cs}(\boldsymbol{X}_{cs}'\boldsymbol{X}_{cs})^{-1}\boldsymbol{X}_{cs}'.$$

For random intercepts, the matrix $\boldsymbol{X}_{cs}$ contains p_s columns and n rows with p_s referring to the number of cluster-constant covariates. The n rows represent one observation of each cluster. For random slopes, the matrix contains a column of ones. By using distinct matrices, the random intercepts or slopes are corrected independently from one another with $(\boldsymbol{I}_n - \boldsymbol{C}_s)\tilde{\boldsymbol{b}}_s$ taking out any correlations between the random effect and the covariates contained in $\boldsymbol{X}_{cs}$. To obtain a single correction matrix that corrects every random effect simultaneously, the matrix $\boldsymbol{C}$ is defined as

$$\boldsymbol{C} = \boldsymbol{P}^{-1}(\boldsymbol{I}_{nq} - \tilde{\boldsymbol{C}})\boldsymbol{P}$$

with block diagonal matrix

$$\tilde{\boldsymbol{C}} = diag(\boldsymbol{C}_1, \ldots, \boldsymbol{C}_q)$$

and permutation matrix $\boldsymbol{P}$ so that

$$\boldsymbol{P}b = \tilde{\boldsymbol{b}} = (\tilde{\boldsymbol{b}}_1', \ldots, \tilde{\boldsymbol{b}}_q')'$$

holds.

Updating Variance-Covariance-Components. After the separate updates of fixed and random effects, the variance-covariance components $\boldsymbol{Q}$ are updated by employing an approximate EM-algorithm using the posterior curvatures $\boldsymbol{F}_i^{ran}$ of the random effects model

$$\hat{\boldsymbol{Q}} = \frac{1}{n} \sum_{i=1}^{n} \left((\boldsymbol{F}_i^{ran})^{-1} + \hat{b}_i \hat{b}_i' \right).$$

The model error for obtaining $\boldsymbol{\Sigma}^{(l)}$ is computed as

$$(\hat{\sigma}^2)^{(l)} = Var(\boldsymbol{dev}^{(l)})$$

with $\boldsymbol{dev}^{(l)}$ referring to the deviance residuals of the current model iteration. The deviance residuals for the three distributions of consideration are calculated as follows (compare [26]):

(1) Gaussian:

$$\boldsymbol{dev}_g^{(l)} = \boldsymbol{y} - \hat{\boldsymbol{\mu}}^{(l)}.$$

(2) Poisson:

$$\boldsymbol{dev}_p^{(l)} = sign(\boldsymbol{y} - \hat{\boldsymbol{\mu}}^{(l)}) \sqrt{2\left(\boldsymbol{y} \log\left(\frac{\boldsymbol{y}}{\hat{\boldsymbol{\mu}}^{(l)}}\right) - (\boldsymbol{y} - \hat{\boldsymbol{\mu}}^{(l)})\right)}.$$

(3) Bernoulli:

$$\boldsymbol{dev}_b^{(l)} = sign(\boldsymbol{y} - \hat{\boldsymbol{\mu}}^{(l)}) \sqrt{2\left(\boldsymbol{y} \log\left(\frac{\boldsymbol{y}}{\hat{\boldsymbol{\mu}}^{(l)}}\right) + (1 - \boldsymbol{y}) \log\left(\frac{1 - \boldsymbol{y}}{1 - \hat{\boldsymbol{\mu}}^{(l)}}\right)\right)}.$$

Tuning. The main tuning parameter of model-based boosting routines is usually given via the number of total boosting iterations [25]. The optimal number of iterations l_{opt} is determined as the one that minimizes the $AIC^{(l)}$ or $BIC^{(l)}$. Corresponding to l_{opt} the other optimal parameter estimates are determined as well. Please note that we rely solely on information criteria as surrogate to computationally more burdensome routines like cross validation or bootstrapping. Considering the AIC is, however, asymptotically equivalent to cross validation which also holds in the case of mixed-effects models [7]. For obtaining the effective degrees of freedom necessary for the computation of said information criteria, we employ the trace of the Hat-matrix as detailed in the following.

Hat matrix. The `bGLMM` algorithm uses the trace of the Hat-matrix $\boldsymbol{H}$ to infer the effective degrees of freedom and consequently the complexity of the model. This is a standard tactic in linear modelling [31] that has been employed in various applications such as smoothing [37] and generalized additive models [5]. Here, it is employed for both the update selection process as well as the determination of l_{opt}. In the following section we will present a concise intuition for the derivation of the Hat-matrix. For a more detailed version compare [34, 36].

By employing Taylor approximation of first order, one can write

$$\hat{\boldsymbol{\mu}}_{\cdot\cdot r}^{(l)} \approx \hat{\boldsymbol{\mu}}^{(l-1)} + \boldsymbol{M}_r^{(l)}(\boldsymbol{y} - \hat{\boldsymbol{\mu}}^{(l-1)})$$

with $\boldsymbol{M}_r^{(l)} = \boldsymbol{\Sigma}_{(l)}^{1/2}\boldsymbol{S}_r^{(l)}\boldsymbol{\Sigma}_{(l)}^{-1/2}$, where $\boldsymbol{S}_r^{(l)}$ denotes the usual generalized ridge regression hat-matrix.

This can be expressed as a recursive process

$$\hat{\boldsymbol{\mu}}_{\cdot\cdot r}^{(l)} \approx \hat{\boldsymbol{\mu}}^{(l-1)} + \boldsymbol{M}_r^{(l)}[(\boldsymbol{y} - \hat{\boldsymbol{\mu}}^{(l-2)}) - \boldsymbol{M}_t^{(l-1)}(\boldsymbol{y} - \hat{\boldsymbol{\mu}}^{(l-2)})]$$

with base case $\hat{\boldsymbol{\mu}}^{(0)} \approx \boldsymbol{M}^{(0)}\boldsymbol{y}$.

This recursive process can be used to obtain the Hat-matrix $\boldsymbol{H}_r^{(l)}$, for which the approximation

$$\hat{\boldsymbol{\mu}}_{\cdot\cdot r}^{(l)} \approx \boldsymbol{H}_r^{(l)}\boldsymbol{y}$$

holds.

Since in our application the update selection process is limited to the fixed effects, the construction of the Hat-matrix differs from the one outlined by [35]. Thus, in the following we will derive the appropriate Hat-matrix for our purposes, corresponding to the l-th iteration step and the r-th fixed effect component.

The Hat-matrix for the global intercept model used in the initialization of the parameters is computed as

$$\boldsymbol{M}^{(0)} = \boldsymbol{I}_N - (\boldsymbol{I}_N - \boldsymbol{M}_{rd}^{(0)})(\boldsymbol{I}_N - \boldsymbol{M}_{fx}^{(0)})$$

with

$$\boldsymbol{M}_{rd}^{(0)} = \boldsymbol{Z}\boldsymbol{C}(\boldsymbol{Z}'\boldsymbol{W}\boldsymbol{Z} + \boldsymbol{Q}_b^{-1})^{-1}\boldsymbol{Z}'\boldsymbol{W}$$

and

$$\boldsymbol{M}_{fx}^{(0)} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W},$$

where $\boldsymbol{X} = \mathbf{1}_N$ and $\boldsymbol{W} = \boldsymbol{D}\boldsymbol{\Sigma}^{-1}\boldsymbol{D}'$.

As described above, the Hat-matrix of each r-th fixed effect is derived in a recursive manner as

$$\boldsymbol{H}_r^{(l)} = \boldsymbol{I}_N - (\boldsymbol{I}_N - \boldsymbol{M}_{f,r}^{(l)})(\boldsymbol{I}_N - \boldsymbol{M}_t^{(l-1)})(\boldsymbol{I}_N - \boldsymbol{M}_t^{(l-2)})\dots(\boldsymbol{I}_N - \boldsymbol{M}^{(0)}),$$

where

$$\boldsymbol{M}_{f,r}^{(l)} = \boldsymbol{I}_N - (\boldsymbol{I}_N - \nu\boldsymbol{S}_r^{(l)})$$

and

$$\boldsymbol{S}_r^{(l)} = \boldsymbol{\Sigma}_{(l)}^{1/2}\boldsymbol{W}_{(l)}^{1/2}\tilde{\boldsymbol{X}}_r(\tilde{\boldsymbol{X}}_r\boldsymbol{W}_{(l)}\tilde{\boldsymbol{X}}_r')^{-1}\tilde{\boldsymbol{X}}_r\boldsymbol{W}_{(l)}^{1/2}\boldsymbol{\Sigma}_{(l)}^{-1/2}$$

with $\tilde{\boldsymbol{X}}_r = [\mathbf{1}, \boldsymbol{X}_r]$.

The $\boldsymbol{M}_t^{(l-h)}$ corresponds to the selected fixed effect t of the respective previous iteration with interval $h = 1, 2, \dots$ and is computed as

$$\boldsymbol{M}_t^{(l-h)} = \boldsymbol{I}_N - (\boldsymbol{I}_N - \nu_b\boldsymbol{S}_{rand}^{(l-h)})(\boldsymbol{I}_N - \nu\boldsymbol{S}_{f,t}^{(l-h)})$$

with

$$\boldsymbol{S}_{f,t}^{(l-h)} = \boldsymbol{\Sigma}_{(l-h)}^{1/2}\boldsymbol{W}_{(l-h)}^{1/2}\tilde{\boldsymbol{X}}_t(\tilde{\boldsymbol{X}}_t\boldsymbol{W}_{(l-h)}\tilde{\boldsymbol{X}}_t')^{-1}\tilde{\boldsymbol{X}}_t\boldsymbol{W}_{(l-h)}^{1/2}\boldsymbol{\Sigma}_{(l-h)}^{-1/2}$$

and

$$\boldsymbol{S}_{rand}^{(l-h)} = \boldsymbol{\Sigma}_{(l-h)}^{1/2}\boldsymbol{W}_{(l-h)}^{1/2}\boldsymbol{Z}\boldsymbol{C}(\boldsymbol{Z}'\boldsymbol{W}_{(l-h)}\boldsymbol{Z} + \boldsymbol{Q}_{b,(l-h)}^{-1})^{-1}\boldsymbol{Z}'\boldsymbol{W}_{(l-h)}^{1/2}\boldsymbol{\Sigma}_{(l-h)}^{-1/2}.$$

In order to obtain the degrees of freedom in the selection of the optimal iteration, the Hat-matrix is computed in a similar manner as

$$\boldsymbol{H}^{(l)} = \boldsymbol{I}_N - (\boldsymbol{I}_N - \boldsymbol{M}_j^{(l)})(\boldsymbol{I}_N - \boldsymbol{M}_j^{(l-1)})(\boldsymbol{I}_N - \boldsymbol{M}_t^{(l-2)})\dots(\boldsymbol{I}_N - \boldsymbol{M}^{(0)}).$$

The $\boldsymbol{M}_t^{(l-h)}$ with $h = 0, 1, \dots$ corresponds to the best performing fixed effect for the current iteration as well as for the respective previous iterations. The computation is the same as noted above.

Standard errors. Following [34], approximate standard errors can be computed for each iteration l based on the global hat matrix $\boldsymbol{H}^{(l)}$ which also serves for tuning. Let $\boldsymbol{C}_r^{(l)}$ denote the matrix that achieves the final coefficient estimate for component $\hat{\beta}_r^{(l)}$, i.e. $\hat{\beta}_r^{(l)} \approx \boldsymbol{C}_r^{(l)}\boldsymbol{y}$. See [13, p. 40] for a detailed derivation of $\boldsymbol{C}_r^{(l)}$. With $cov(\boldsymbol{y}) = \boldsymbol{\Sigma}_l$,

$$cov(\boldsymbol{C}_r^{(l)}\boldsymbol{y}) = \boldsymbol{C}_r^{(l)}cov(\boldsymbol{y})(\boldsymbol{C}_r^{(l)})^T$$

yields approximate standard errors for $\hat{\beta}_r^{(l)}$. However, this procedure neglects any uncertainties arising from the variable selection process and, thus, may lead to biased estimates. An alternative approach relies on empirical bootstrapping, where data is resampled B times with the cluster-structure preserved. Standard errors can be derived as empirical estimates from the B sample-estimates. We apply this procedure for the data examples in Section 4 with $B \in \{100, 1000\}$.

3. Simulation study

In the following section we evaluate the performance of the `lbbGMM` algorithm on simulated data sets. We compare its performance regarding the estimation of the fixed effects as well as the estimation of the covariance of the random effects to the `bGLMM` algorithm. Furthermore, we evaluate its performance regarding variable selection. We look at various scenarios in which cluster-constant covariates are present and also consider the results of the `glmmPQL` function from the `MASS` package as a type of benchmark performance representing conventional estimation approaches. The `glmmPQL` function iteratively calls on the the `lme` function from the `nlme` package [27].

The simulation settings are predominantly based on the simulation studies that were conducted by [35] as well as [11, 12]. As effect selection alongside the estimation of random components is of high interest in this contribution, we consider varying values for the number of noise variables p as well as for the random effects variance σ_b^2 and look at Poisson and Bernoulli data, as those are the distributions implemented in both the `bGLMM` and `lbbGMM` algorithm. Furthermore, we are mostly focusing on random structures consisting of random intercepts only. This is sensible since random intercept models are the most prevalent in practical applications.

Both `bGLMM` and `lbbGMM` are employing the BIC as the selection criterion across the whole study. For the `bGLMM` algorithm, we set $\nu = 1$ as suggested by [35] to decrease the computational effort without any significant loss of power. We set the maximum number of iterations for both to $l_{max} = 500$.

For the evaluation of the estimation accuracy we look at the mean squared errors (mse) for $\boldsymbol{\beta}$ and $\sigma_{\boldsymbol{b}}$ respectively:

$$mse_{\beta} = \|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|^2, \qquad mse_{\sigma_b} = (\sigma_{\boldsymbol{b}} - \hat{\sigma}_{\boldsymbol{b}})^2.$$

We also evaluate the performance of variable selection for the two boosting algorithms. The false positive (f.p.) rate reports how many of the noise variables were selected as informative. The false negative (f.n.) rate displays how many of the actually informative covariates were kept at zero and discarded as noise. False negatives are of special interest, as the wrongly identified random effects may result in actual informative effects not being selected due to the biased random effects estimates. Since the `glmmPQL` function estimates all covariates, it is not capable of variable selection and consequently disregarded for this part of the evaluation.

All presented results are averaged over 100 independent simulations from each single scenario, i.e. each combination of p and σ_b^2. This is a popular choice in the model-based boosting literature and a good balance between meaningfulness and manageable computational burden. Variability of the results is in general low, as can be traced in [35] or [11] which is expected to hold in the present study as well. However, an additional investigation regarding varying monte carlo errors for different numbers of independent simulation runs is presented in Section 3.5.

While the predictive performance on new data is also of high interest, this evaluation is omitted in the present contribution as the wrongly identified random effects lead to biased effect estimates but not necessarily to wrong model fits. This is the case since the biases of random and fixed effects cancel each other out and only the direct interpretation of the effects tends to be faulty, which can be seen in the data example discussed in the introduction. However, as the algorithm is tuned according to information criteria and in addition overcomes the problematic behaviour by `bGLMM` with respect to the identification of fixed and random effects, it is expected that the method will also perform decent on test data.

3.1. Poisson data

First, we simulate data using a random intercept Poisson model:

$$\eta_{ij} = \sum_{t=1}^{p} x_{ijt}\beta_t + b_i.$$

Here, the link function relating the linear predictor and the distribution parameter is the log link function, so that

$$\mathbb{E}[y_{ij}] = \exp(\eta_{ij}) = \lambda_{ij}, \quad y_{ij} \sim Pois(\lambda_{ij}).$$

We have a balanced design of $i = 1, \dots, 40$ subjects with respective $j = 1, \dots, 10$ observations per subject. For the number of fixed effects we consider the different settings of $p \in \{5, 10, 20, 50\}$. Throughout all settings, the respective fixed effects are $\beta_1 = -4, \beta_2 = -6, \beta_3 = 10, \beta_4 = 5$ and $\beta_t = 0$ for $t = 5, \dots p$. We also construct two of the covariates as cluster-constant, so that for $t = 2, 4$ we have $x_{i1t} = x_{i2t} = \dots = x_{i10t}$. The vectors $\boldsymbol{x}'_{ij} = (x_{ij1}, \dots, x_{ij50})$ follow a Uniform distribution so that $\boldsymbol{x}'_{ij} \sim U[-0.14, 0.14]$. The random intercepts are constructed as $b_i \sim N(0, \sigma_b^2)$. For $\sigma_{\boldsymbol{b}}^2$ we consider the different settings of $\sigma_{\boldsymbol{b}} \in (0.4, 0.8, 1.2)$. Overall, this results in twelve different simulation scenarios for the evaluation of the performance of our three approaches. Per each distinct setting, we simulate and evaluate 100 independent data sets.

The results for the estimation of $\boldsymbol{\beta}$ are summarized in Table 2. One can see that lbbGMM clearly outperforms bGLMM throughout all scenarios with respect to mse_{β}. In comparison to the estimation via glmmPQL the lbbGMM algorithm achieves a mostly similar performance, with a generally superior showing for the higher-dimensional cases, especially for $p = 50$.

Table 2. Results for mse_{β}, mse_{σ_b} and variable selection properties with glmmPQL, bGLMM and lbbGMM approaches on simulated Poisson data; bold values indicate scenarios where lbbGMM outperforms bGLMM.

		glmmPQL		bGLMM				lbbGMM			
σ_b	p	mse_{β}	mse_{σ_b}	mse_{β}	mse_{σ_b}	f.p.	f.n.	mse_{β}	mse_{σ_b}	f.p.	f.n.
0.4	5	9.78	0.006	49.50	0.273	0.55	0.33	**8.67**	**0.003**	**0.16**	**0.02**
0.4	10	7.43	0.006	53.92	0.311	0.15	0.34	**6.98**	**0.004**	0.17	**0.02**
0.4	20	20.71	0.006	53.84	0.244	0.07	0.35	**13.47**	**0.004**	0.15	**0.03**
0.4	50	31.25	0.014	56.28	0.289	0.05	0.38	**12.91**	**0.004**	0.12	**0.03**
0.8	5	23.94	0.014	119.42	0.216	0.66	0.28	**19.55**	**0.013**	0.16	**0.04**
0.8	10	24.46	0.015	55.81	0.214	0.18	0.33	**21.83**	**0.013**	**0.14**	**0.02**
0.8	20	17.42	0.015	55.09	0.198	0.09	0.33	**13.42**	0.015	0.12	**0.03**
0.8	50	24.85	0.015	57.48	0.232	0.07	0.36	**13.19**	**0.012**	0.12	**0.03**
1.2	5	27.36	0.033	62.18	0.146	0.69	0.23	**25.72**	0.033	**0.09**	**0.03**
1.2	10	58.71	0.032	59.28	0.156	0.21	0.24	**41.99**	0.037	**0.19**	**0.03**
1.2	20	37.57	0.066	56.98	0.136	0.11	0.25	**28.79**	**0.044**	0.14	**0.03**
1.2	50	91.79	0.031	111.65	0.119	0.07	0.28	**68.06**	0.041	0.11	**0.03**

Table 2 additionally depicts variable selection properties where lbbGMM consistently shows a false positive rate of around 10–20%. This is a slightly worse showing compared to bGLMM in the higher-dimensional settings, but still overall effective. With respect to false negatives lbbGMM generally achieves rates of 1–3% which is a clear improvement compared to bGLMM. While the false positives still tend to be quite high in lower dimensional settings, this is a known issue for model-based gradient boosting as the greedy nature of the fitting algorithm tends to expose higher selection rates of noise variables in cases with small numbers of overall features. This is discussed broadly in the statistical boosting literature and several efforts have been made in the past years to further reduce the amount of falsely selected noise variables. These approaches vary from probing [15], where the algorithm is stopped as soon as an artificially constructed shadow-variable without any information gets selected, over deselection [33], where selected variables with sufficiently small impact on the overall reduction of predictive risk get selected out of the model again, to subspace boosting [32], which combines several variables in badges and, thus, is able to decrease the false positives rate. Further approaches are also developed, e.g. stability selection [16] or modified selection criteria [28].

Similar results are revealed regarding the estimation of σ_b as also shown in Table 2. Again, we can observe that the lbbGMM algorithm clearly outperforms bGLMM in terms of mse_{σ_b} and generally achieves similar results to glmmPQL.

Figure 2 illustrates the boosting process of the lbbGMM algorithm. It displays the coefficient paths for an example simulated data set where $\sigma_b = 0.8$ and $p = 5$ with one noise variable. One can see that there does not appear to be any systematic difference in the selection process between the coefficients of the cluster-varying and the cluster-constant variables. The coefficient with the biggest absolute value in the simulation setting (V3) is updated first. The coefficient with the second highest absolute value (V2), which belongs to a cluster-constant variable, is updated next. Overall, the coefficient paths are stable and converge. The coefficient belonging to the noise variable stays at zero and does not receive an update.

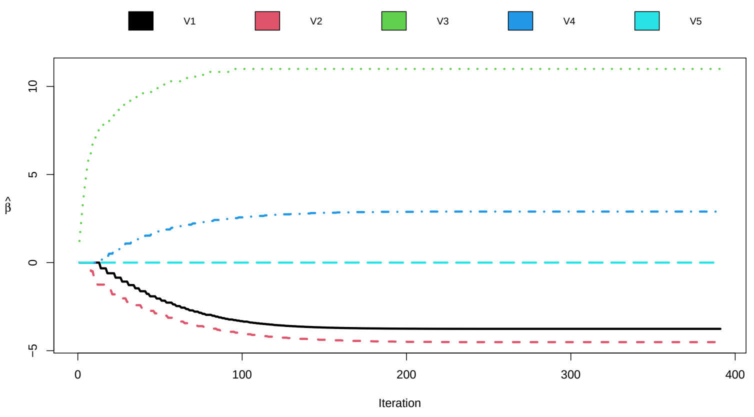


Figure 2. Example for `lbbGMM` Boosting Process on Poisson Data ($\sigma_b = 0.8$).

Figure 3 displays this plot for the same data set when employing the `bGLMM` algorithm. Here, one can observe that the coefficients belonging to the cluster-varying covariates are favored in the updating process with the updates for the coefficients belonging to cluster-constant covariates receiving fewer and weaker updates resulting in them staying close to zero.

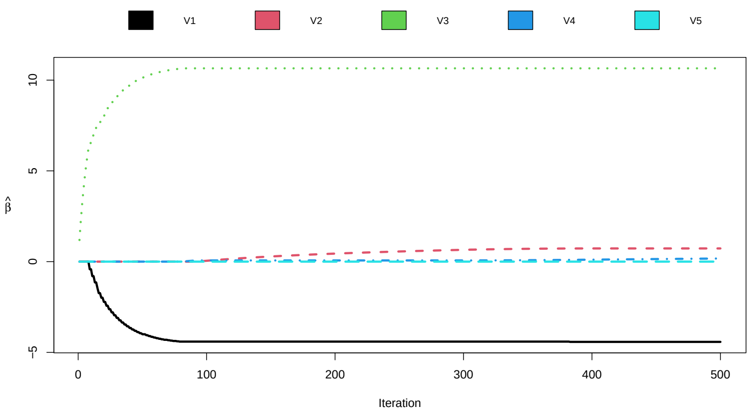


Figure 3. Example for `bGLMM` Boosting Process on Poisson Data ($\sigma_b = 0.8$).

Table 3 shows a comparison between the mse_β for the cluster-constant (V2,V4) and cluster-varying (V1,V3) covariates for all three methods for $\sigma_b = 0.8$ and $p = 5$. One can see that for `bGLMM` the cluster-constant covariates V2 and V4 show a much higher mse_β than the cluster-varying covariates. This difference - while still present - is not as pronounced with the other two methods. Table 3 additionally reveals the apparent cause of the increased mse_β for `bGLMM`. The cluster-constant variables show an extremely high percentage of false negatives as a result of having been neglected in the selection process. Again, one can see that for `lbbGMM` this issue is largely resolved by achieving much lower false negatives rates.

Table 3. mse_{β} and share of false negatives separated for each covariate with Poisson data where $\sigma_{b} = 0.8$ and $p = 5$.

	`glmmPQL`	`bGLMM`		`lbbGMM`	
	mse_{β_k}	mse_{β_k}	f.n.	mse_{β_k}	f.n.
V1	0.26	0.35	0.00	0.53	0.00
V2	17.96	92.87	0.57	13.28	0.08
V3	0.41	0.44	0.00	0.47	0.00
V4	4.97	25.16	0.53	5.16	0.06
V5	0.23	0.17	-	0.02	-

3.2. Bernoulli data

We also simulate data using a random intercept Bernoulli model:

$$\eta_{ij} = \sum_{t=1}^{p} x_{ijt}\beta_t + b_i,$$

where we apply the appropriate link function so that

$$\mathbb{E}[y_{ij}] = \frac{\exp(\eta_{ij})}{1 + \exp(\eta_{ij})} = \pi_{ij}, \quad y_{ij} \sim B(1, \pi_{ij}).$$

We again consider a balanced design of $i = 1, \ldots, 40$ subjects with respective $j = 1, \ldots, 10$ observations per subject. For the number of fixed effects we consider the different settings of $p \in (5, 10, 20, 50)$. The respective fixed effects are $\beta_1 = -5, \beta_2 = -10, \beta_3 = 15, \beta_4 = 8$ and $\beta_t = 0$ for $t = 5, \ldots p$. We again construct two of the covariates as cluster-constant, so that $x_{i1t} = x_{i2t} = \cdots = x_{i10t}$ for $t = 2, 4$. The vectors $\boldsymbol{x}'_{ij} = (x_{ij1}, \ldots, x_{ij50})$ follow a Uniform distribution so that $\boldsymbol{x}'_{ij} \sim U[-0.1, 0.1]$. The random intercepts are again constructed as $b_i \sim N(0, \sigma_{b}^2)$ with the different settings of $\sigma_{b} \in (0.4, 0.8, 1.2)$. We again arrive at twelve different simulation scenarios for the evaluation of the performance of our three approaches with 100 simulated data sets per distinct setting.

The results regarding $\boldsymbol{\beta}$, σ_{b} and selection properties are summarized in Table 4. Please note that in the case of Bernoulli data, the `lbbGMM` algorithm employs the simpler approximation for obtaining the *df* that was also employed by [11] (details in Section 3.3). The overall estimation accuracy has suffered for all three methods in comparison to the previous simulations of Poisson data. This is owed to Bernoulli data containing less information within itself. However, `bGLMM` seems to have deteriorated less and achieves somewhat consistent results in comparison to the Poisson case. Overall, the `lbbGMM` algorithm is still able to outperform `bGLMM` in terms of mse_{β} in the settings where $\sigma_{b} \in (0.4, 0.8)$. Furthermore, `lbbGMM` also again greatly outperforms `glmmPQL` in the cases of higher-dimensional data. For $\sigma_{b} = 1.2$ however, the performance of `lbbGMM` declines, evidently due to a surge in the occurrence of false negatives. Across all settings the boosting methods generally outperform the `glmmPQL` approach in terms of mse_{β}, most notably in the high-dimensional cases.

Table 4. Results for mse_{β}, mse_{σ_b} and variable selection properties with `glmmPQL`, `bGLMM` and `lbbGMM` approaches on simulated binomial data; bold values indicate scenarios where `lbbGMM` outperforms `bGLMM`.

		`glmmPQL`		`bGLMM`				`lbbGMM`			
σ_b	p	mse_{β}	mse_{σ_b}	mse_{β}	mse_{σ_b}	f.p.	f.n.	mse_{β}	mse_{σ_b}	f.p.	f.n.
0.4	5	88.24	0.056	113.40	0.017	0.02	0.17	**46.63**	0.068	0.04	**0.09**
0.4	10	198.91	0.071	113.88	0.017	0.01	0.15	**47.66**	0.069	0.05	**0.10**
0.4	20	162.25	0.077	116.07	0.018	0.01	0.17	**43.83**	0.076	0.05	**0.09**
0.4	50	375.06	0.099	112.61	0.020	0.01	0.16	**74.98**	0.085	0.05	**0.10**
0.8	5	65.26	0.047	117.88	0.048	0.00	0.16	**69.47**	0.340	0.02	**0.13**
0.8	10	138.01	0.037	128.88	0.048	0.01	0.21	**90.95**	0.296	0.02	0.22
0.8	20	146.02	0.060	118.65	0.042	0.01	0.20	**76.57**	0.339	0.04	**0.17**
0.8	50	378.98	0.104	124.42	0.054	0.01	0.18	**89.59**	0.384	0.04	**0.16**
1.2	5	97.11	0.067	129.44	0.195	0.01	0.21	177.88	0.374	**0.01**	0.43
1.2	10	498.83	0.064	134.76	0.196	0.02	0.22	178.86	0.379	**0.01**	0.43
1.2	20	193.21	0.068	126.99	0.212	0.02	0.20	164.66	0.473	**0.01**	0.38
1.2	50	542.87	0.145	136.06	0.211	0.02	0.23	176.76	0.471	**0.02**	0.38

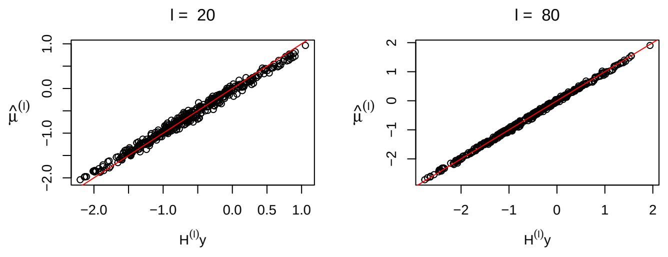
3.3. Contemplation of df approximation

As mentioned, [11] used a simpler method to estimate degrees of freedom for model complexity, counting the number of non-zero $\beta_j \neq 0$ (for $j \leq p$) and variance-covariance parameters. While this method is computationally efficient compared to Hat-matrix approaches, it is less methodologically rigorous. Table 5 shows estimation results on Poisson data using `lbbGMM` with this simpler degrees of freedom approximation. The results for mse_{β}, mse_{σ_b} and false negative rates are comparable to those in Table 2, while the false positive rates are improved.

Table 5. Results for mse_{β} and mse_{σ_b} with `lbbGMM` on simulated Poisson data employing the simple approximation of df.

		`lbbGMM`			
σ_b	p	mse_{β}	f.p.	f.n.	mse_{σ_b}
0.4	5	9.737	0.02	0.023	0.004
0.4	10	6.621	0.03	0.023	0.004
0.4	20	13.927	0.027	0.03	0.004
0.4	50	12.444	0.024	0.03	0.004
0.8	5	20.641	0.020	0.040	0.012
0.8	10	22.242	0.03	0.028	0.013
0.8	20	12.393	0.017	0.035	0.015
0.8	50	12.923	0.025	0.04	0.012
1.2	5	26.562	0.03	0.035	0.033
1.2	10	45.249	0.037	0.045	0.036
1.2	20	29.271	0.031	0.033	0.044
1.2	50	71.933	0.026	0.038	0.04

For the simulation study on Bernoulli data, we chose to employ the simple approximation instead of the approximation via Hat-matrices to arrive at our presented estimation results. This decision was made due to an issue arising when deriving the Hat-matrices on Bernoulli data. As previously outlined, for the selection of the optimal iteration in the boosting process the Hat-matrix of each iteration l is obtained and the following approximation should hold $\hat{\boldsymbol{\mu}}^{(l)} \approx \boldsymbol{H}^{(l)}\boldsymbol{y}$. Figures 4 and 5 show an illustration of this approximation along the boosting process for simulations of Gaussian and Poisson data respectively. The data set at hand is simulated with $\sigma_{\boldsymbol{b}} = 0.4$ and we look at the approximation after iterations $l \in (20, 80, 120, 160)$. The red line denotes the bisector where both sides of the formula are equal. One can see that for these two distribution the approximation holds well across the iterations. Consequently, the employment of the Hat-matrix for obtaining the df performs adequately as well and the estimation results for Poisson data as presented in Table 2 are satisfactory.



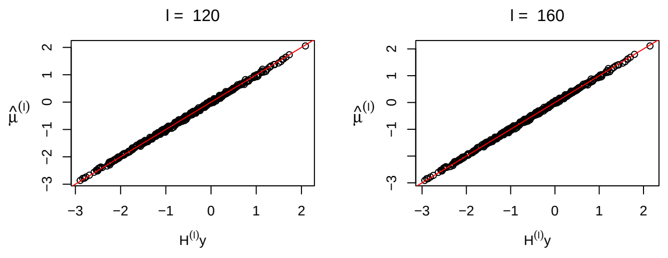


Figure 4. Plots of $\hat{\boldsymbol{\mu}}^{(l)}$ against $\boldsymbol{H}^{(l)}\boldsymbol{y}$ for Gaussian data with $\sigma_{\boldsymbol{b}} = 0.4$ and $l \in (20, 80, 120, 160)$.

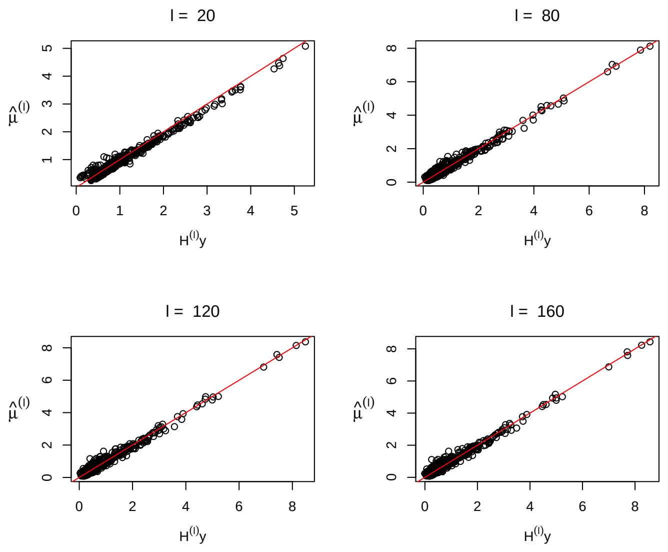


Figure 5. Plots of $\hat{\boldsymbol{\mu}}^{(l)}$ against $\boldsymbol{H}^{(l)}\boldsymbol{y}$ for Poisson data with $\sigma_{\boldsymbol{b}} = 0.4$ and $l \in (20, 80, 120, 160)$.

In contrast, Figure 6 illustrates the approximation accuracy for the simulated Bernoulli data. Here it is apparent that the approximation does not hold as well across all iterations and instead shows some scattering. Consequently, the employment for obtaining the df does not perform well either and is not preferable over using the simple approximation.

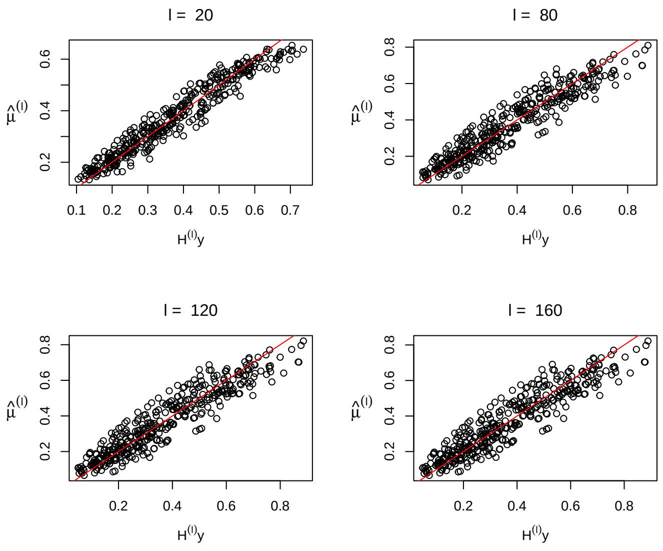


Figure 6. Plots of $\hat{\boldsymbol{\mu}}^{(l)}$ against $\boldsymbol{H}^{(l)}\boldsymbol{y}$ for Bernoulli data with $\sigma_b = 0.4$ and $l \in (20, 80, 120, 160)$.

3.4. *Computational effort*

We showcase the computational advantages which `lbbGMM` holds over `bGLMM` by conducting a simulation study with varying numbers of clusters. The setup mimics Section 3.1 using normal data with $p = 5$, $\sigma_b = 0.8$ and numbers of clusters $n = 20, 40, \ldots, 200$. The progression of averaged computation times (over 10 independent runs) with increasing number of clusters is depicted in Figure 7.

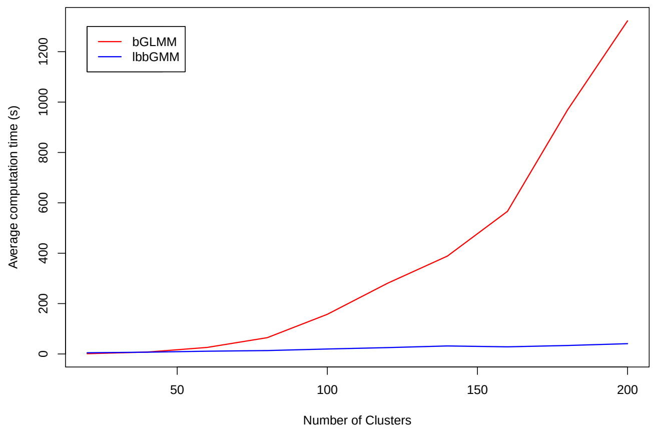


Figure 7. Computation times for `lbbGMM` and `bGLMM` with varying number n of clusters.

While both competitors start fairly equal in cases with a low number of total observations, the computational effort for `bGLMM` quickly adapts to exponential increase as the demanding calculations for model complexity, which `bGLMM` executes multiple times within every single boosting iteration, quickly state a large share of the total computational effort. However, `lbbGMM` clearly profits from the disentangled and less complex calculations for model complexity which is revealed by the simulation, as computation time increases linearly in n.

3.5. Monte Carlo error

To justify the choice of 100 independent repetitions we investigate different amounts of total simulation runs with respect to the variability of the estimation precision. As proposed in [19], we focus on a computationally managable scenario with $p = 5$ covariates overall, similar as in Section 3.4. The `lbbGMM` routine is applied to r independently generated data sets and mse_β computed. Figure 8 depicts the varying interquartile range (IQR) of mse_β for $r = 50, 60, \ldots, 150$ alongside with the Monte Carlo error (MCE), which is calculated as the empirical standard deviation of all r mse_β values weighted by $1/\sqrt{r}$.

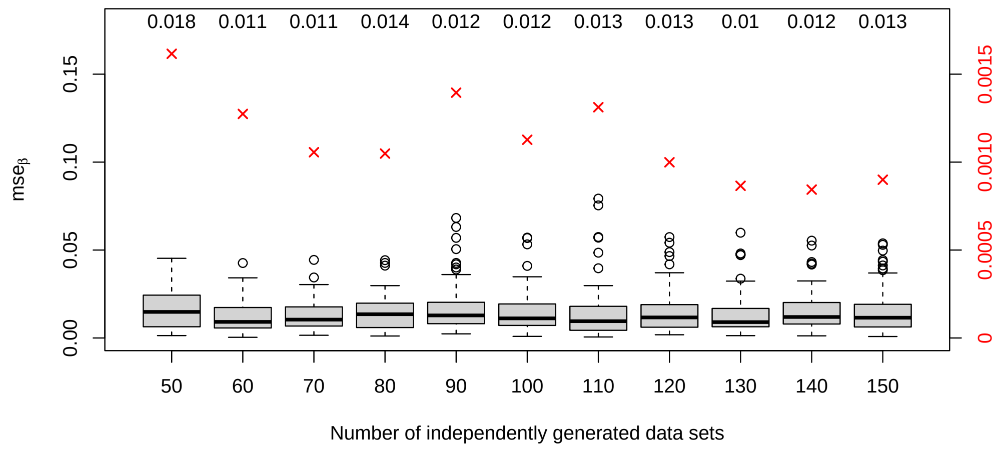


Figure 8. Variability in mse_β across different choices for r. Numbers above boxes show the corresponding IQR, red crosses depict the MCE.

As Figure 8 reveals, no drastic change in overall variability of mse_β estimates can be observed, while the MCE slowly decreases with additional independent simulation runs. However, the steady variability yields no drastic improvement in simulation precision for computationally feasible choices of r. Since, in addition, the simulation study itself conducts 12 different scenarios with $r = 100$ repetitions each, the specific choice of r can be stated as robust.

4. Data applications

We showcase the improvements of the `lbbGMM` routine by analyzing two medical data sets. Both analyses are by no means meant to be conclusive but rather intend to show how the newly designed algorithm manages to overcome the identification issues, which are present in `bGLMM`. Thus, both of the applications include significant and cluster-constant covariates which are severely underestimated by bGLMM, one of them being the AIDS data [1] which already served as motivational example in the beginning.

4.1. CD4 data

The CD4 data set introduced in the beginning is analyzed again, this time employing the lbbGMM algorithm.

We use the BIC as the information criterion and for the learning rates set

$$\nu = \nu_b = 0.1.$$

The maximum number of boosting steps l_{max} has been chosen as 500. As mentioned, we expect to observe a significant estimated effect of the variable *noAIDS/AIDS* on the response variable *Number of CD4 cells*. The estimation results of the lbbGMM algorithm are presented in Table 6. The optimal boosting iteration selected by the BIC was $l_{opt} = 271$. For easy comparison, the table also contains the previous estimation results that were presented in Section 1 for the bGLMM and glmmPQL methods. The bGLMM algorithm is specified with the same settings as lbbGMM where $\nu = 0.1$ and BIC is employed as the criterion and l_{max} is set to 500. For bGLMM the optimal iteration is selected as $l_{opt} = 166$.

One can see that lbbGMM achieves a similar result to glmmPQL and estimates a clear effect for the *noAIDS/AIDS* covariate, one which bGLMM fails to detect. Additionally, Table 6 shows the p-values for the glmmPQL estimation results for the fixed effects. Here, one can see that in terms of the conventional understanding of significance the cluster-constant variable *noAIDS/AIDS* is shown as highly significant as well.

Furthermore, Figure 9 presents the boxplots for the random intercept estimates of subjects with categories AIDS and noAIDS. One can see that in the case of lbbGMM the random intercept estimates do not compensate for the effect of the cluster-constant variable (compare Figure 1). For this application, though it may be trivial in terms of meaningfulness, the lbbGMM algorithm was evidently able to rectify the issue presented by cluster-constant covariates.

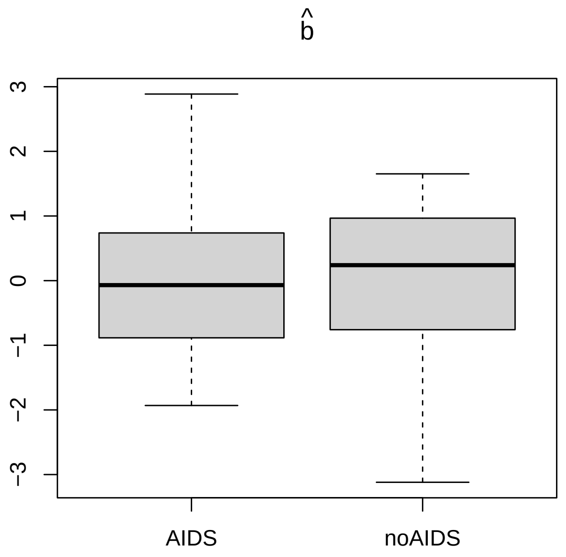


Figure 9. Boxplots of b estimates for noAIDS/AIDS with lbbGMM

Table 6. Estimation results for $\boldsymbol{\beta}$ on Seizure data set.

	glmmPQL (p-value)		bGLMM	lbbGMM (sd)
time	-0.028	($\leq$ 0.01)	-0.029	-0.016 (0.01)
treatment	0.127	(0.23)	0.000	0.111 (0.10)
gender	-0-061	(0.74)	0.000	0.000 (0.19)
noAIDS/AIDS	1.163	($\leq$ 0.01)	0.202	1.244 (0.12)
intolerance	0.094	(0.48)	0.010	0.089 (0.11)

4.2. *Seizure data*

For another application we analyze the Seizure data [21], which contains data from a clinical trial of 59 epileptics. The dependent variable is the number of observed epileptic seizures in four two-weeks intervals. As explanatory variables we include time, indicator of the fourth visit, treatment, age (log), and a baseline number of observed seizures (log). The last three variables are cluster-constant. We again employ the BIC as the information criterion for `lbbGMM` as well as `bGLMM` and the learning rates are set to $\nu = \nu_b = 0.1$. We set $l_{\max} = 500$. Table 7 presents an overview of the estimation results for the fixed effects when employing `glmmPQL`, `bGLMM` and `lbbGMM` methods. The optimal iteration was $l_{opt} = 294$ for `lbbGMM` and $l_{opt} = 26$ for `bGLMM`. In the case of `lbbGMM`, standard errors were again computed based on $B = 1000$ independent resamples according to Section 2.4.2.

Table 7. Estimation results for $\boldsymbol{\beta}$ on Seizure data set.

	glmmPQL (p-value)		bGLMM	lbbGMM (sd)
time	-0.030	(0.51)	-0.005	-0.018 (0.04)
fourth	-0.102	(0.39)	-0.069	-0.096 (0.11)
treat	-0.270	(0.08)	-0.119	-0.291 (0.15)
age	0.378	(0.28)	-0.045	0.177 (0.26)
base	1.023	($\leq$ 0.01)	0.174	0.960 (0.10)

Again, one can observe that `bGLMM` underestimates the cluster-constant covariates, whereas `lbbGMM` delivers estimates that are similar to `glmmPQL`. Especially for the baseline variable, `lbbGMM` is able to detect a strong effect that is much more in line with the `glmmPQL` results. The p-values for the `glmmPQL` estimation reveal the baseline covariate to appear highly significant. Hence, their underestimation by `bGLMM` is especially notable.

5. Discussion

The new `lbbGMM` algorithm has successfully implemented the measures proposed by [11] to deal with cluster-constant covariates in GLMMs. The evaluations based on simulations as well as the CD4 data and Seizure data show that `lbbGMM` is generally able to solve the problem of the `bGLMM` algorithm, where cluster-varying covariates are favored in the boosting process. The changes to the random effects update process, such as the employment of the correction matrix, appear efficient for rectifying the issue of the random effects taking over the boosting updates. However, `lbbGMM` seems to show the most promising results when applied to Poisson data and random structures consisting of only random intercepts for now. In the case of Bernoulli data, the employment of `lbbGMM` seems only situationally advantageous in comparison to `bGLMM`, showing an inferior performance for the settings of $\sigma_b = 1.2$ and for the estimation of σ_b itself. Furthermore, `lbbGMM` overall retains the advantages of boosting methods over the conventional estimation approaches and shows high functionality in terms of variable selection as well as estimation accuracy in high-dimensional data cases.

While the new algorithm is able to extend the usage of the `lbbLMM` algorithm [11] from LMMs to GLMMs by including the new distributions of Poisson and Bernoulli, as of now it is limited to these distributions. Nevertheless, it should be noted, that the `bGLMM` algorithm is also limited to them as they are the most common distributions in practical usage of GLMMs. The implementation of denoting the model complexity via the usage of Hat-matrices has been overall successful, though it might still need some reworking for the application on Bernoulli data. While entailing an increased computational effort, this method for obtaining the degrees of freedom is a more theoretically sound and tested method rather than the simple approximation and should consequently offer a more general and vast application potential. This work mostly focused on random intercepts since it is the most common and practical usage. When including random slopes `lbbGMM` shows a superior performance with regard to the fixed effects estimation in comparison to `bGLMM`. For the estimation of $\boldsymbol{Q}$ some further reworking is still required. Overall, most of the self-imposed limitations on the `lbbGMM` algorithm are sensible and do not restrict its practical use for most real-life applications. In terms of performance, for now `lbbGMM` seems to provide the highest benefit for applications on Poisson data when cluster-constant covariates are present. Some potential extensions arise from the limitations outlined in the previous section. One of which is including other distributions of the exponential family, such as the Gamma distribution. Another one is a further revision of the inclusion of random slopes in order to improve on the present issue of the covariance estimation. This then also opens up the option of additionally implementing a selection process for the random effects. Another aspect that could be looked at for a further expansion of `lbbGMM` is the incorporation of non-linear predictor functions, such as the estimation of smooth effects based on P-splines. This was also outlined by [35] and would offer even more flexibility and potential applications.

Author contributions

Johanna Gerstmeyer: Formal analysis, Methodology, Investigation, Writing-original draft; Elisabeth Bergherr: Funding acquisition, Project administration, Writing-review & editing; Colin Griesbach: Conceptualization, Supervision, Writing-review & editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The work on this article was funded by the DFG (Number 426493614). We acknowledge support by the Open Access Publication Funds of the Göttingen University.

Conflict of interest

The authors declare no conflict of interest.

References

1. D. I. Abrams, A. I. Goldman, C. Launer, J. A. Korvick, J. D. Neaton, L. R. Crane, et al., A comparative trial of didanosine or zalcitabine after treatment with zidovudine in patients with human immunodeficiency virus infection, *New England J. Medic.*, **330** (1994), 657–662. http://doi.org/10.1056/NEJM199403103301001
2. H. Akaike, Information theory and the extension of the maximum likelihood principle, In: *Second International Symposium on Information Theory*, 1973, 267–281.
3. N. E. Breslow, D. G. Clayton, Approximate inference in generalized linear mixed models, *J. Amer. Stat. Assoc.*, **88** (1993), 9–52. https://doi.org/10.1080/01621459.1993.10594284
4. P. Bühlmann, B. Yu, Boosting with the L2 loss: Regression and classification, *J. Amer. Stat. Assoc.*, **98** (2003), 324–339. https://doi.org/10.1198/016214503000125
5. L. Fahrmeir, T. Kneib, S. Lang, B. Marx, *Regression*, Berlin: Springer-Verlag, 2013.
6. L. Fahrmeir, G. Tutz, *Multivariate Statistical Modelling Based on Generalized Linear Models*, New York: Springer-Verlag, 2001.
7. Y. Fang, Asymptotic equivalence between cross-validations and akaike information criteria in mixed-effects models, *J. Data Sci.*, **9** (2011), 15–21.
8. Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm, In: *Proceedings of the Thirteenth International Conference on Machine Learning Theory*, 1996, 148–156.
9. Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, *J. Comput. Syst. Sci.*, **55** (1997), 119–139.
10. J. H. Friedman, Greedy function approximation: A gradient boosting machine, *Ann. Stat.*, **29** (2001), 1189–1232.
11. C. Griesbach, A. Groll, E. Bergherr, Addressing cluster-constant covariates in mixed effects models via likelihood-based boosting techniques, *PLos One*, **16** (2021a), 0254178. https://doi.org/10.1371/journal.pone.0254178
12. C. Griesbach, B. Säfken, E. Waldmann, Gradient boosting for linear mixed models, *Int. J. Biostat.*, **17** (2021b), 317–329. https://doi.org/10.1515/ijb-2020-0136
13. A. Groll, Variable selection by regularization methods for generalized mixed models, PhD thesis, *Ludwig-Maximilians-Universität München, Munich, Germany*, 2011.
14. A. Groll, GMMBoost: Likelihood-based boosting for generalized mixed models, *R package version 1.1.3.*, 2013.
15. T. Hepp, J. Thomas, A. Mayr, B. Bischl, Probing for sparse and fast variable selection with model-based boosting, *Comput. Math. Meth. Medic.*, **2017** (2017), 1421409. https://doi.org/10.1155/2017/1421409

16. B. Hofner, L. Boccuto, M. Göker, Controlling false discoveries in high-dimensional situations: Boosting with stability selection, *BMC Bioinform.*, **16** (2015), 144. https://doi.org/10.1186/s12859-015-0575-3

17. T. Hothorn, P. Buehlmann, T. Kneib, M. Schmid, B. Hofner, Mboost: Model-based boosting, *R package version 2.9-1*, 2018.

18. T. Kneib, T. Hothorn, G. Tutz, Variable selection and model choice in geoadditive regression models, *Biometrics*, **65** (2009), 626–634. https://doi.org/10.1111/j.1541-0420.2008.01112.x

19. E. Koehler, E. Brown, S. Haneuse, On the assessment of monte carlo error in simulation-based statistical analyses, *Amer. Stat.*, **63** (2009), 155–162. https://doi.org/10.1198/tast.2009.0030

20. N. M. Laird, J. H. Ware, Random-effects models for longitudinal data, *Biometrics*, **38** (1982), 963–974.

21. I. E. Leppik, F. E. Dreifuss, T. Bowman, N. Santilli, M. P. Jacobs, C. Crosby, et al., A double-blind crossover evaluation of progabide in partial seizures: 3: 15 PM8, *Neurology*, **35** (1985), 385.

22. X. Lin, N. E. Breslow, Bias correction in generalized linear mixed models with multiple components of dispersion, *J. Amer. Stat. Assoc.*, **91** (1996), 1007–1016.

23. N. T. Longford, A fast scoring algorithm for maximum likelihood estimation in unbalanced mixed models with nested random effects, *Biometrika*, **74** (1987), 817–827. https://doi.org/10.1093/biomet/74.4.817

24. A. Mayr, H. Binder, O. Gefeller, M. Schmid, The evolution of boosting algorithms - from machine learning to statistical modelling, *Meth. Inform. Medic.*, **53** (2014), 419–427.

25. A. Mayr, B. Hofner, M. Schmid, The importance of knowing when to stop. a sequential stopping rule for component-wise gradient boosting, *Meth. Inform. Medic.*, **51** (2012), 178–186. https://doi.org/10.3414/ME11-02-0030

26. P. McCullagh, J. Nelder, *Generalized Linear Models*, London: Chapman and Hall, 1989.

27. J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, R Core Team, nlme: Linear and Nonlinear Mixed Effects Models, *R package version, 3.1-148*, 2020.

28. S. Potts, E. Bergherr, C. Reinke, C. Griesbach, Prediction-based variable selection for component-wise gradient boosting, *Int. J. Biostat.*, **20** (2024), 293–314. https://doi.org/10.1515/ijb-2023-0052

29. R Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2025.

30. G. Schwarz, Estimating the dimension of a model, *Ann. Stat.*, **6** (1978), 461–464.

31. D. J. Spiegelhalter, N. G. Best, B. P. Carlin, A. Van Der Linde, Bayesian measures of model complexity and fit, *J. Royal Stat. Soc. Ser. B*, **64** (2002), 583–639. https://doi.org/10.1111/1467-9868.00353

32. C. Staerk, A. Mayr, Randomized boosting with multivariable base-learners for high-dimensional variable selection and prediction, *BMC Bioinform.*, **22** (2021), 44. https://doi.org/10.1186/s12859-021-04340-z

33. A. Strömer, C. Staerk, N. Klein, L. Weinhold, S. Titze, A. Mayr, Deselection of base-learners for statistical boosting–with an application to distributional regression, *Stat. Meth. Med. Res.*, **31** (2022), 207–224. https://doi.org/10.1177/09622802211051088

34. G. Tutz, H. Binder, Generalized additive models with implicit variable selection by likelihood-based boosting, *Biometrics*, **62** (2006), 961–971. https://doi.org/10.1111/j.1541-0420.2006.00578.x
35. G. Tutz, A. Groll, Generalized linear mixed models based on boosting, In: *Statistical Modelling and Regression Structures*, 2010, 197–216. https://doi.org/10.1007/978-3-7908-2413-1_11
36. G. Tutz, F. Reithinger, A boosting approach to flexible semiparametric mixed models, *Stat. Medic.*, **26** (2007), 2872–2900. https://doi.org/10.1002/sim.2738
37. G. Wahba, A comparison of GCV and GML for choosing the smoothing parameter in the generalized spline smoothing problem, *Ann. Stat.*, **13** (1985), 1378–1402.