Capabilities of R Package mixAK for Clustering Based on Multivariate Continuous and Discrete Longitudinal Data

Arnošt Komárek
Faculty of Mathematics and Physics
Charles University in Prague

Lenka Komárková
Faculty of Management
University of Economics in Prague

Abstract

R package **mixAK** originally implemented routines primarily for Bayesian estimation of finite normal mixture models for possibly interval-censored data. The functionality of the package was considerably enhanced by implementing methods for Bayesian estimation of mixtures of multivariate generalized linear mixed models proposed in Komárek and Komárková (2013). Among other things, this allows for a cluster analysis (classification) based on multivariate continuous and discrete longitudinal data that arise whenever multiple outcomes of a different nature are recorded in a longitudinal study. This package also allows for a data-driven selection of a number of clusters as methods for selecting a number of mixture components were implemented. A model and clustering methodology for multivariate continuous and discrete longitudinal data is overviewed. Further, a step-by-step cluster analysis based jointly on three longitudinal variables of different types (continuous, count, dichotomous) is given, which provides a user manual for using the package for similar problems.

Keywords: cluster analysis, generalized linear mixed model, functional data, multivariate longitudinal data, R package.

This is extended version of a paper published as

> Arnošt Komárek, Lenka Komárková (2014).
> Capabilities of R Package **mixAK** for Clustering Based on Multivariate Continuous and Discrete Longitudinal Data.
> *Journal of Statistical Software*, **59**(12), 1–38.
> URL http://www.jstatsoft.org/v59/i12/.

When citing the material from this document, please, cite the above paper.

This document was built on **September 5, 2014**.

1. Introduction

It is a common practice in longitudinal studies to gather multiple outcomes, both continuous and discrete at each subject's follow-up visit leading to multivariate longitudinal data. In many research areas, the interest then lies in classifying (clustering) the subjects into groups (clusters) on the basis of such multivariate longitudinal data. To be more specific, we first introduce a basic notation. Suppose that $\mathbf{t}_i = \left(t_{i,1}, \ldots, t_{i,n_i}\right)^\top$, $i = 1, \ldots, N$, are the visit times of N subjects involved in a longitudinal study, where n_i is the ith subject number of visits which may differ between subjects, as well as the actual visit times. Further, let $\mathbf{Y}_{i,r} = \left(Y_{i,r,1}, \ldots, Y_{i,r,n_i}\right)^\top = \left(Y_{i,r,1}(t_{i,1}), \ldots, Y_{i,r,n_i}(t_{i,n_i})\right)^\top$, $r = 1, \ldots, R$, be the random vectors representing the ith subject's longitudinal measurements of the rth outcome being recorded. The problem of clustering based on multivariate longitudinal data lies in using complete observational vectors $\mathbf{Y}_i = \left(\mathbf{Y}_{i,1}^\top, \ldots, \mathbf{Y}_{i,R}^\top\right)^\top$ together with $\mathbf{t}_i$, $i = 1, \ldots, N$, and possibly other exogenous covariates for classification of subjects into one of the K groups where K is either known or unknown.

1.1. Model-based clustering

A model-based clustering became a popular method of classification in situations where it is suitable to distinguish the K clusters by different probabilistic models (Bock 1996; Fraley and Raftery 2002). Initially, we assume that K is known. As usual in this context, we introduce the unobservable component allocations $U_1, \ldots, U_N \in \{1, \ldots, K\}$,

$$\mathsf{P}(U_i = k;\, \mathbf{w}) = w_k, \qquad i = 1, \ldots, N,\ k = 1, \ldots, K, \tag{1}$$

where $\mathbf{w} = \left(w_1, \ldots, w_K\right)^\top$ is a vector of unknown cluster proportions which are positive and sum to unity. The meaning of the component allocations is so that $U_i = k$ when the ith subject's observational random vector $\mathbf{Y}_i$ was generated by the kth probabilistic model represented by a model density $f_{i,k}(\mathbf{y}_i;\, \boldsymbol{\xi}_k,\, \boldsymbol{\xi})$, where $\boldsymbol{\xi}_k$ is a vector of cluster-specific and $\boldsymbol{\xi}$ a vector of common unknown parameters. The subscript i in $f_{i,k}$, which is a conditional density of $\mathbf{Y}_i$ given $U_i = k$, points to the fact that $f_{i,k}$ may depend on subject specific factors like the visit times $\mathbf{t}_i$ or other covariates, and allows us to consider also regression models as cluster characteristics. The marginal density of $\mathbf{Y}_i$, and hence also the likelihood contribution of the ith subject, is then a mixture density

$$f_i(\mathbf{y}_i;\, \boldsymbol{\theta}) = \sum_{k=1}^{K} w_k\, f_{i,k}(\mathbf{y}_i;\, \boldsymbol{\xi}_k,\, \boldsymbol{\xi}), \tag{2}$$

where $\boldsymbol{\theta} = \left(\mathbf{w}^\top, \boldsymbol{\xi}_1^\top, \ldots, \boldsymbol{\xi}_K^\top, \boldsymbol{\xi}^\top\right)^\top$ denotes a vector of all unknown model parameters. Model-based clustering is then based on the estimated values $\widehat{p}_{i,k}$, $i = 1, \ldots, N$, $k = 1, \ldots, K$, of the individual component probabilities

$$p_{i,k} = p_{i,k}(\boldsymbol{\theta}) = \mathsf{P}\big(U_i = k \,\big|\, \mathbf{Y}_i = \mathbf{y}_i;\, \boldsymbol{\theta}\big) = \frac{w_k\, f_{i,k}(\mathbf{y}_i;\, \boldsymbol{\xi}_k,\, \boldsymbol{\xi})}{f_i(\mathbf{y}_i;\, \boldsymbol{\theta})}, \qquad i = 1, \ldots, N,\ k = 1, \ldots, K. \tag{3}$$

A possible classification rule whose specific choice depends on a considered loss function from a misclassification, is to assign subject i into the cluster $g(i)$ such that $\widehat{p}_{i,g(i)} = \max_{k=1,\ldots,K} \widehat{p}_{i,k}$.

1.2. R package mixAK

The R (R Core Team 2014) package **mixAK** (Komárek 2014), which is available from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=mixAK, started as a set of routines for Bayesian estimation of finite mixture models and subsequent clustering based on possibly censored data. In the earlier version of the package, which is described in Komárek (2009), only a simple case was covered where the densities $f_{i,k}$ in (2) were all assumed to be multivariate normal with means and covariance matrices depending on k only, i.e., $f_{i,k} = f_k \equiv \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, $\boldsymbol{\xi}_k = \left(\boldsymbol{\mu}_k^\top, \mathsf{vec}^\top(\boldsymbol{\Sigma}_k)\right)^\top$, $k = 1, \ldots, K$.

Following two methodological papers (Komárek, Hansen, Kuiper, van Buuren, and Lesaffre 2010; Komárek and Komárková 2013), the package has been considerably extended as follows. It is assumed that the densities $f_{i,k}$ correspond to multivariate generalized linear mixed models, which leads to a density f_i (Equation 2) being a mixture of multivariate generalized linear mixed models. We describe this concept in more detail in Section 2. These extensions result in two new principal capabilities of the package that are only rarely implemented elsewhere. These are:

(i) joint analysis of multivariate longitudinal data, both continuous and discrete, by the mean of an extension of a well understood generalized linear mixed model;

(ii) cluster analysis based on multivariate longitudinal data.

Even though we shall exemplify the whole methodology on problems from the area of longitudinal studies, all methods as well as the **mixAK** package have much broader area of applications in fact covering data exhibiting all types of repeated measurements. For example, they can be directly applied for the cluster analysis of the functional data, to name one.

In this manuscript, we concentrate on introducing the new capabilities of the package **mixAK** related to the cluster analysis of multivariate longitudinal data in particular. To this end, we first finalize the introduction by a brief overview of other possible R implementations of methods which might be considered for clustering based on longitudinal data. Nevertheless, we also explain that most of these seeming competitors to our package can only be used for clustering longitudinal data under restrictive assumptions not requested by our package. Further, in Section 2, we first overview the particular form of the mixture model (2) that underlies the methods implemented in the package **mixAK** and then discuss how to use the model for clustering and also how to infer a number of mixture components. The main part of the paper is given in Section 3, where we provide a detailed R example. It is a cluster analysis based on a medical dataset involving three longitudinal outcomes of different natures (continuous, count, dichotomous), which served as a motivating example for Komárek and Komárková (2013). This paper finishes with conclusions and an outlook in Section 4.

1.3. Possible competitors

A variety of model-based clustering methods have been implemented and the methods are available as contributed R packages. If the longitudinal data at hand are regularly sampled ($n_1 = \cdots = n_N = n$, $\mathbf{t}_1 = \cdots = \mathbf{t}_N = \mathbf{t}$), it is reasonable to assume that the vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_N$ are not only independent but also identically distributed, i.e., i.i.d. Consequently,

it is possible to consider the clustering methods based on finite mixtures of classical distributions. Besides the earlier version of the **mixAK** package, these are available in the R packages **mclust** (Fraley, Raftery, Murphy, and Scrucca 2012) or **teigen** (Andrews and McNicholas 2013), for instance, where the mixture components are assumed to follow either a multivariate normal or t-distributions. Mixtures with not only continuous but also discrete components are implemented in the R packages **mixdist** (Macdonald and Du 2012) or **Rmixmod** (Auder, Lebret, Iovleff, and Langrognet 2014). Cluster analysis with high dimensional data can also be performed by the means of the R package **HDclassif** (Bergé, Bouveyron, and Girard 2012). Nevertheless, for all of the above-mentioned approaches, the mixture parameters are not allowed in a structural way to depend on exogenous factors like the visit time vectors $\mathbf{t}_1, \ldots, \mathbf{t}_N$, for example. This makes them impractical or inefficient for actual longitudinal data. The possibility to model the dependence of mixture parameters on factors like the visit times $\mathbf{t}_1, \ldots, \mathbf{t}_N$ and use the resulting model for clustering is provided by the R packages **longclust** (McNicholas, Jampani, and Subedi 2012) or **MFDA** (Zhong and Ma 2012). However, only regularly sampled, and only continuous longitudinal data can be clustered using those packages.

In biostatistical applications, as well as in other research areas, the longitudinal data are typically irregularly sampled, i.e., having in general different values of numbers of visits $n_1, \ldots, n_N$ and/or different visit time vectors $\mathbf{t}_1, \ldots, \mathbf{t}_N$. Model-based clustering methods for such data suggested in the literature are then usually based on a mixture of suitable regression models. A mixture of linear mixed models can base the clustering for continuous longitudinal data, the approach being implemented in the R package **mixtools** (Benaglia, Chauveau, Hunter, and Young 2009). Both continuous and discrete longitudinal data can be clustered via mixtures of various types of mixed models using the R package **lcmm** (Proust-Lima, Philipps, Diakite, and Liquet 2013). However, it is not possible to use *jointly* continuous and discrete outcomes in one analysis. This rules out this package for clustering based on general multivariate longitudinal data, where different distributional assumptions, e.g., gaussian for elements of $\mathbf{Y}_{i,1}$, and bernoulli for elements of $\mathbf{Y}_{i,2}$, $i = 1, \ldots, N$, might be unavoidable. According to the best of our knowledge, the only R package allowing for cluster analysis based on actual multivariate longitudinal data of different nature (both continuous and discrete) is **flexmix** (Grün and Leisch 2008). It implements a mixture of generalized linear models, possibly with repeated measurements estimated using likelihood principles by the means of the EM algorithm. Nevertheless, in the case of multivariate responses (i.e., $R > 1$ in our notation), the model used by **flexmix** assumes that these are independent for $r = 1, \ldots, R$ which might be unrealistic.

The mixture of multivariate generalized linear mixed models provides some features, which make the **mixAK** package different and to some extent more broadly applicable compared to the above-mentioned implementations. Briefly, those features are such that they allow for (i) irregularly sampled longitudinal data; (ii) multivariate longitudinal data with responses of different nature (continuous and discrete); and (iii) multivariate responses are not necessarily assumed to be independent for one subject.

2. Model and clustering procedure

Methodology which underlies the procedures for clustering based on multivariate longitudinal data implemented in the package **mixAK** is described in detail in Komárek and Komárková (2013) and its electronic supplement. Nevertheless, to make this paper relatively standalone

and to introduce the necessary notation, we provide a brief overview in this section.

2.1. Multivariate mixture generalized linear mixed model

The mixture model (2) assumed by the package **mixAK** for the observable random vectors $\mathbf{Y}_i = \big(Y_{i,1,1}, \ldots, Y_{i,R,n_i}\big)^\top$, $i = 1, \ldots, N$, as a basis for the clustering procedure is determined by the following assumptions.

(A1) For each $i = 1, \ldots, N$, $r = 1, \ldots, R$ and each $j = 1, \ldots, n_i$, $Y_{i,r,j}$ follows a distribution $\mathcal{D}_r$ from the exponential family with the dispersion parameter ϕ_r (fixed or unknown depending on the considered exponential family) and the mean given by

$$h_r^{-1}\big\{\mathsf{E}(Y_{i,r,j} \mid \mathbf{B}_{i,r} = \mathbf{b}_{i,r};\, \boldsymbol{\alpha}_r)\big\} = \mathbf{x}_{i,r,j}^\top \boldsymbol{\alpha}_r + \mathbf{z}_{i,r,j}^\top \mathbf{b}_{i,r}, \tag{4}$$

where

(i) h_r^{-1} is a chosen link function;

(ii) $\mathbf{x}_{i,r,j}^\top \in \mathbb{R}^{p_r}$, $\mathbf{z}_{i,r,j}^\top \in \mathbb{R}^{q_r}$ are vectors of known covariates (visit times $t_{i,j}$ and possibly other factors) such that the matrix $\big(\mathbb{X}_r\ \mathbb{Z}_r\big)$, where

$$\mathbb{X}_r = \begin{pmatrix} \mathbf{x}_{1,r,1}^\top \\ \vdots \\ \mathbf{x}_{N,r,n_N}^\top \end{pmatrix}, \quad \mathbb{Z}_r = \begin{pmatrix} \mathbf{z}_{1,r,1}^\top \\ \vdots \\ \mathbf{z}_{N,r,n_N}^\top \end{pmatrix},$$

is of a full column rank $p_r + q_r$;

(iii) $\boldsymbol{\alpha}_r \in \mathbb{R}^{p_r}$ is a vector of unknown parameters (fixed effects);

(iv) $\mathbf{B}_{i,r} \in \mathbb{R}^{q_r}$ is a random vector (random effects).

(A2) For each $i = 1, \ldots, N$, the conditional distribution of the joint (over R markers) random effects vector $\mathbf{B}_i = \big(\mathbf{B}_{i,1}^\top, \ldots, \mathbf{B}_{i,R}^\top\big)^\top \in \mathbb{R}^q$, $q = \sum_{r=1}^R q_r$, given $U_i = k$ (given the i subject belongs to the kth group) is a (multivariate) normal with unknown mean $\boldsymbol{\mu}_k \in \mathbb{R}^q$ and unknown $q \times q$ positive definite covariance matrix $\mathbb{D}_k$, $k = 1, \ldots, K$, i.e.,

$$\mathbf{B}_i \,\big|\, U_i = k \sim \mathcal{N}_q\big(\boldsymbol{\mu}_k, \mathbb{D}_k\big), \qquad k = 1, \ldots, K. \tag{5}$$

(A3) For each $i = 1, \ldots, N$, the random variables $Y_{i,1,1}, \ldots, Y_{i,R,n_i}$ are conditionally independent given the random effects vector $\mathbf{B}_i$.

(A4) Random vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_N$ are independent.

(A5) Random effects vectors $\mathbf{B}_1, \ldots, \mathbf{B}_N$ are independent.

In summary, the cluster-specific model parameters $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_K$ are composed of the means and covariance matrices of the conditional distributions of random effects, i.e.,

$$\boldsymbol{\xi}_k = \big(\boldsymbol{\mu}_k^\top, \mathsf{vec}^\top(\mathbb{D}_k)\big)^\top, \quad k = 1, \ldots, K.$$

The model parameters common to all clusters are the fixed effects and the dispersion parameters, i.e.,

$$\boldsymbol{\xi} = \big(\boldsymbol{\alpha}_1^\top, \ldots, \boldsymbol{\alpha}_R^\top, \phi_1, \ldots, \phi_R\big)^\top. \tag{6}$$

The cluster specific model density $f_{i,k}$, $i = 1, \ldots, N$, $k = 1, \ldots, K$, is then

$$f_{i,k}(\mathbf{y}_i;\, \boldsymbol{\xi}_k,\, \boldsymbol{\xi}) = \int_{\mathbb{R}^q} \Big\{ \prod_{r=1}^{R} \prod_{j=1}^{n_i} f_{\mathcal{D}_r}(y_{i,r,j};\, \boldsymbol{\alpha}_r,\, \phi_r,\, \mathbf{b}_{i,r}) \Big\}\, \varphi(\mathbf{b}_i;\, \boldsymbol{\mu}_k,\, \mathbb{D}_k)\, \mathrm{d}\mathbf{b}_i, \tag{7}$$

where $f_{\mathcal{D}_r}$ is the exponential family density following from the assumption (A1). Further, $\varphi(\cdot;\, \boldsymbol{\mu}_k, \mathbb{D}_k)$ is a density of the (multivariate) normal distribution with a mean $\boldsymbol{\mu}_k$ and a covariance matrix $\mathbb{D}_k$, $k = 1, \ldots, K$, following from assumption (A2).

With $R = 1$, the model density $f_{i,k}$ in (7) is the ith subject's likelihood contribution as if the generalized linear mixed model (GLMM) with $\mathcal{N}(\boldsymbol{\mu}_k,\, \mathbb{D}_k)$ distributed random effects is assumed for the observed data. With $R > 1$, a multivariate generalized linear mixed model (MGLMM) is obtained where the dependence among the random vectors $\mathbf{Y}_{i,1}, \ldots, \mathbf{Y}_{i,R}$ representing different markers is induced by non-diagonal covariance matrix $\mathbb{D}_k$ of the random effects vector $\mathbf{B}_i$ in general. Finally, substituting $f_{i,k}$ from (7) into (2) lead to the ith subject's likelihood contribution

$$f_i(\mathbf{y}_i;\, \boldsymbol{\theta}) = \sum_{k=1}^{K} w_k \int_{\mathbb{R}^q} \Big\{ \prod_{r=1}^{R} \prod_{j=1}^{n_i} f_{\mathcal{D}_r}(y_{i,r,j};\, \boldsymbol{\alpha}_r,\, \phi_r,\, \mathbf{b}_{i,r}) \Big\}\, \varphi(\mathbf{b}_i;\, \boldsymbol{\mu}_k,\, \mathbb{D}_k)\, \mathrm{d}\mathbf{b}_i, \tag{8}$$

$$= \int_{\mathbb{R}^q} \Big\{ \prod_{r=1}^{R} \prod_{j=1}^{n_i} f_{\mathcal{D}_r}(y_{i,r,j};\, \boldsymbol{\alpha}_r,\, \phi_r,\, \mathbf{b}_{i,r}) \Big\} \Big\{ \sum_{k=1}^{K} w_k\, \varphi(\mathbf{b}_i;\, \boldsymbol{\mu}_k,\, \mathbb{D}_k) \Big\}\, \mathrm{d}\mathbf{b}_i. \tag{9}$$

It follows from the expression above that the model assumed for the observable random vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_N$ can be interpreted either as a mixture of multivariate generalized linear mixed models with normally distributed random effects (Equation 8), or as a multivariate generalized linear mixed model with a normal mixture in the random effects distribution (Equation 9), where the overall mean and the overall covariance matrix of the random effects $\mathbf{B}_i$, $i = 1, \ldots, N$, are given by

$$\boldsymbol{\beta} = \mathsf{E}\big(\mathbf{B}_i;\, \boldsymbol{\theta}\big) = \sum_{k=1}^{K} w_k \boldsymbol{\mu}_k, \tag{10}$$

$$\mathbf{D} = \mathsf{VAR}\big(\mathbf{B}_i;\, \boldsymbol{\theta}\big) = \sum_{k=1}^{K} w_k \Big\{ \mathbf{D}_k + \big(\boldsymbol{\mu}_k - \sum_{j=1}^{K} w_j \boldsymbol{\mu}_j\big)\big(\boldsymbol{\mu}_k - \sum_{j=1}^{K} w_j \boldsymbol{\mu}_j\big)^{\top} \Big\}. \tag{11}$$

Consequently, we call our model a multivariate mixture generalized linear mixed model (MMGLMM). Following the above considerations, the model likelihood and its observed data deviance are

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{N} f_i(\mathbf{y}_i;\, \boldsymbol{\theta}), \qquad D(\boldsymbol{\theta}) = -2\, \log\big\{L(\boldsymbol{\theta})\big\}, \tag{12}$$

respectively.

Finally, we point out that in the package **mixAK**, the following exponential distributions $\mathcal{D}_r$ and the link functions h_r^{-1} from assumption (A1) are implemented: (a) Gaussian with the identity link, i.e., a linear mixed model for the rth marker where the dispersion parameter ϕ_r is the unknown residual variance; (b) Poisson with the log link where $\phi_r = 1$; (c) Bernoulli with the logit link where again $\phi_r = 1$.

2.2. Bayesian inference

For largely computational reasons, the Bayesian approach based on the output from the Markov chain Monte Carlo (MCMC) simulation is exploited to infer the unknown model parameter vector $\boldsymbol{\theta}$ and to perform the clustering. In a sequel, let $p(\cdot)$ and $p\big(\cdot \,\big|\, \cdot\big)$ be generic symbols for (conditional) distributions and densities. As is usual in Bayesian statistics, the latent quantities (random effects $\mathbf{B}_i$ and component allocations U_i, $i = 1, \ldots, N$) are considered as additional model parameters with the joint prior distribution for all model parameters specified hierarchically following the structure of the model outlined in Sections 1.1 and 2.1 as

$$\begin{aligned} p(\boldsymbol{\theta}, \mathbf{b}_1, \ldots, \mathbf{b}_N, u_1, \ldots, u_N) &= \prod_{i=1}^{N} \Big\{ p\big(\mathbf{b}_i \,\big|\, u_i, \boldsymbol{\theta}\big)\, p\big(u_i \,\big|\, \boldsymbol{\theta}\big) \Big\}\, p(\boldsymbol{\theta}) \\ &= \prod_{i=1}^{N} \Big\{ \varphi(\mathbf{b}_i;\, \boldsymbol{\mu}_{u_i},\, \mathbb{D}_{u_i})\, w_{u_i} \Big\}\, p(\boldsymbol{\theta}). \end{aligned}$$

The prior distribution $p(\boldsymbol{\theta})$ of the primary model parameters is then specified to be weakly informative. Finally, the MCMC methods are used to generate a sample

$$\mathcal{S}_M = \left\{ \left(\boldsymbol{\theta}^{(m)}, \mathbf{b}_1^{(m)}, \ldots, \mathbf{b}_N^{(m)}, u_1^{(m)}, \ldots, u_N^{(m)} \right) : m = 1, \ldots, M \right\} \tag{13}$$

from the joint posterior distribution $p\big(\boldsymbol{\theta}, \mathbf{b}_1, \ldots, \mathbf{b}_N, u_1, \ldots, u_N \big|\, \mathbf{y}\big)$ whose margin $p\big(\boldsymbol{\theta} \big|\, \mathbf{y}\big)$ is the posterior distribution of interest, i.e., $p\big(\boldsymbol{\theta} \big|\, \mathbf{y}\big) \propto L(\boldsymbol{\theta})\, p(\boldsymbol{\theta})$. With respect to the intended clustering, a well-known problem arising from the invariance of the likelihood $L(\boldsymbol{\theta})$ under permutation of the component labels is resolved using the relabeling algorithm of Stephens (2000) adapted for use in a context of our model. Detailed description of the assumed prior distribution and the MCMC algorithm are given in Komárek and Komárková (2013, Section 2.2, Appendices A, B).

2.3. Clustering procedure

It is explained in Section 1.1 that the model-based clustering is based on the estimated values of the individual component probabilities (3). Within the Bayesian framework, the natural estimates of the values $p_{i,k}$, $i = 1, \ldots, N$, $k = 1, \ldots, K$, are their posterior means, MCMC estimates of which are easily obtainable from the generated posterior sample $\mathcal{S}_M$, i.e.,

$$\begin{aligned} \widehat{p}_{i,k} = \mathsf{E}\big\{ p_{i,k}(\boldsymbol{\theta}) \,\big|\, \mathbf{Y} = \mathbf{y} \big\} &= \mathsf{P}\big(U_i = k \,\big|\, \mathbf{Y} = \mathbf{y} \big) \\ &= \int p_{i,k}(\boldsymbol{\theta})\, p\big(\boldsymbol{\theta} \big|\, \mathbf{y}\big)\, d\boldsymbol{\theta} \approx \frac{1}{M} \sum_{m=1}^{M} p_{i,k}\big(\boldsymbol{\theta}^{(m)}\big). \end{aligned} \tag{14}$$

Nevertheless, other characteristics of the posterior distributions of the component probabilities, e.g., the posterior medians, can also be used. On top of that, uncertainty in the classification can be to some extent taken into account by exploring either the full posterior distribution of the component probabilities, or by calculating their credible intervals. We illustrate this in Section 3.7.

2.4. Selecting a number of mixture components

Selecting a number of mixture components, that is, selecting a number of clusters if this is not known in advance coincides with the problem of model selection. In the area of mixture models, models with different numbers of components are usually fitted and then compared by a suitable characteristic of model complexity and model fit. Within the **mixAK** package, two approaches can easily be exploited that we briefly describe in the rest of this section.

Penalized expected deviance

The most commonly used Bayesian characteristic of model complexity and model fit is probably the Deviance Information Criterion (DIC, Spiegelhalter, Best, Carlin, and van der Linde 2002). Nevertheless, as shown by Celeux, Forbes, Robert, and Titterington (2006), its use in mixture models is controversial. An alternative measure, the Penalized Expected Deviance (PED), derived from cross-validation arguments, was suggested by Plummer (2008). It does not suffer from the controversies described by Celeux *et al.* (2006) and can be used even in the context of the mixture models. Also the **mixAK** package exploits the PED for model comparison. The Penalized Expected Deviance whose lower value indicates a better model is defined as

$$\mathsf{PED} = \mathsf{E}\big\{D(\boldsymbol{\theta}) \,\big|\, \mathbf{Y} = \mathbf{y}\big\} + p_{opt}. \tag{15}$$

In Equation (15), $D(\boldsymbol{\theta})$ is the observed data deviance (Equation 12), and p_{opt} is the penalty term called optimism whose value can be estimated by the use of importance sampling and the use of two parallel chains.

Posterior distribution of the deviances

An alternative procedure of the model selection has been suggested by Aitkin, Liu, and Chadwick (2009), later described also in Aitkin (2010, Chapters 7 and 8). They propose basing the model comparison on the full posterior distribution of the deviances. They argue that the model comparison based on one-number criteria (like DIC but also the previously-discussed PED) ignores the uncertainty in the comparison which increases with the increasing number of the model parameters. On the other hand, this uncertainty is taken into account when using the whole posterior distribution of the deviance for the comparison.

Suppose that we want to compare model 1 (with $K = K_1$) and model 2 (with $K = K_2 > K_1$) having the deviances $D_{K_1}(\boldsymbol{\theta})$ and $D_{K_2}(\boldsymbol{\theta})$, respectively. The posterior probability

$$P_{K_2,K_1}(\mathbf{y}) = \mathsf{P}\big\{D_{K_2}(\boldsymbol{\theta}) - D_{K_1}(\boldsymbol{\theta}) < 0 \,\big|\, \mathbf{Y} = \mathbf{y}\big\} \tag{16}$$

now quantifies our certainty on whether model 2 is better than model 1. The penalization for the increasing complexity of the model is included implicitly in this procedure as models with more parameters lead to more diffuse posterior distribution of the deviance and hence decrease in $P_{K_2,K_1}(\mathbf{y})$. Aitkin *et al.* (2009, Section 4) further suggest calculating the posterior probability

$$P^*_{K_2,K_1}(\mathbf{y}) = \mathsf{P}\big\{D_{K_2}(\boldsymbol{\theta}) - D_{K_1}(\boldsymbol{\theta}) < -2\,\log(9) \doteq -4.39 \,\big|\, \mathbf{Y} = \mathbf{y}\big\}. \tag{17}$$

They argue that if this probability is high (0.9 or more), we have quite strong evidence in favor of a model with $K = K_2$ over model with $K = K_1$.

Finally, Aitkin (2010) suggests performing an overall comparison of the posterior distributions of deviances by plotting the posterior cumulative distribution functions (cdf's)

$$F_{D_j}\big(d \,\big|\, \mathbf{Y} = \mathbf{y}\big) = \mathsf{P}\big\{D_j(\boldsymbol{\theta}) \leq d \,\big|\, \mathbf{Y} = \mathbf{y}\big\}, \qquad j = 1, 2. \tag{18}$$

of the deviances. We illustrate this in Section 3.8.

3. Example using the package mixAK

The aim of this section is to provide a detailed step-by-step R analysis of a particular dataset, to highlight its most important features and to show how to extract the most important results. Even though we comment the results shown in this manuscript in most cases as well, it goes beyond the scope of this paper to provide them in full context or to explain their meaning in detail. All this is given in an accompanying methodological paper (Komárek and Komárková 2013) where the same dataset is analyzed. This section can also be considered as a user manual which allows the readers to run their own similar analyses by only a mild modification of the example code.

3.1. Data

Longitudinal data clustering capabilities of the R package **mixAK** will be illustrated on the analysis of a subset of the data from a Mayo Clinic trial on 312 patients with primary biliary cirrhosis (PBC) conducted in 1974–1984 (Dickson, Grambsch, Fleming, Fisher, and Langworthy 1989). The primary data are available at `http://lib.stat.cmu.edu/datasets/pbcseq`. In this paper, only $N = 260$ subjects known to be alive at 910 days of follow-up, and only the longitudinal measurements by this point will be considered. The corresponding data are available as a `data.frame` `PBC910` inside the **mixAK** package. Upon loading the package and the data, we print the columns important for our analysis and few tail rows to describe the data structure.

```
R> library("mixAK")
R> data("PBC910", package = "mixAK")
R> tail(PBC910)[, c(1, 2, 3, 6:9)]
```

```
     id day month lbili platelet spiders jspiders
913 311 187  6.14 0.405      382      NA       NA
914 311 397 13.04 0.642      408       0    0.266
915 312   0  0.00 1.856      200       1    0.886
916 312 206  6.77 1.705      189       0    0.188
917 312 390 12.81 2.001      148       0    0.173
918 312 775 25.46 2.791      138       1    0.736
```

It is a longitudinal dataset with one row per visit. There are 1 to 5 visits per subject (identified by column `id`) performed at time of `day` days (`month` months) of follow-up. At each visit, measurements of three markers ($R = 3$) are recorded: continuous logarithmic bilirubin (`lbili`), discrete platelet count (`platelet`) and dichotomous indication of blood vessel malformations (`spiders`). The column `jspiders` is a jittered version of `spiders` which we shall use for drawing of some descriptive plots.

As it is exemplified on data for subject `id` = 311, the value of some of the markers considered might be missing at some visits. In this case, the value of the dichotomous `spiders` is not available at the visit performed at 187 days of follow-up. Still, the non-missing values of variables `lbili` and `platelet` from the visit at 187 days shall contribute to the likelihood, the estimation and clustering procedure. We take missingness of any of the outcome variables into account by modifying the expressions for the ith subject likelihood contributions (Equations 8 and 9) such that for given $r = 1, \ldots, R$, the product over j takes only the available outcome values into account. Note that this "all available cases" approach is valid as soon as the missingness mechanism can be assumed to be ignorable: data being missing at random or completely at random in the classical taxonomy of Rubin (1976).

In the rest of this section, the random vectors $\mathbf{Y}_{i,1}$, $\mathbf{Y}_{i,2}$, $\mathbf{Y}_{i,3}$, $i = 1, \ldots, N$, intended for the cluster analysis, shall correspond to the values of `lbili`, `platelet`, `spiders`, respectively. The column `month` shall lead to the time vectors $\mathbf{t}_1, \ldots, \mathbf{t}_N$. No other covariates will be used in the underlying model.

3.2. Basic data exploration

Basic exploration of longitudinal data may consist of drawing longitudinal profiles observed, possibly highlighting the profiles of selected subjects. For this purpose, package **mixAK** offers two functions:

- `getProfiles()` which creates a list of `data.frame`s (one `data.frame` per subject) with selected variables;
- `plotProfiles()` which creates a spaghetti graph with observed longitudinal profiles per subject.

As an illustration, we extract the longitudinal profiles of variables `lbili`, `platelet`, `spiders`, `jspiders` and then draw the longitudinal profiles of `lbili` while highlighting the first and the last subject in the dataset (`id` 2 and 312), see the upper panel of Figure 1.

```
R> ip <- getProfiles(t = "month",
+                    y = c("lbili", "platelet", "spiders", "jspiders"),
+                    id = "id", data = PBC910)
R> plotProfiles(ip = ip, data = PBC910, var = "lbili", tvar = "month",
+               main = "Log(bilirubin)", highlight = c(1, length(ip)),
+               xlab = "Time (months)", ylab = "Log(bilirubin)")
```

The remaining panels of Figure 1 are drawn analogously while using arguments `trans = log` (lower left panel showing the longitudinal profiles of the log-transformed response `platelet`) and `lines = FALSE, points = TRUE` (lower right panel showing jittered values of the dichotomous response `spiders`) of the function `plotProfiles()`. The reason for drawing the logarithmic transformation of the variable `platelet` is that a log link will be proposed in Section 3.3 for this response and the created plot then better helps to choose a mean structure of the related GLMM. Further, with a dichotomous longitudinal response, it is almost impossible to draw a fully informative plot of observed data and according to the best of our knowledge, no standard way of doing that in the literature exists. The lower right-hand panel of Figure 1 with vertically jittered values of the response `spiders` is then a possible option

which at least allows us into some extent to compare visually the proportions of zeros and ones at each occasion. An alternative could be a suitably adapted spine plot but we do not pursue this option here.

The full code to draw Figure 1 is the following.

```
R> iShow <- c(1, length(ip))     ### indeces of subjects to be highlighted
R> layout(autolayout(3))
R> plotProfiles(ip = ip, data = PBC910, var = "lbili", tvar = "month",
+          auto.layout = FALSE, main = "Log(bilirubin)",
+          xlab = "Time (months)", ylab = "Log(bilirubin)",
```

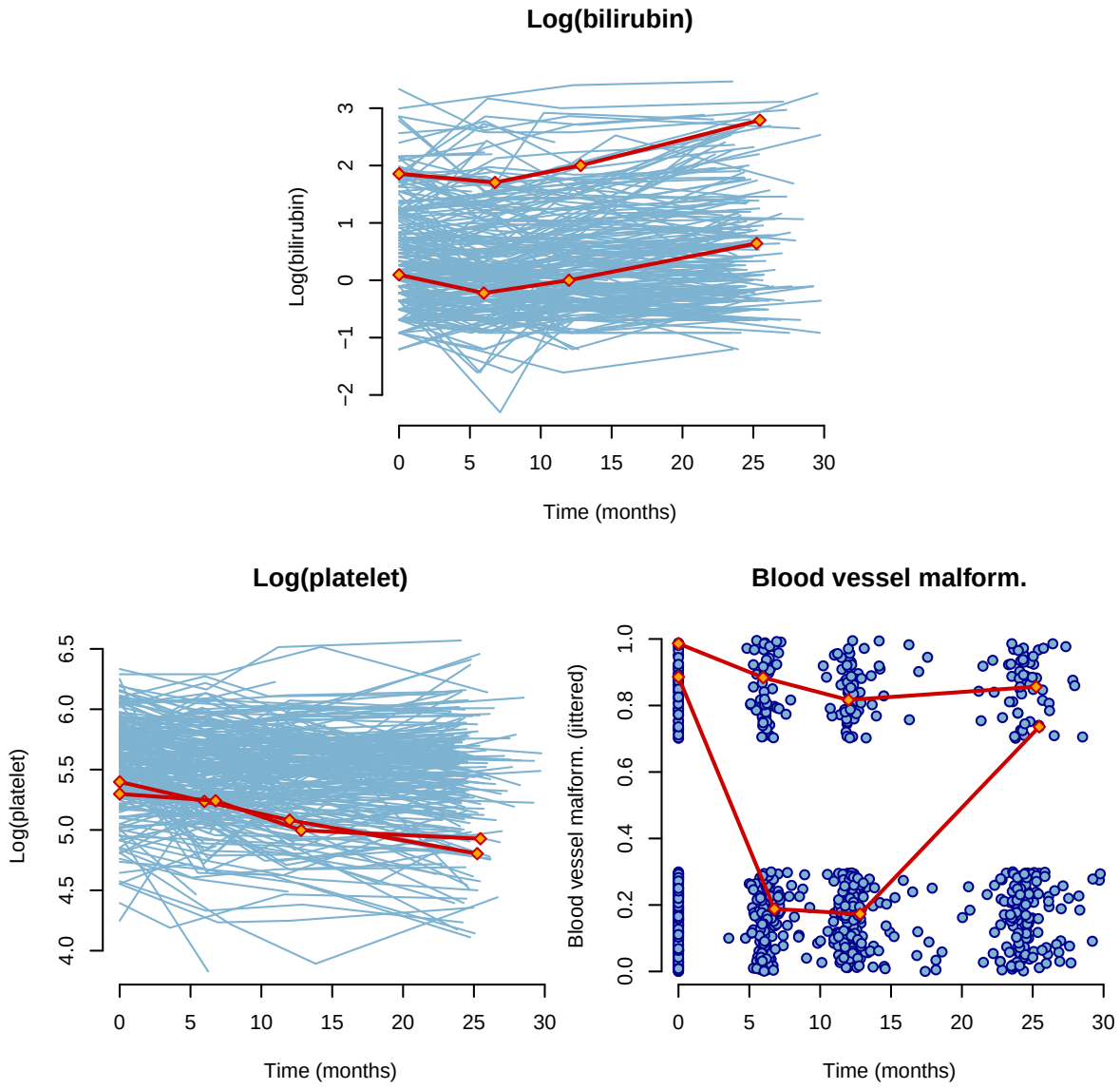

Figure 1: Observed (transformed) longitudinal profiles of considered markers, red lines: profiles of two selected subjects (`id` 2 and 312).

```
+         highlight = iShow)
R> plotProfiles(ip = ip, data = PBC910, var = "platelet", tvar = "month",
+         auto.layout = FALSE, main = "Log(platelet)",
+         trans = log, xlab = "Time (months)", ylab = "Log(platelet)",
+         highlight = iShow)
R> plotProfiles(ip = ip, data = PBC910, var = "jspiders", tvar = "month",
+         lines = FALSE, points = TRUE,
+         auto.layout = FALSE, main = "Blood vessel malform.",
+         xlab = "Time (months)", ylab = "Blood vessel malform. (jittered)",
+         highlight = iShow)
```

3.3. Model

Being partially motivated by Figure 1, by the nature of the considered longitudinal markers and also by other considerations, the following generalized linear mixed models in assumption (A1) of the MMGLMM shall be exploited for the analysis:

(i) the continuous nature of the variable `lbili` suggests considering a Gaussian GLMM, i.e., a linear mixed model, for the random vectors $\mathbf{Y}_{i,1}$, $i = 1, \ldots, N$. Further, Figure 1 together with additional exploration of individual longitudinal profiles (not shown) suggest modeling the evolution of each subject by a line over time where the parameters of the line may differ across subjects. This leads to the following random intercept and random slope model for the means of the elements of the response vectors:

$$\mathsf{E}\big(Y_{i,1,j} \,\big|\, \mathbf{B}_{i,1} = \mathbf{b}_{i,1}\big) = b_{i,1,1} + b_{i,1,2} t_{i,j}, \qquad \mathbf{b}_{i,1} = \big(b_{i,1,1},\, b_{i,1,2}\big)^\top,$$

$i = 1, \ldots, N$, $j = 1, \ldots, n_i$. Additional refinement of the model, for instance towards capturing possibly non-linear evolution of the outcome over time, is of course possible. Nevertheless, this goes beyond the scope of this software related paper.

In a general notation of Section 2.1, there are no fixed effects, and the random effects covariate vector $\mathbf{z}_{i,1,j}$ equals $\big(1,\, t_{i,j}\big)^\top$. The dispersion parameter ϕ_1 is an unknown residual variance of the underlying linear mixed model;

(ii) the count nature of the variable `platelet` leads us to consider a Poisson GLMM for the random vectors $\mathbf{Y}_{i,2}$, $i = 1, \ldots, N$. Analogously to the previous case of the variable `lbili`, exploration of individual longitudinal profiles (Figure 1 and others) suggests considering subject specific lines over time for logarithmically linked means of the elements of the response vectors:

$$\log\Big\{\mathsf{E}\big(Y_{i,2,j} \,\big|\, \mathbf{B}_{i,2} = \mathbf{b}_{i,2}\big)\Big\} = b_{i,2,1} + b_{i,2,2} t_{i,j}, \qquad \mathbf{b}_{i,2} = \big(b_{i,2,1},\, b_{i,2,2}\big)^\top,$$

$i = 1, \ldots, N$, $j = 1, \ldots, n_i$. Analogously to the model for the variable `lbili`, there are no fixed effects, and the random effects covariate vector $\mathbf{z}_{i,2,j}$ equals $\big(1,\, t_{i,j}\big)^\top$. The dispersion parameter ϕ_2 is now constantly equal to 1;

(iii) the dichotomous nature of the variable `spiders` dictates to use a Bernoulli GLMM for which the logit link is a classical choice. The assumed mean structure of the model is the following:

$$\mathrm{logit}\Big\{\mathsf{E}\big(Y_{i,3,j} \,\big|\, B_{i,3} = b_{i,3};\, \alpha_3\big)\Big\} = b_{i,3} + \alpha_3\, t_{i,j}, \tag{19}$$

$i = 1, \ldots, N$, $j = 1, \ldots, n_i$. In this case, the fixed effects covariate vector $\mathbf{x}_{i,3,j} = \big(t_{i,j}\big)$, and the random effects covariate vector $\mathbf{z}_{i,3,j} = \big(1\big)$.

Inclusion of the random intercept in the model above is motivated by observing that subjects differ in predisposition towards development of the blood vessel malformations. Due to the fact that we cannot rule out change of this predisposition over time, an additional linear term is included in the model. Nevertheless, due to the relatively low number of repeated measurement per subject, it is difficult to effectively estimate a model with a dichotomous outcome allowing also for subject-specific slopes, which are then included only as a fixed effect.

In summary, the model parameters (6) common to all clusters are the slope α_3 from the logit model for the variable `spiders`, and the dispersion parameter ϕ_1 from the linear mixed model for the variable `lbili`, i.e., $\boldsymbol{\xi} = \big(\alpha_3,\, \phi_1\big)^\top$. The joint random effects vectors $\mathbf{B}_1, \ldots, \mathbf{B}_N$ are five-dimensional, $\mathbf{B}_i = \big(B_{i,1,1},\, B_{i,1,2},\, B_{i,2,1},\, B_{i,2,2},\, B_{i,3}\big)^\top$, $i = 1, \ldots, N$, with the overall mean $\boldsymbol{\beta} = \big(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5\big)^\top$, and the overall covariance matrix $\mathbf{D} = \big(d_{l,m}\big)_{l,m=1\ldots,5}$. Initially, a model with $K = 2$ mixture components will be fitted.

3.4. Posterior Markov chain Monte Carlo simulation

Two functions from the **mixAK** package are related to the generation of a posterior sample (13) needed for the inference and the clustering. They include:

- `GLMM_MCMC()` is the main function which runs the MCMC algorithm. Internally, most of the calculation is provided by a compiled C++ code to fasten the computational time. With default values of the input arguments, the function generates two MCMC samples started from two sets of initial values which is primarily needed to calculate the penalized expected deviance (PED) useful for model comparison as explained in Section 2.4. The function returns a `list` class of which is set to `GLMM_MCMClist`. It contains some information concerning the model and two objects of class `GLMM_MCMC` holding the two sampled chains, initial values used to start the MCMC, specific choices of hyperparameters of the prior distribution and basic posterior summary statistics. Several methods which we shall introduce subsequently are available for objects classes `GLMM_MCMClist` and `GLMM_MCMC` to handle and visualize the results;
- `NMixRelabel()` is a generic function with methods for objects of classes `GLMM_MCMClist` and `GLMM_MCMC` which applies the relabeling algorithm and returns appropriately modified input object. Analogously to the `GLMM_MCMC()` function, the main calculation is internally conducted using the compiled C++ code.

As an illustration, we run the MCMC algorithm for (100 × 10) burn-in and (1 000 × 10) subsequent iterations with 1:10 thinning to get two samples of length $M = 1\,000$ from the joint posterior distribution obtained by starting the MCMC from two different sets of initial values. Note that a longer MCMC is usually needed to get reliable results but we keep it shorter here to make it quickly reproducible. By default, the two chains are generated sequentially. Nevertheless, on multicore processors this task might be parallelized using the tools provided by the standard R package **parallel** by setting the `parallel` argument to `TRUE`. Indicated computational time was achieved on Intel Core i7 2.7 GHz CPU with 2 × 8 GB RAM running on a Linux Debian OS.

```
R> set.seed(20042007)
R> mod <- GLMM_MCMC(y = PBC910[, c("lbili", "platelet", "spiders")],
+      dist = c("gaussian", "poisson(log)", "binomial(logit)"),
+      id = PBC910[, "id"],
+      x = list(lbili    = "empty",
+               platelet = "empty",
+               spiders  = PBC910[, "month"]),
+      z = list(lbili    = PBC910[, "month"],
+               platelet = PBC910[, "month"],
+               spiders  = "empty"),
+      random.intercept = rep(TRUE, 3),
+      prior.b = list(Kmax = 2),
+      nMCMC = c(burn = 100, keep = 1000, thin = 10, info = 100),
+      parallel = FALSE)

Chain number 1
==============
MCMC sampling started on Fri Sep  5 13:29:32 2014.
Burn-in iteration 100
Iteration 1100
MCMC sampling finished on Fri Sep  5 13:30:00 2014.

Chain number 2
==============
MCMC sampling started on Fri Sep  5 13:30:00 2014.
Burn-in iteration 100
Iteration 1100
MCMC sampling finished on Fri Sep  5 13:30:28 2014.

Computation of penalized expected deviance started
  on Fri Sep  5 13:30:28 2014.
Computation of penalized expected deviance finished
  on Fri Sep  5 13:31:08 2014.
```

The meaning of the most important arguments of the `GLMM_MCMC` function is briefly the following. The argument `y` is a `data.frame` with the observed values of the longitudinal markers in its columns. The mean structure of the GLMM's from assumption (A1) is indicated by arguments `x` (a `list` of vectors/matrices/data frames of the fixed effects covariates except the intercept term), `z` (a `list` of vectors/matrices/data frames of the random effects covariates except the intercept term) and a logical argument `random.intercept`. Note that we assume a hierarchically centered GLMM (Gelfand, Sahu, and Carlin 1995) where the random effects have in general non-zero mean (their overall mean $\boldsymbol{\beta}$ is given by Equation 10). Hence, the marker specific parts of the `lists` in `x` and `z` arguments may not contain the same variables. Otherwise, the model parameters would become unidentifiable as $\boldsymbol{\beta}$ would be estimated twice, once through the mixture means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and once through the corresponding fixed effects. The keyword `"empty"` is used to indicate that there are no fixed or random effects, respectively in a model for a specific marker. Purely to simplify the programming work, the `GLMM_MCMC`

function is coded such that the intercept is always included in the model and the fact whether it is random or fixed is indicated by argument `random.intercept`. Assumed distributions and the link functions for each marker are indicated by the argument `dist`.

The only obligatory part of the prior distribution which has to be specified by the user is the number of mixture components (clusters) given as a `Kmax` element of the `list` in the `prior.b` argument. All other values of prior hyperparameters are selected automatically to achieve a weakly informative prior distribution using the guidelines described in Komárek and Komárková (2013, Appendix A). The function `GLMM_MCMC` also automatically generates the reasonable initial values to start the MCMC simulation. User-defined prior hyperparameters and initial values can be supplied by using the appropriate values of the arguments `prior.alpha`, `prior.b`, `prior.eps`, and `init.alpha`, `init2.alpha`, `init.b`, `init2.b`, `init.eps`, `init2.eps`, respectively.

Before we proceed, we point out that the object `mod` returned by the function `GLMM_MCMC` is a `list` with two main elements `mod[[1]]` and `mod[[2]]` holding the two sampled chains, their initial values, selected posterior summary statistics based on these chains and some additional quantities derived from both sampled chains. The `class` of the `mod` object is set to `GLMM_MCMClist`, class of the main elements `mod[[1]]` and `mod[[2]]` is `GLMM_MCMC`. Basic information concerning the structure of the object `mod` will be given in the subsequent sections. More details and a description of how to change the default values of the prior hyperparameters and the initial values to start the MCMC simulation are given in Appendix A.

As it is mentioned in Section 2.2, a problem arising from the invariance of the model likelihood under permutation of the component labels must be resolved and one possibility is to apply a suitable relabeling algorithm. When running the `GLMM_MCMC` function, only a simple relabeling based on the ordering of the first margins of the mixture means was applied. The resulting relabeling is primarily reflected in the elements `order_b` and `rank_b` which are also present in the `mod[[1]]` and `mod[[2]]` objects.

```
R> mod[[1]]$order_b[1:3,]
     order1 order2
[1,]      1      2
[2,]      1      2
[3,]      1      2
```

```
R> mod[[1]]$rank_b[1:3,]
     rank1 rank2
[1,]     1     2
[2,]     1     2
[3,]     1     2
```

Both `order_b` and `rank_b` are $M \times K$ matrices having the following meaning. Let $o_{m,k}$, be the elements of the matrix `order_b`, and $r_{m,k}$ $m = 1, \ldots, M$, $k = 1, \ldots, K$, the elements of the matrix `rank_b`. After relabeling, component number κ, $\kappa \in \{1, \ldots, K\}$ at iteration m is given by the $o_{m,\kappa}$th component in the original sample, or vice versa, component number ι in the original sample equals to the $r_{m,\iota}$th component in the relabeled sample. Nevertheless, as illustrated in Stephens (2000, Section 3.1), the relabeling applied above is based in fact on imposing an artificial identifiability constraint and is not always satisfactory. This is especially in situations when the chosen identifiability constraint does not separate the mixture components well. Due to this reason, Stephens (2000) suggested a relabeling algorithm which arise from attempting to minimize the posterior expected loss under a class of loss functions suitable for assessment of a clustering procedure. To apply his algorithm on our posterior samples, we run the `NMixRelabel` function (it is done separately for each chain) with its `type` argument set to `"stephens"`:

```
R> mod <- NMixRelabel(mod, type = "stephens", keep.comp.prob = TRUE)

Re-labelling chain number 1
===========================
MCMC Iteration (simple re-labelling) 1000
Stephens' re-labelling iteration (number of labelling changes): 1 (0)

Re-labelling chain number 2
===========================

MCMC Iteration (simple re-labelling) 1000
Stephens' re-labelling iteration (number of labelling changes): 1 (0)
```

Objects `mod[[1]]` and `mod[[2]]` have the same structure as before with the exception that all results which are not invariant towards label switching were (re-)calculated to reflect the new labeling of the mixture components. These are the elements:

- `order_b`,
- `rank_b`,

and also not yet mentioned elements:

- `poster.mean.w_b`,
- `poster.mean.mu_b`,
- `poster.mean.Sigma_b`,
- `poster.mean.Q_b`,
- `poster.mean.Li_b`,
- `poster.comp.prob`,
- `poster.comp.prob_u`,
- `poster.comp.prob_b`,
- `comp.prob`,
- `comp.prob_b`,
- `quant.comp.prob`,
- `quant.comp.prob_b`.

Finally, by setting `keep.comp.prob = TRUE` in the `NMixRelabel` call, we additionally generated the posterior samples $p_{i,k}\left(\boldsymbol{\theta}^{(m)}\right)$, $i = 1, \ldots, N$, $k = 1, \ldots, K$, $m = 1, \ldots, M$ (Equation 14), that subject i belongs to the kth group (where k refers to a new labeling of the components) which will be used in Section 3.7 for the purposes of clustering.

3.5. Posterior samples and basic convergence diagnostics

Posterior samples of the primary model parameters, random hyperparameters and some additional derived quantities are kept in the `mod[[1]]` and `mod[[2]]` objects as their matrix or vector elements. The most important ones include: `Deviance` (sampled model deviance

$L(\boldsymbol{\theta})$, Equation 12), `alpha` (sampled fixed effects $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_R$), `sigma_eps` (sampled square roots of dispersion parameters $\phi_1, \ldots, \phi_R$), `mixture_b` (sampled overall random effects means $\boldsymbol{\beta}$, Equation 10, and standard deviations and correlations derived from the overall random effects covariance matrix $\mathbf{D}$, Equation 11). As illustration, we print the first ten sampled values of the model deviance $L(\boldsymbol{\theta})$ from the first chain:

```
R> print(mod[[1]]$Deviance[1:10], digits = 9)

 [1] 14086.4438 14095.8671 14113.7103 14097.6683 14109.1802 14102.2894
 [7] 14110.1587 14098.9003 14109.3774 14096.8390
```

Classical tools for convergence diagnostics as implemented, e.g., in the R package **coda** (Plummer, Best, Cowles, and Vines 2006) can be used to evaluate the convergence of the performed MCMC simulation. As an illustration, we use the **coda** package routine `autocorr()` to calculate estimated autocorrelations (in the first and the second chain) in our Markov chain of the model deviances (the output was edited into two columns):

```
R> library("coda")
R> DevChains <- mcmc.list(mcmc(mod[[1]]$Deviance), mcmc(mod[[2]]$Deviance))
R> autocorr(DevChains)

[[1]]                    [[2]]
, , 1                    , , 1

Lag 0  1.00000          Lag 0   1.00000
Lag 1  0.24155          Lag 1   0.21846
Lag 5  0.10663          Lag 5   0.12404
Lag 10 0.06345          Lag 10  0.03833
Lag 50 0.00043          Lag 50 -0.01368
```

Two extra routines are available within the **mixAK** package to access the posterior samples and produce basic diagnostics plots. These are:

- `NMixChainComp()` is a generic function with a method for objects of class `GLMM_MCMC` which extracts the posterior samples of the mixture parameters from the resulting object: weights $\mathbf{w}$, means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$ or their derivatives (standard deviations and correlations based on $\mathbf{D}_1, \ldots, \mathbf{D}_K$). It has a logical argument `relabel` that determines whether the original sample is required (`relabel = FALSE`) or a relabeled sample (with default `relabel = TRUE`) as calculated by the `NMixRelabel` function in Section 3.4;
- `tracePlots()` is again a generic function with methods for objects of classes `GLMM_MCMC` and `GLMM_MCMClist` which produces traceplots (parallel if applied to the object of class `GLMM_MCMClist`) of selected class of model parameters. In case that traceplots of mixture related parameters are required, a logical argument `relabel` with the same meaning as in the case of the `NMixChainComp()` function determines whether the original or the relabeled sample is to be plotted.

We first illustrate the use of the `NMixChainComp()` function and extract from the object `mod[[1]]` (the first sampled chain) the relabeled sample of the mixture means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and then print the first three sampled values.

```
R> muSamp1 <- NMixChainComp(mod[[1]], relabel = TRUE, param = "mu_b")
R> print(muSamp1[1:3,])

      mu1.1   mu1.2 mu1.3    mu1.4 mu1.5 mu2.1   mu2.2 mu2.3    mu2.4
[1,] -0.258 0.00645  5.59 -0.00690 -3.82 1.240 0.00322  5.43 -0.00729
[2,] -0.257 0.00345  5.59 -0.00534 -4.41 0.951 0.01426  5.39 -0.00277
[3,] -0.323 0.00625  5.61 -0.00583 -3.83 0.838 0.01144  5.41 -0.00315
      mu2.5
[1,] -1.101
[2,] -1.069
[3,] -0.563
```

The first five columns of a matrix `muSamp1` contain the sampled values of $\boldsymbol{\mu}_1$, the remaining five columns contain the sampled values of $\boldsymbol{\mu}_2$. By changing the value of the argument `param`, sampled values of other mixture parameters are provided. In particular, `param` values of `"w_b"`, `"var_b"`, `"sd_b"`, `"cor_b"`, provide the sampled values mixture weights, variances (diagonal elements of matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$), their square roots (standard deviations), and correlation coefficients derived from matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$, respectively. Further, the `param` values of `"Sigma_b"`, `"Q_b"`, `"Li_b"` provide the sampled values of the lower triangles of the mixture covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$, their inversions, and Cholesky factors of their inversions, respectively. More detailed description of the storage of the posterior samples is given in Appendix B.

Traceplots are a useful basic tool for exploring the behavior of the sampled Markov chains. When two parallel chains are generated, it is useful to draw both chains in two different colors into one plot as it is done by the **mixAK** function `tracePlots`. As illustration, we draw the traceplots of the two parallel chains of the model deviance $D(\boldsymbol{\theta})$, see Figure 2.

```
R> tracePlots(mod, param = "Deviance")
```

By changing the argument `param`, traceplots of other model parameters are drawn. In particular, by setting the argument `param` to `"alpha"` and `"sigma_eps"`, respectively, traceplots of the fixed effects vector $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_R$ and the square roots of the dispersion parameters $\phi_1, \ldots, \phi_R$, respectively, are drawn. Likewise, the `param` argument values of `"Eb"`, `"SDb"` and `"Corb"`, respectively, lead to traceplots for the overall means $\boldsymbol{\beta}$ (Equation 10) of the random effects, and the standard deviations and correlation coefficients, respectively, derived from the overall covariance matrix $\mathbf{D}$ (Equation 11).

Additionally, traceplots of sampled (as such before applying any relabeling algorithm) mixture weights $w_1, \ldots, w_K$, components of the mixture means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and standard deviations from the mixture covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$, respectively, can be drawn by setting the `param` argument of the `tracePlots` function to `"w_b"`, `"mu_b"`, and `"sd_b"`, respectively. Traceplots of these quantities after current relabeling (reflected by the `order_b` and `rank_b` components of the objects `mod[[1]]` and `mod[[2]]`, respectively) are drawn by setting the optional `tracePlots` argument `relabel` to `TRUE` (output not shown).

```
R> tracePlots(mod, param = "w_b",  relabel = TRUE)
R> tracePlots(mod, param = "mu_b", relabel = TRUE)
R> tracePlots(mod, param = "sd_b", relabel = TRUE)
```

Finally, we can draw the traceplots of the hyperparameters $\gamma_{b,1}, \ldots, \gamma_{b,5}$, and $\gamma_{\phi,1}$, respectively, if we set the `tracePlots` argument `param` to `"gammaInv_b"` and `"gammaInv_eps"`, respectively. Finally, when the first argument of the `tracePlots` function is changed to `mod[[1]]` or `mod[[2]]`, respectively, only the single traceplots of the first and the second chain, respectively, are drawn.

3.6. Posterior summary statistics

To summarize the estimated models, we usually calculate the posterior summary statistics and credible intervals for important model parameters. In the context of the MMGLMM, these include: the fixed effects $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_R$ and the dispersion parameters $\phi_1, \ldots, \phi_R$, and the overall mean $\boldsymbol{\beta}$ (Equation 10) and the overall covariance matrix $\mathbf{D}$ (Equation 11) of the random effects. Furthermore, the posterior summary statistics of the observed data deviance $D(\boldsymbol{\theta})$ (Equation 12) is usually reported to provide a basic model fit characteristic. Note that up to now mentioned quantities are invariant towards label switching and hence their posterior summary statistics might be calculated even without performing any relabeling. Having the subsequent clustering in mind, posterior summary statistics for the mixture parameters (weights $w_1, \ldots, w_K$, means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$) are of additional

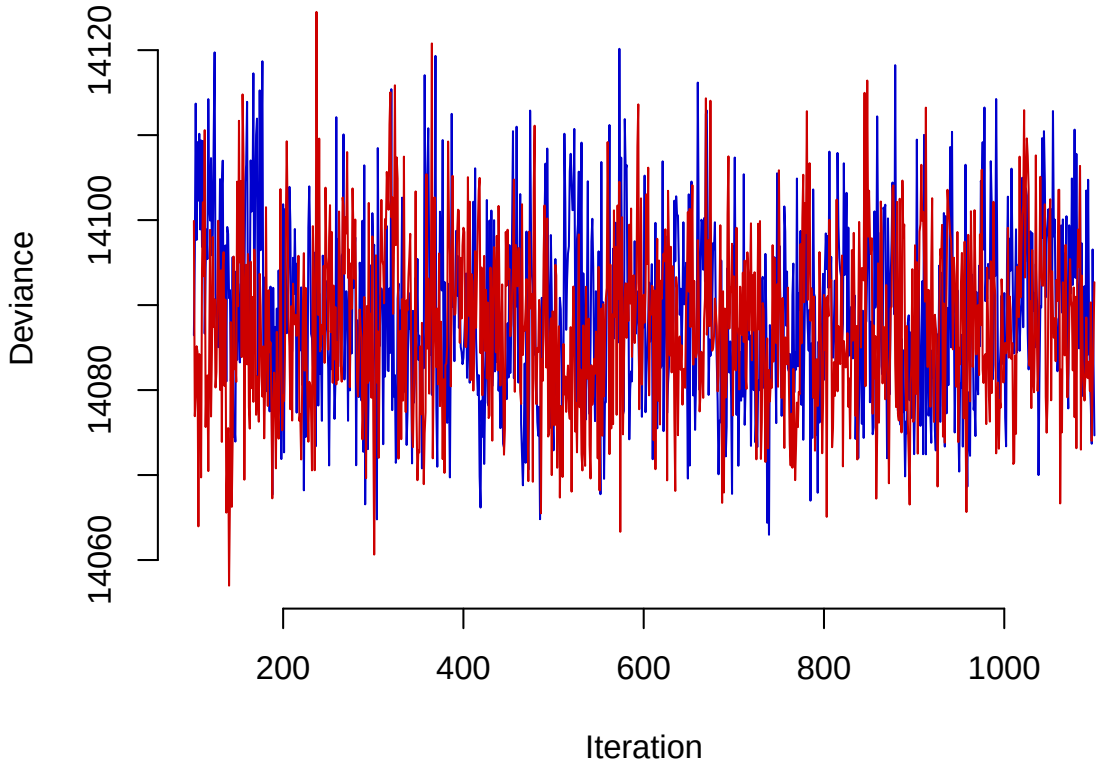

Figure 2: Traceplots of the two parallel chains of the model deviance $D(\boldsymbol{\theta})$.

interest. Nevertheless, a suitable relabeling of the posterior sample must be first conducted to calculate these, as we did in Section 3.4.

In this section, we first show how to easily obtain basic posterior summary statistics of the important model parameters by the mean of routines implemented in the **mixAK** package. Second, we exemplify usage of the posterior samples stored in the object `mod` created in Section 3.4 in connection with the **coda** package to obtain more detailed posterior summaries.

Posterior summary statistics for the GLMM related parameters

Basic posterior summary statistics for many of the above-mentioned quantities have in fact already been calculated by the function `GLMM_MCMC` and are stored as the following components of the objects `mod[[1]]` and `mod[[2]]`: `summ.Deviance` (observed data deviance $D(\boldsymbol{\theta})$), `summ.alpha` (fixed effect $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_R$), `summ.sigma_eps` (square roots of the dispersion parameters $\phi_1, \ldots, \phi_R$), `summ.b.Mean` (overall means $\boldsymbol{\beta}$ of random effects), `summ.b.SDCorr` (standard deviations and the correlation coefficients derived from the overall covariance matrix $\mathbf{D}$ of random effects). To inspect their values in a synoptic form, we simply print the object `mod` (output has been shortened).

```
R> print(mod)

     Generalized linear mixed model for 3 responses estimated using MCMC
     ======================================================================

Penalized expected deviance:
----------------------------
D.expect   p(opt)      PED  wp(opt)     wPED
 14088.3     74.5  14162.8     74.8  14163.1

Deviance posterior summary statistics:
----------------------------------------------
         Mean Std.Dev.  Min.  2.5% 1st Qu. Median 3rd Qu. 97.5%  Max.
Chain 1 14089     10.5 14063 14071   14081  14088   14096 14111 14120
Chain 2 14088     10.1 14057 14069   14080  14087   14094 14108 14125

Posterior summary statistics for fixed effects:
-----------------------------------------------
          Mean Std.Dev.    Min.   2.5% 1st Qu. Median 3rd Qu.  97.5%   Max.
Chain 1 0.0277   0.0128 -0.0124 0.0029  0.0189 0.0278  0.0367 0.0519 0.0735
Chain 2 0.0281   0.0130 -0.0209 0.0024  0.0200 0.0282  0.0367 0.0519 0.0769

Distribution of random effects is a normal mixture with 2 components
--------------------------------------------------------------------
Posterior summary statistics for moments of mixture for random effects:
--------------------------------------------------------------------
Means:
        b.Mean.1(Chain 1) b.Mean.1(Chain 2) b.Mean.2(Chain 1)
Mean               0.3185            0.3103           0.00775
```

```
Std.Dev.              0.0557              0.0566             0.00182
Min.                  0.1422              0.1130             0.00180
2.5%                  0.2056              0.2017             0.00443
1st Qu.               0.2805              0.2716             0.00650
Median                0.3194              0.3097             0.00776
3rd Qu.               0.3545              0.3498             0.00898
97.5%                 0.4288              0.4171             0.01151
Max.                  0.4996              0.4833             0.01401
         b.Mean.2(Chain 2) b.Mean.3(Chain 1) b.Mean.3(Chain 2)
Mean               0.00785             5.5262              5.5273
...

Standard deviations and correlations:
         b.SD.1(Chain 1) b.SD.1(Chain 2) b.Corr.2.1(Chain 1)
Mean              0.8736          0.8728               0.0527
...

Posterior summary statistics for standard deviations
of residuals of continuous responses:
-----------------------------------------------------
         Mean Std.Dev.  Min.  2.5% 1st Qu. Median 3rd Qu. 97.5%  Max.
Chain 1 0.314   0.0101 0.282 0.294   0.307  0.314   0.321 0.334 0.344
Chain 2 0.314   0.0104 0.284 0.293   0.307  0.314   0.321 0.337 0.352
```

On top of the output, we have a summary of the Penalized Expected Deviance introduced in Section 2.4. We return to it in Section 3.8 when discussing a selection of the number of components in this particular example. The rest of the output is devoted to the posterior summary statistics (posterior mean, posterior standard deviation, posterior 0%, 2.5%, 25%, 50%, 75%, 97.5% and 100% quantiles) of the most important model parameters. Among other things, we also directly see the 95% equal-tail credible intervals. In particular, the output shows the posterior summaries of the following model parameters: (i) the model deviance $D(\boldsymbol{\theta})$, e.g., its posterior means estimated using the first and the second chain are 14 089 and 14 088, respectively, the estimated posterior medians have values of 14 088 and 14 087, respectively; (ii) the fixed effects $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_R$; (iii) the overall mean vector $\boldsymbol{\beta}$ of the random effects; (iv) standard deviations and correlations derived from the overall covariance matrix $\mathbf{D}$ of the random effects; (v) square roots of the dispersion parameters $\phi_1, \ldots, \phi_R$.

To get the posterior summary including estimates of Monte Carlo errors in the estimation of the posterior means, one may use the `summary` procedure from the **coda** package applied directly to matrices of sampled values. As illustration, we create a matrix containing the sampled values of the means $\boldsymbol{\beta}$ of random effects, the fixed effect α_3 (slope from the logit model for a variable `spiders`), and the square root of the dispersion parameter ϕ_1 (residual standard deviation from the linear mixed model for a variable `lbili`), and calculate the posterior summary using the **coda** package.

```
R> name.Eb <- paste("b.Mean.", 1:5, sep = "")
R> RegrChain1 <- cbind(mod[[1]]$mixture_b[, name.Eb],
+                      mod[[1]]$alpha, mod[[1]]$sigma_eps)
```

```
R> RegrChain2 <- cbind(mod[[2]]$mixture_b[, name.Eb],
+                      mod[[2]]$alpha, mod[[2]]$sigma_eps)
R> colnames(RegrChain1) <- colnames(RegrChain2) <-
+      c(paste(rep(c("lbili", "platelet", "spiders"), each = 2),
+              ":", rep(c("Intcpt", "Slope"), 3), sep=""),
+        "lbili:res_std_dev")
R> RegrChain1 <- mcmc(RegrChain1)
R> RegrChain2 <- mcmc(RegrChain2)
R> summary(mcmc.list(RegrChain1, RegrChain2))

Iterations = 1:1000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                     Mean      SD Naive SE Time-series SE
lbili:Intcpt      0.31439 0.05628 1.26e-03       1.23e-03
lbili:Slope       0.00780 0.00180 4.03e-05       4.49e-05
platelet:Intcpt   5.52674 0.02215 4.95e-04       4.29e-04
platelet:Slope   -0.00669 0.00115 2.57e-05       2.57e-05
spiders:Intcpt   -2.90215 0.47597 1.06e-02       2.44e-02
spiders:Slope     0.02790 0.01288 2.88e-04       3.31e-04
lbili:res_std_dev 0.31397 0.01028 2.30e-04       2.54e-04

2. Quantiles for each variable:

                     2.5%      25%      50%      75%    97.5%
lbili:Intcpt      0.20293  0.27519  0.31443  0.35174  0.42221
lbili:Slope       0.00429  0.00659  0.00782  0.00895  0.01140
platelet:Intcpt   5.48533  5.51110  5.52607  5.54155  5.56978
platelet:Slope   -0.00898 -0.00743 -0.00668 -0.00591 -0.00442
spiders:Intcpt   -3.92235 -3.19575 -2.86952 -2.57760 -2.07790
spiders:Slope     0.00247  0.01941  0.02797  0.03668  0.05194
lbili:res_std_dev 0.29368  0.30685  0.31391  0.32065  0.33517
```

The above output is now based on values from both sampled Markov chains. It shows, for example, that the estimated posterior mean of the overall mean of the random intercept from the linear mixed model for the variable `lbili` (parameter β_1 in our notation) is equal to 0.31446 and the Monte Carlo error in estimation of the posterior mean equals 0.00118. The related 95% equal-tail credible interval is (0.20195, 0.42167). Analogously, the posterior summary statistics for the standard deviations and the correlation coefficients derived from the overall covariance matrix $\mathbf{D}$ of the random effects are calculated using the following code (output not shown):

```
R> name.SDb <- paste("b.SD.", 1:5, sep = "")
R> SDbChain1 <- mcmc(mod[[1]]$mixture_b[, name.SDb])
R> SDbChain2 <- mcmc(mod[[2]]$mixture_b[, name.SDb])
R> summary(mcmc.list(SDbChain1, SDbChain2))
R> #
R> name.Corb <- paste("b.Corr.", c(2:5, 3:5, 4:5, 5), ".", rep(1:4, 4:1),
+                     sep = "")
R> CorbChain1 <- mcmc(mod[[1]]$mixture_b[, name.Corb])
R> CorbChain2 <- mcmc(mod[[2]]$mixture_b[, name.Corb])
R> summary(mcmc.list(CorbChain1, CorbChain2))
```

Analogously, other **coda** routines like `HPDinterval()`, `densplot()` and others might be used to calculate the highest posterior density (HPD) credible intervals, the posterior densities and other posterior quantities. For example, the HPD credible intervals are calculated (separately for the first and the second chain) using the following code (output reformatted into two columns):

```
R> HPDinterval(mcmc.list(RegrChain1, RegrChain2))
```

```
[[1]]
                     lower    upper
lbili:Intcpt        0.21674  0.4357
lbili:Slope         0.00414  0.0112
platelet:Intcpt     5.48631  5.5697
platelet:Slope     -0.00912 -0.0046
spiders:Intcpt     -3.74831 -2.0247
spiders:Slope       0.00414  0.0526
lbili:res_std_dev   0.29262  0.3320
attr(,"Probability")
[1] 0.95
```

```
[[2]]
                      lower    upper
lbili:Intcpt        0.20257  0.41721
lbili:Slope         0.00417  0.01136
platelet:Intcpt     5.48334  5.56916
platelet:Slope     -0.00896 -0.00442
spiders:Intcpt     -3.97051 -2.04282
spiders:Slope       0.00244  0.05194
lbili:res_std_dev 0.29136  0.33228
attr(,"Probability")
[1] 0.95
```

Posterior summary statistics for the mixture components

With respect to the clustering intended, posterior summary statistics of the mixture parameters (weights $\mathbf{w}$, means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$) are of primary interest as they provide characteristics of the mixture and hence also of the clusters. To this end, the **mixAK** package offers a function `NMixSummComp()` which reports posterior means of the mixture weights, mixture means and mixture covariance matrices based on the relabeled sample. On top of that, standard deviations and correlations derived from the posterior means of the covariance matrices are reported:

```
R> NMixSummComp(mod[[1]])
```

```
Component 1
    Weight:  0.589
    Mean:    -0.211 0.00426 5.58 -0.00559 -4.27
```

```
    Covariance matrix:
          m1        m2        m3        m4        m5
m1  1.80e-01  8.75e-05 -3.62e-02 -3.71e-04  0.528122
m2  8.75e-05  6.60e-05  9.01e-05 -1.84e-05  0.003039
m3 -3.62e-02  9.01e-05  9.41e-02 -1.31e-04 -0.049586
m4 -3.71e-04 -1.84e-05 -1.31e-04  1.09e-04  0.000837
m5  5.28e-01  3.04e-03 -4.96e-02  8.37e-04 14.986695
    Standard deviations:  0.424 0.00813 0.307 0.0105 3.87

    Correlation matrix:
       m1      m2      m3      m4      m5
m1  1.0000  0.0254 -0.2780 -0.0835  0.3215
m2  0.0254  1.0000  0.0361 -0.2161  0.0966
m3 -0.2780  0.0361  1.0000 -0.0409 -0.0418
m4 -0.0835 -0.2161 -0.0409  1.0000  0.0207
m5  0.3215  0.0966 -0.0418  0.0207  1.0000

--------------------------------------------

Component 2
    Weight:  0.411
    Mean:    1.09 0.0128 5.46 -0.00811 -0.825

    Covariance matrix:
         m1        m2        m3        m4       m5
m1  0.61258 -0.004468  0.036689 -0.002318  0.32495
m2 -0.00447  0.000941 -0.000397  0.000171  0.00801
m3  0.03669 -0.000397  0.157806 -0.000479 -0.16971
m4 -0.00232  0.000171 -0.000479  0.000531 -0.00217
m5  0.32495  0.008009 -0.169713 -0.002168  5.84178
    Standard deviations:  0.783 0.0307 0.397 0.0231 2.42

    Correlation matrix:
      m1      m2      m3      m4      m5
m1  1.000 -0.1861  0.1180 -0.1285  0.1718
m2 -0.186  1.0000 -0.0326  0.2419  0.1080
m3  0.118 -0.0326  1.0000 -0.0523 -0.1768
m4 -0.128  0.2419 -0.0523  1.0000 -0.0389
m5  0.172  0.1080 -0.1768 -0.0389  1.0000

--------------------------------------------
```

That is, for example the posterior means of the parameters of the first mixture component based on the performed MCMC simulation and a relabeled sample (the first chain) are

$$\begin{aligned} \widehat{w}_1 &= \mathsf{E}\big(w_1 \,\big|\, \mathbf{Y} = \mathbf{y}\big) = 0.589, \\ \widehat{\boldsymbol{\mu}}_1 &= \mathsf{E}\big(\boldsymbol{\mu}_1 \,\big|\, \mathbf{Y} = \mathbf{y}\big) = \big(-0.211,\ 0.00426,\ 5.58,\ -0.00559,\ -4.27\big)^\top, \end{aligned} \tag{20}$$

$$\widehat{\mathbf{D}}_1 = \mathsf{E}\big(\mathbf{D}_1 \,\big|\, \mathbf{Y} = \mathbf{y}\big)$$

$$= \begin{pmatrix} 0.180 & 0.0000875 & -0.0362 & -0.000371 & 0.528 \\ 0.0000875 & 0.0000660 & 0.0000901 & -0.0000184 & 0.00304 \\ -0.0362 & 0.0000901 & 0.0941 & -0.000131 & -0.0496 \\ -0.000371 & -0.0000184 & -0.000131 & 0.000109 & 0.000837 \\ 0.528 & 0.00304 & -0.0496 & 0.000837 & 15.0 \end{pmatrix}.$$

Analogously to the case of the GLMM related parameters, more detailed posterior summary of the mixture parameters can be obtained by using the **coda** routines with the posterior samples of the mixture parameters extracted by the mean of the **mixAK** function `NMixChainComp()`. For example, more detailed posterior summary statistics and the 95% HPD credible intervals for the mixture means based on the first sampled chain are obtained using:

```
R> summary(mcmc(muSamp1))

Iterations = 1:1000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean      SD Naive SE Time-series SE
mu1.1 -0.21126 0.06511 2.06e-03       5.02e-03
mu1.2  0.00426 0.00197 6.21e-05       1.39e-04
mu1.3  5.57800 0.04174 1.32e-03       4.11e-03
mu1.4 -0.00559 0.00121 3.83e-05       5.48e-05
mu1.5 -4.27154 0.73912 2.34e-02       5.22e-02
mu2.1  1.08805 0.14350 4.54e-03       1.18e-02
mu2.2  0.01283 0.00421 1.33e-04       1.64e-04
mu2.3  5.45775 0.05600 1.77e-03       3.59e-03
mu2.4 -0.00811 0.00265 8.37e-05       8.82e-05
mu2.5 -0.82469 0.39729 1.26e-02       1.97e-02

2. Quantiles for each variable:

           2.5%      25%      50%      75%    97.5%
mu1.1 -0.334236 -0.25434 -0.21351 -0.16864 -0.08649
mu1.2  0.000298  0.00292  0.00422  0.00568  0.00790
mu1.3  5.494958  5.55050  5.58047  5.60731  5.65415
mu1.4 -0.008006 -0.00633 -0.00563 -0.00481 -0.00306
mu1.5 -6.005397 -4.69913 -4.20171 -3.74902 -3.02452
mu2.1  0.799439  0.99361  1.08969  1.18661  1.35502
mu2.2  0.004342  0.01009  0.01260  0.01539  0.02165
mu2.3  5.355440  5.41873  5.45577  5.49392  5.56880
```

```
mu2.4 -0.013460 -0.00978 -0.00806 -0.00627 -0.00310
mu2.5 -1.626945 -1.09172 -0.82137 -0.57188 -0.04295


R> HPDinterval(mcmc(muSamp1))

          lower    upper
mu1.1 -0.340065 -0.09240
mu1.2  0.000785  0.00825
mu1.3  5.497083  5.65449
mu1.4 -0.008090 -0.00328
mu1.5 -5.836688 -2.94372
mu2.1  0.782802  1.33216
mu2.2  0.004047  0.02134
mu2.3  5.355365  5.56869
mu2.4 -0.013130 -0.00285
mu2.5 -1.546493  0.01009
attr(,"Probability")
[1] 0.95
```

3.7. Clustering

In this section, we first discuss the possibilities of characterization of the mixture components in more detail, i.e., clusters that were found by our analysis. Second, we exemplify usage of the posterior samples stored in the object `mod` for classification of individual subjects into those clusters.

Cluster specific mean longitudinal profiles

As we mentioned in Section 3.5, clusters in our model are characterized by the mixture parameters (weights $w_1, \ldots, w_K$, means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$) and their derivatives for which we already calculated posterior summary statistics, see, e.g., Equation (20) for the posterior means of the parameters of the first mixture component. To visualize those results and to see better the characteristics of the different clusters, we can calculate estimated values of the cluster specific mean longitudinal profiles of the response variables. To this end, a generic function `fitted()` was extended in the **mixAK** package by a method for objects of class `GLMM_MCMC`. It provides the empirical Bayes estimates of those longitudinal profiles. In the following, let $\widehat{\boldsymbol{\mu}}_k$, $\widehat{\mathbf{D}}_k$, $k = 1, \ldots, K$, be the posterior means of mixture means and covariance matrices, respectively. Further, let $\widehat{\boldsymbol{\mu}}_{k,r}$ and $\widehat{\mathbf{D}}_{k,r}$ denote for each $k = 1, \ldots, K$, $r = 1, \ldots, R$ a proper subvector of $\widehat{\boldsymbol{\mu}}_k$ and a submatrix of $\widehat{\mathbf{D}}_k$, respectively, corresponding to the random effects vector for the rth response variable. In particular, the `fitted` function calculates the values of

$$\begin{aligned}\mathsf{E}\big(Y_{new,r,j} \,\big|\, U_{new} = k;\, \widehat{\boldsymbol{\theta}}\big) \\ = \int \mathsf{E}\big(Y_{new,r,j} \,\big|\, \mathbf{B}_{new,r} = \mathbf{b}_{new,r};\, \widehat{\boldsymbol{\alpha}}\big)\, \varphi(\mathbf{b}_{new,r};\, \widehat{\boldsymbol{\mu}}_{k,r},\, \widehat{\mathbf{D}}_{k,r})\, \mathrm{d}\mathbf{b}_{new,r}, \quad (21)\end{aligned}$$

$r = 1, \ldots, R$, $k = 1, \ldots, K$, $j = 1, \ldots, n_{new}$ for selected combinations of covariates entering the model expression (4) of $\mathsf{E}\big(Y_{new,r,j} \,\big|\, \mathbf{B}_{new,r} = \mathbf{b}_{new,r}; \widehat{\boldsymbol{\alpha}}\big)$.

In our example, the only covariates included in the model are the visit times. The following code then calculates the values of (21) for $r = 1, \ldots, R$, $k = 1, \ldots, K$, and the covariate values $t_{new,1} = 0$, $t_{new,2} = 0.3$, …, $t_{new,101} = 30$ (i.e., $n_{new} = 101$):

```
R> delta <- 0.3
R> tpred <- seq(0, 30, by = delta)
R> fit <- fitted(mod[[1]], x =list("empty", "empty", tpred),
+                           z =list(tpred, tpred, "empty"),
+                 glmer = TRUE)
R> names(fit) <- c("lbili", "platelet", "spiders")
```

The covariate combinations for which (21) is to be calculated are given as arguments `x` (fixed effects covariates) and `z` (random effects covariates).

Further, we point out that with `glmer = FALSE` in the call to `fitted`, calculation is faster but the integral in (21) is only rather inaccurately approximated by $\mathsf{E}\big(Y_{new,r,j} \,\big|\, \mathbf{B}_{new,r} = \widehat{\boldsymbol{\mu}}_{k,r}; \widehat{\boldsymbol{\alpha}}\big)$, that is, the random effect values are replaced by their means rather than being integrated out.

The resulting object `fit` is a `list` of length $R = 3$, where each `list` component is a $n_{new} \times K$ matrix. The kth column, $k = 1, \ldots, K$, of the rth, $r = 1, \ldots, R$, matrix gives the calculated values of $\mathsf{E}\big(Y_{\cdot,r,\cdot} \,\big|\, U_\cdot = k, \widehat{\boldsymbol{\theta}}\big)$ for the covariate combinations corresponding to the vectors or matrices specified by the `x`, `x2`, `z`, `z2` arguments. For instance, the first three values of the component specific estimated longitudinal profiles for the variable `platelet` are as follows.

```
R> print(fit[["platelet"]][1:3,], digits = 5)
```

```
       [,1]   [,2]
[1,] 277.28 253.83
[2,] 276.81 253.18
[3,] 276.34 252.55
```

With respect to the interpretation of the clusters found, it is probably more useful to draw the estimated cluster specific mean longitudinal profiles, perhaps together with the data observed. To achieve this, we use a mild modification of the code shown in Section 3.2 for Figure 1. A plot of observed longitudinal profiles together with the cluster specific mean profiles for a variable `lbili` (see the upper panel of Figure 3) is drawn using the following commands:

```
R> K <- mod[[1]]$prior.b$Kmax
R> clCOL <- c("darkgreen", "red3")
R> plotProfiles(ip = ip, data = PBC910, var = "lbili", tvar = "month",
+          col = "azure3", main = "Log(bilirubin)",
+          xlab = "Time (months)", ylab = "Log(bilirubin)")
R> for (k in 1:K) lines(tpred, fit[["lbili"]][, k], col = clCOL[k], lwd = 2)
```

The remaining panels of Figure 3 are drawn analogously. We now see from Figure 3 that the first (green) cluster is characterized by lower bilirubin values and also by less frequent blood

vessel malformations whose probability only slightly rise over time. On the other hand, the second (red) cluster corresponds to subjects with higher bilirubin values and more frequent blood vessel malformations whose occurrence increases more steeply over time. With respect to the longitudinal evolution of the platelet counts, both groups behave almost equally on average.

The full code to draw Figure 3 is the following.

```
R> K <- mod[[1]]$prior.b$Kmax
R> clCOL <- c("darkgreen", "red3")
R> obsCOL <- "azure3"
```

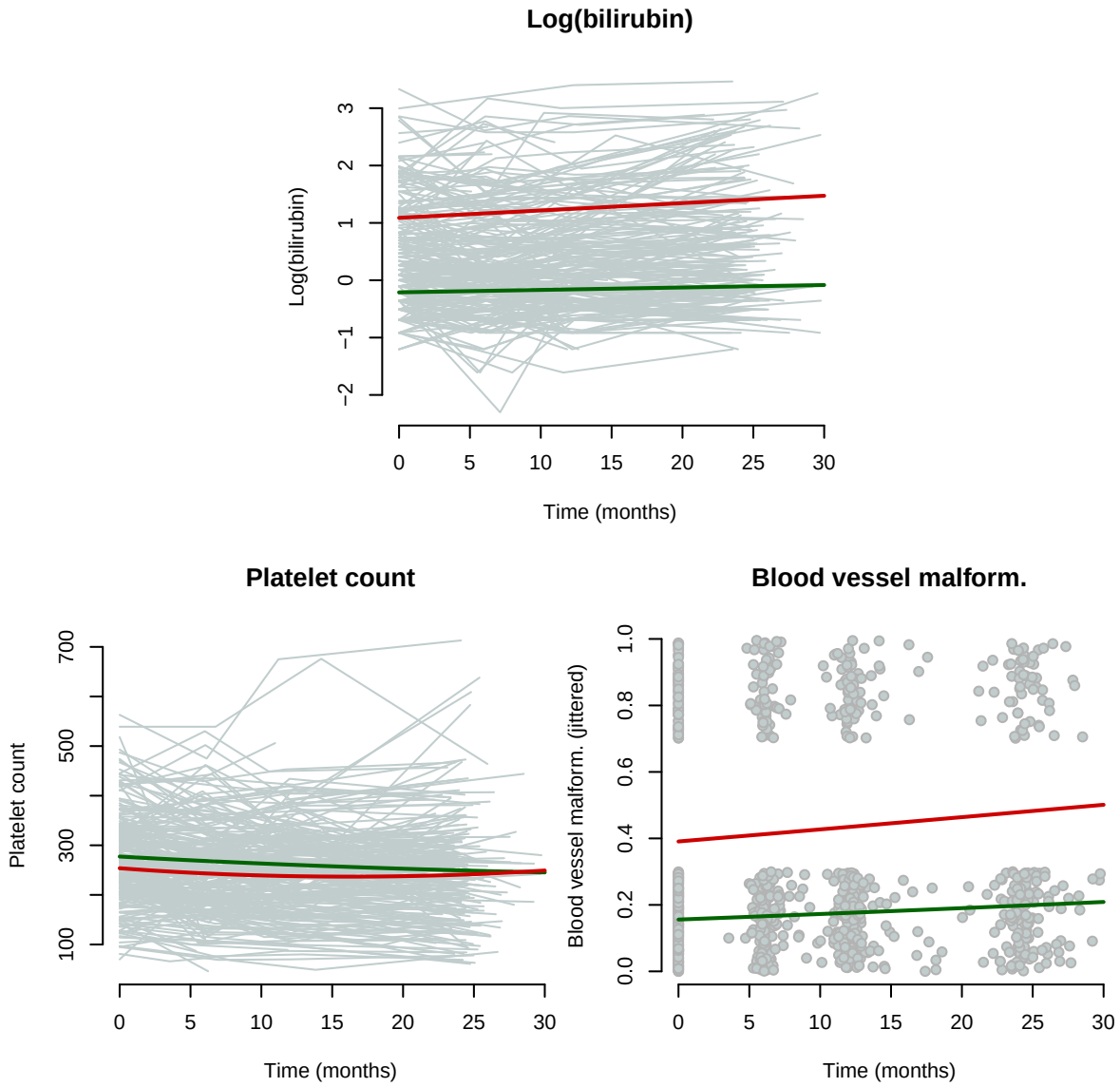

Figure 3: Observed longitudinal profiles of considered markers together with the estimated cluster specific mean profiles (cluster 1 in green, cluster 2 in red).

```
R> layout(autolayout(3))
R> plotProfiles(ip = ip, data = PBC910, var = "lbili", tvar = "month",
+         auto.layout = FALSE, main = "Log(bilirubin)",
+         xlab = "Time (months)", ylab = "Log(bilirubin)", col = obsCOL)
R> for (k in 1:K) lines(tpred, fit[["lbili"]][, k], col = clCOL[k], lwd = 2)
R> plotProfiles(ip = ip, data = PBC910, var = "platelet", tvar = "month",
+         auto.layout = FALSE, main = "Platelet count",
+         xlab = "Time (months)", ylab = "Platelet count", col = obsCOL)
R> for (k in 1:K) lines(tpred, fit[["platelet"]][, k], col = clCOL[k], lwd = 2)
R> plotProfiles(ip = ip, data = PBC910, var = "jspiders", tvar = "month",
+         lines = FALSE, points = TRUE,
+         auto.layout = FALSE, main = "Blood vessel malform.",
+         xlab = "Time (months)", ylab = "Blood vessel malform. (jittered)",
+         bg = obsCOL, col = "grey70")
R> for (k in 1:K) lines(tpred, fit[["spiders"]][, k], col = clCOL[k], lwd = 2)
```

Individual component probabilities

We explain in Section 1.1 that model-based clustering is based on the individual component probabilities (ICPs) $p_{i,k} = p_{i,k}(\boldsymbol{\theta}) = \mathsf{P}\big(U_i = k \,\big|\, \mathbf{Y}_i = \mathbf{y}_i; \boldsymbol{\theta}\big)$, $i = 1, \ldots, N$, $k = 1, \ldots, K$, see Equation (3). Upon running the `NMixRelabel()` procedure in Section 3.4, an MCMC sample from the posterior distribution of $p_{i,k}$ is available within the resulting `GLMM_MCMC` objects `mod[[1]]` and `mod[[2]]` for each $i = 1, \ldots, N$, $k = 1, \ldots, K$. These are the values

$$p_{i,k}^{(m)} = p_{i,k}\big(\boldsymbol{\theta}^{(m)}\big) = \mathsf{P}\big(U_i = k \,\big|\, \mathbf{Y}_i = \mathbf{y}_i; \boldsymbol{\theta}^{(m)}\big), \qquad m = 1, \ldots, M, \tag{22}$$

where $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}$ is an MCMC sample from the posterior distribution $p\big(\boldsymbol{\theta} \,\big|\, \mathbf{y}\big)$. In the following paragraphs, we show how to use different characteristics of the posterior distributions of each $p_{i,k}$ for clustering purposes.

Before we proceed to classification itself, we show how to access the values of (22). They are stored as an $M \times (N \cdot K)$ matrix `comp.prob` of `GLMM_MCMC` objects where each row corresponds to one MCMC iteration and the columns of the mth row provide in a sequence the values of $p_{1,1}^{(m)}, \ldots, p_{1,K}^{(m)}, \ldots, p_{N,1}^{(m)}, \ldots, p_{N,K}^{(m)}$. The labeling of components corresponds to that obtained by applying a relabeling algorithm in Section 3.4. For example, the first three sampled values (from the first chain) of the ICPs of the first four subjects in the dataset are as follows:

```
R> print(mod[[1]]$comp.prob[1:3, 1:8])
```

```
     P(1,1) P(1,2) P(2,1) P(2,2)  P(3,1) P(3,2)   P(4,1) P(4,2)
[1,]  0.803  0.197  0.704  0.296 0.84183  0.158 0.054087  0.946
[2,]  0.729  0.271  0.437  0.563 0.16346  0.837 0.013635  0.986
[3,]  0.187  0.813  0.173  0.827 0.00965  0.990 0.000235  1.000
```

Finally, we point out that a similar matrix called `comp.prob_b` is present inside the `GLMM_MCMC` objects. It provides the values of $\mathsf{P}\big(U_i = k \,\big|\, \mathbf{Y}_i = \mathbf{y}_i; \mathbf{B}_i = \mathbf{b}_i^{(m)}, \boldsymbol{\theta}^{(m)}\big)$ (which are different from those given by Equation 22), $i = 1, \ldots, N$, $k = 1, \ldots, K$, $m = 1, \ldots, M$, where $\mathbf{b}_i^{(m)}$ are

the sampled values of the ith subject random effects. The values of $\mathsf{P}\big(U_i = k \,\big|\, \mathbf{Y}_i = \mathbf{y}_i;\, \mathbf{B}_i = \mathbf{b}_i^{(m)},\, \boldsymbol{\theta}^{(m)}\big)$ might be of interest in some situations. Nevertheless, we shall not use them for any purposes in this manuscript.

Clustering procedure

As indicated in Section 2.3, the clustering procedure in a Bayesian setting can straightforwardly be based on the posterior means $\widehat{p}_{i,k} = \mathsf{E}\big\{p_{i,k}(\boldsymbol{\theta}) \,\big|\, \mathbf{Y} = \mathbf{y}\big\}$, $i = 1, \ldots, N$, $k = 1, \ldots, K$, of the ICPs, see also Equation (14). Their MCMC based estimates are included in the `GLMM_MCMC` objects as an $N \times K$ matrix `poster.comp.prob`. We print its first three rows which provide the MCMC estimated values of $\widehat{p}_{i,k}$ for $i = 1, 2, 3$, $k = 1, 2$:

```
R> print(mod[[1]]$poster.comp.prob[1:3,])
```

```
      [,1]  [,2]
[1,] 0.683 0.317
[2,] 0.558 0.442
[3,] 0.489 0.511
```

The most common classification procedure which assigns subject i into group $g(i)$ satisfying $\widehat{p}_{i,g(i)} = \max_{k=1,\ldots,K} \widehat{p}_{i,k}$ is then achieved by the following code which also stores the values of $\widehat{p}_{i,g(i)}$, $i = 1, \ldots, N$, in a vector `pMean`:

```
R> groupMean <- apply(mod[[1]]$poster.comp.prob, 1, which.max)
R> pMean <- apply(mod[[1]]$poster.comp.prob, 1, max)
R> table(groupMean)
```

```
groupMean
  1   2
161  99
```

That is, 161 and 99 subjects are classified in the first and the second group, respectively, and represented in Figure 3 by the green and the red cluster specific mean profiles, respectively.

Nevertheless, the posterior mean is only one possible one-number characteristic of the posterior distribution. A reasonable, and often even more suitable characteristic is the posterior median which can also base the clustering procedure. The posterior medians, together with the 2.5% and 97.5% quantiles of the posterior distributions of the ICPs are available in the `quant.comp.prob` elements of the `GLMM_MCMC` objects. As illustration, we again provide the posterior medians of the ICPs for the first three subjects:

```
R> print(mod[[1]]$quant.comp.prob[["50%"]][1:3,])
```

```
      [,1]  [,2]
[1,] 0.786 0.214
[2,] 0.615 0.385
[3,] 0.495 0.505
```

Analogously, the 2.5% and 97.5% quantiles of the posterior distributions of the ICPs for the first three subjects are obtained as

```
R> print(mod[[1]]$quant.comp.prob[["2.5%"]][1:3,])

       [,1]   [,2]
[1,] 0.0327 0.0111
[2,] 0.0311 0.0385
[3,] 0.0217 0.0518


R> print(mod[[1]]$quant.comp.prob[["97.5%"]][1:3,])

      [,1]  [,2]
[1,] 0.989 0.967
[2,] 0.962 0.969
[3,] 0.948 0.978
```

Classification procedure which assigns the subject into a group for which the posterior median of the individual component probability is maximal is implemented as follows.

```
R> groupMed <- apply(mod[[1]]$quant.comp.prob[["50%"]], 1, which.max)
R> pMed <- apply(mod[[1]]$quant.comp.prob[["50%"]], 1, max)
R> table(groupMed)

groupMed
  1   2
162  98


R> table(groupMean, groupMed)

         groupMed
groupMean   1   2
        1 161   0
        2   1  98
```

Classification of one subject (id 66) has changed. Nevertheless, both the posterior mean and the posterior median of $p_{i,g(i)}$ are close to the classification threshold of 0.5 as we illustrate using the following code:

```
R> pMeanMed <- data.frame(Mean = pMean, Median = pMed)
R> rownames(pMeanMed) <- unique(PBC910$id)
R> print(pMeanMed[groupMean != groupMed, ])

    Mean Median
66 0.502  0.505
```

Finally, we point out that the `GLMM_MCMC` objects additionally contain related matrix elements called `poster.comp.prob_u` and `poster.comp.prob_b` having the same structure as the `poster.comp.prob` matrix. They all provide the MCMC based estimates of the posterior means of the ICPs $p_{i,k}$, $i = 1, \ldots, N$, $k = 1, \ldots, K$, nevertheless, calculated in different ways stemming from different representations of the conditional probabilities $\mathsf{P}(U_i = k \,|\, \mathbf{Y} = \mathbf{y}) = \mathsf{E}\big\{p_{i,k}(\boldsymbol{\theta}) \,\big|\, \mathbf{Y} = \mathbf{y}\big\}$. That is,

$$\widehat{p}_{i,k} = \mathsf{P}(U_i = k \,|\, \mathbf{Y} = \mathbf{y}) = \mathsf{E}\big\{\mathbb{I}(U_i = k) \,\big|\, \mathbf{Y} = \mathbf{y}\big\} \tag{23}$$

$$= \int \mathsf{P}(U_i = k \,|\, \mathbf{Y}_i = \mathbf{y}_i;\, \mathbf{B}_i = \mathbf{b}_i, \boldsymbol{\theta})\, p(\boldsymbol{\theta},\, \mathbf{b}_i \,|\, \mathbf{y})\, d(\boldsymbol{\theta},\, \mathbf{b}_i), \tag{24}$$

$i = 1, \ldots, N$, $k = 1, \ldots, K$. The `poster.comp.prob_u` values are calculated using the representation (23) as

$$\widehat{p}^{[1]}_{i,k} = M^{-1} \sum_{m=1}^{M} \mathbb{I}(u_i^{(m)} = k), \qquad i = 1, \ldots, N,\ k = 1, \ldots, K.$$

Analogously, the `poster.comp.prob_b` values are calculated using the representation (24) as

$$\widehat{p}^{[2]}_{i,k} = M^{-1} \sum_{m=1}^{M} \mathsf{P}\big(U_i = k \,\big|\, \mathbf{Y}_i = \mathbf{y}_i;\, \mathbf{B}_i = \mathbf{b}_i^{(m)},\, \boldsymbol{\theta}^{(m)}\big), \qquad i = 1, \ldots, N,\ k = 1, \ldots, K.$$

Additionally, the `GLMM_MCMC` objects include the components `quant.comp.prob_b` and `comp.prob_b` which have the same meaning as the components `quant.comp.prob` and `comp.prob`, nevertheless, they are now based on the posterior samples for the values of $\mathsf{P}\big(U_i = k \,\big|\, \mathbf{Y}_i = \mathbf{y}_i;\, \mathbf{B}_i = \mathbf{b}_i^{(m)},\, \boldsymbol{\theta}^{(m)}\big)$.

Posterior distribution and credible intervals of the individual component probabilities

Up to now, only one-number characteristics of the posterior distributions of the ICPs were used for the purposes of clustering which in fact corresponds to the use of the point estimates in a frequentist setting. Nevertheless, the one-number characteristics have a different informative value for different subjects due to different variability of the posterior distributions, i.e., as point estimates they have a different precision when talking in frequentist terms. To illustrate this, Figure 4 shows histograms of sampled values of $p_{i,1}$ (i.e., their estimated posterior densities) for three subjects (id 2, 7, 11) together with their posterior mean and median. The figure was created by applying a standard R function `hist` on appropriate columns of the `mod[[1]]$comp.prob` matrix. The full code towards Figure 4 is the following.

```
R> IDS <- unique(PBC910$id)
R> N <- ncol(mod[[1]]$comp.prob) / K
R> ID <- c(2, 7, 11)
R> par(mfrow = c(1, 3))
R> for (id in ID){
+    i <- (1:N)[IDS == id]
+    hist(mod[[1]]$comp.prob[, (i - 1) * K + 1], xlim = c(0, 1), prob = TRUE,
+       xlab = expression(paste("P(U=1|Y=y; ", theta, ")",
+                               sep = "")),
```

```
+       col = rainbow_hcl(1, start=60),
+       main = paste("ID ", id, "   (",
+         format(round(pMean[i], 3), nsmall = 3), ",  ",
+         format(round(pMed[i], 3), nsmall = 3), ")",
+         sep=""))
+  }
```

Besides a different variability of the posterior distributions of the ICPs, Figure 4 also shows that those posterior distributions are often skewed which lead us to conclude that if only a one-number characteristic of the posterior distribution should be used, then the posterior medians are probably a better choice than the posterior means.

When also taking into account the variability of the posterior distribution of the ICPs, we also suggest incorporating the credible intervals of the ICPs in the classification. We ultimately classify the ith subject into group $g(i)$ only if $\widehat{p}_{i,g(i)} = \max_{k=1,\ldots,K} \widehat{p}_{i,k}$ ($\widehat{p}_{i,k}$ are suitable one-number characteristics of the posterior distributions of the ICPs, e.g., the posterior means or medians) and at the same time the lower limit of the credible interval for $p_{i,g(i)}$ exceeds a certain threshold, for instance 0.5. Indeed, some subjects may remain unclassified with such a rule. Nevertheless, this may not be a problem in practice if we view this classification exercise as one step in a multi-level classification procedure. Subjects with unknown cluster pertinence (and only those subjects) may proceed to another (more complicated or more expensive) level of the full classification procedure. A classification which exploits the HPD credible intervals (calculated by the mean of the **coda** function `HPDinterval()`) for the ICPs with a threshold for ultimate classification of 0.5 is implemented by the following code. The

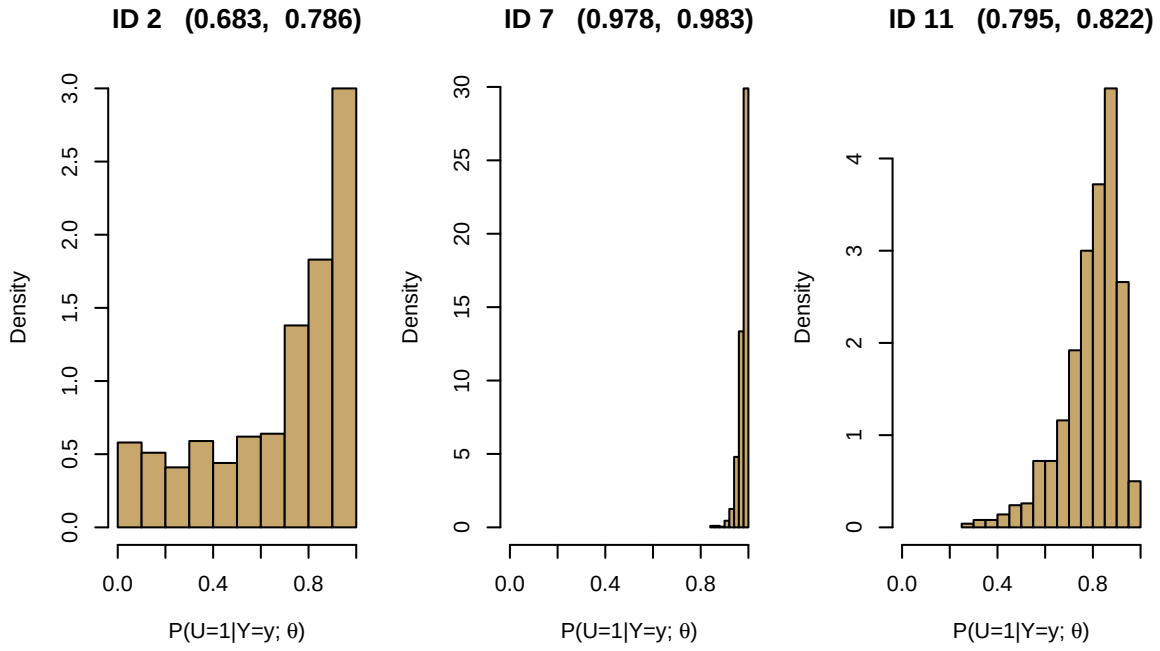

Figure 4: Histograms of sampled values of the individual component probabilities $p_{i,1}(\boldsymbol{\theta})$ for three subjects. Above the plot: posterior mean and posterior median of $p_{i,1}(\boldsymbol{\theta})$.

missing value indicator `NA` is used to mark the classification for those subjects for whom none of the lower limits of the credible intervals exceeds the threshold.

```
R> pHPD <- HPDinterval(mcmc(mod[[1]]$comp.prob))
R> pHPDlower <- matrix(pHPD[, "lower"], ncol = 2, byrow = TRUE)
R> pHPDupper <- matrix(pHPD[, "upper"], ncol = 2, byrow = TRUE)
R> rownames(pHPDlower) <- rownames(pHPDupper) <- unique(PBC910$id)
R> groupHPD <- groupMed
R> groupHPD[groupHPD == 1 & pHPDlower[, 1] <= 0.5] <- NA
R> groupHPD[groupHPD == 2 & pHPDlower[, 2] <= 0.5] <- NA
R> table(groupHPD, useNA = "ifany")

groupHPD
   1    2 <NA>
 123   70   67
```

That is, applying the above-described classification procedure based on the credible intervals for the ICPs lead to 123 subjects being classified in group 1 and 70 subjects being classified in group 2. For 67 subjects, the ultimate classification cannot be determined due to the fact that their observed longitudinal markers do not provide enough certainty for classification into any of the two considered groups. To see how many subjects originally classified in the first and the second group, respectively, remained unclassified using the HPD credible intervals, we create a variable `groupMeanHPD` which will be a combination of `groupMean` and `groupHPD`:

```
R> groupMeanHPD <- as.character(groupMean)
R> groupMeanHPD[is.na(groupHPD) & groupMean == 1] <- "1_NA"
R> groupMeanHPD[is.na(groupHPD) & groupMean == 2] <- "2_NA"
R> groupMeanHPD <- factor(groupMeanHPD, levels = c("1", "2", "1_NA", "2_NA"))
R> table(groupMeanHPD)

groupMeanHPD
   1    2 1_NA 2_NA
 123   70   38   29
```

Observed longitudinal profiles by clusters

Results of the clustering procedure may be visualized by plotting the observed longitudinal profiles using different colors according to the subject's classification. To achieve this for a classification stored in a vector `groupHPD`, the following procedure leading to Figure 5 can be followed. We first add to the original `data.frame` `PBC910` `factor`s created from the classification vectors `groupMed` and `groupHPD` while turning `NA` into one of the `groupMed` `factor` levels.

```
R> TAB <- table(PBC910$id)
R> PBC910$groupMed <- factor(rep(groupMed, TAB))
R> PBC910$groupHPD <- factor(rep(groupHPD, TAB))
R> PBC910$groupHPD <- addNA(PBC910$groupHPD)
```

Second, we again extract the longitudinal profiles while keeping also the group indicators in the resulting object.

```
R> ip <- getProfiles(t = "month",
+     y = c("lbili", "platelet", "spiders", "jspiders", "groupMed", "groupHPD"),
+     id = "id", data = PBC910)
R> print(ip[[1]])

  month   lbili platelet spiders jspiders groupMed groupHPD
1  0.00  0.0953      221       1    0.986        1     <NA>
```

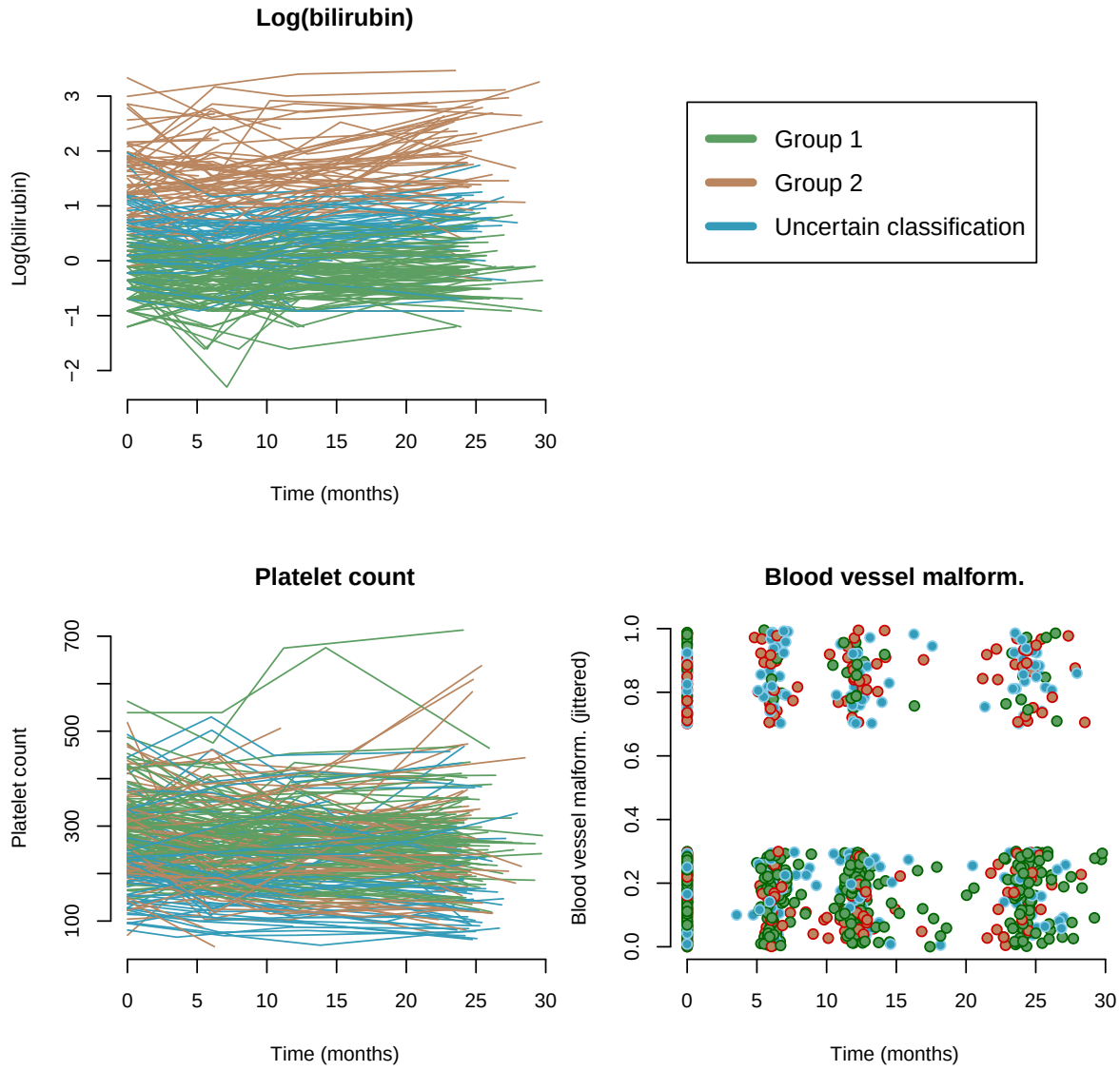

Figure 5: Observed longitudinal profiles of considered markers colored according to classification based on the HPD credible intervals for the individual component probabilities.

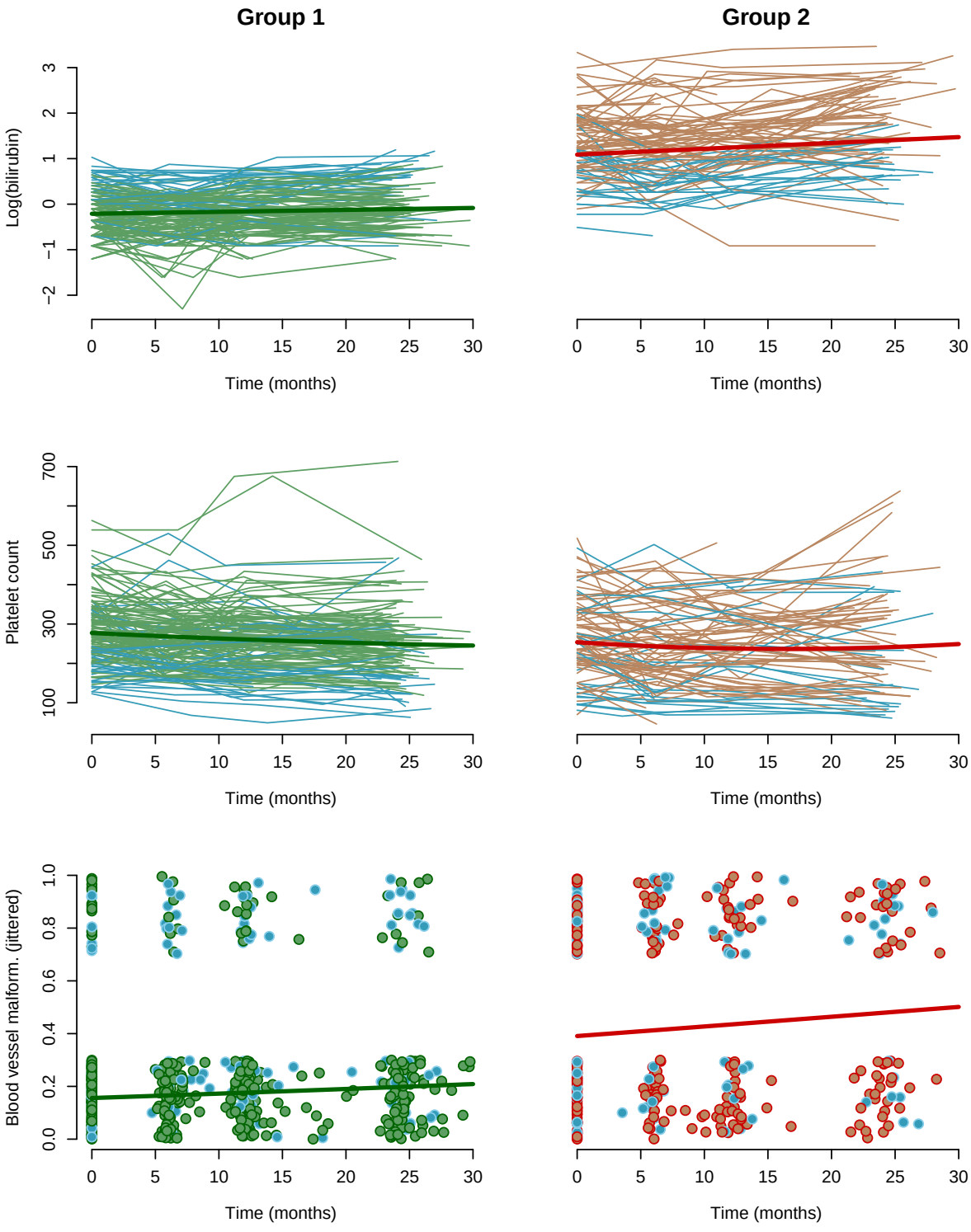

Figure 6: Observed longitudinal profiles of considered markers by group according to the posterior medians of ICPs together with the estimated cluster specific mean profiles (thick lines). Left panel: group 1, right panel: group 2. Data from subjects being unclassified according to the HPD credible interval for the ICPs are drawn in a light blue.

```
2  5.98 -0.2231      188       1    0.884        1     <NA>
3 11.99  0.0000      161       1    0.817        1     <NA>
4 25.23  0.6419      122       1    0.856        1     <NA>
```

Finally, we plot the observed longitudinal profiles while using different colors for profiles of subjects belonging to different groups according to a variable `groupHPD`. To this end, arguments `gvar` and `col` of the **mixAK** function `plotProfiles()` are exploited:

```
R> GCOL <- rainbow_hcl(3, start = 220, end = 40, c = 50, l = 60)[c(2, 3, 1)]
R> GCOL2 <- c("darkgreen", "red3", "skyblue")
R> names(GCOL) <- names(GCOL2) <- levels(PBC910$groupHPD)
R> layout(autolayout(4))
R> # Log(bilirubin):
R> plotProfiles(ip = ip, data = PBC910, var = "lbili", tvar = "month",
+     gvar = "groupHPD", col = GCOL,
+     auto.layout = FALSE, main = "Log(bilirubin)",
+     xlab = "Time (months)", ylab = "Log(bilirubin)")
R> # Legend:
R> plot(c(0, 100), c(0, 100), type = "n",
+       xaxt = "n", yaxt = "n", xlab = "", ylab = "")
R> legend(0, 90, legend = c("Group 1", "Group 2",
+                           "Uncertain classification"),
+         lty = 1, lwd = 5, col = GCOL, y.intersp = 1.5, cex=1.1)
R> # Platelet count:
R> plotProfiles(ip = ip, data = PBC910, var = "platelet", tvar = "month",
+     gvar = "groupHPD", col = GCOL,
+     auto.layout = FALSE, main = "Platelet count",
+     xlab = "Time (months)", ylab = "Platelet count")
R> # Blood vessel malformations
R> plotProfiles(ip = ip, data = PBC910, var = "jspiders",  tvar = "month",
+     lines = FALSE, points = TRUE,
+     gvar = "groupHPD", bg = GCOL, col = GCOL2,
+     auto.layout = FALSE, main = "Blood vessel malform.",
+     xlab = "Time (months)", ylab = "Blood vessel malform. (jittered)")
```

Figure 5 now shows even better than the previously discussed Figure 3 that group 1 is composed of subjects with lower bilirubin values compared to group 2. Further, we see that the platelet count values do not contribute greatly to the classification and that the occurrence of blood vessel malformations is visibly lower in group 1 than in group 2. All these findings are indeed in accordance with our findings based on the estimated cluster specific mean profiles discussed earlier with Figure 3. Probably a clearer picture can be obtained by plotting the longitudinal profiles of subjects belonging to different groups in separate plots as shown on Figure 6:

```
R> ips <- list()
R> for (k in 1:K){
+    ips[[k]] <- getProfiles(t = "month",
```

```
+       y = c("lbili", "platelet", "spiders", "jspiders", "groupHPD"),
+       id = "id", data = subset(PBC910, groupMed == k))
+  }
R> yvars <- c("lbili", "platelet", "jspiders")
R> fit.yvars <- c("lbili", "platelet", "spiders")
R> ylabs <- c("Log(bilirubin)", "Platelet count",
+             "Blood vessel malform. (jittered)")


R> par(mfrow = c(3, 2))
R> for (v in 1:length(yvars)){
+    for (k in 1:2){
+      YLIM <- range(PBC910[, yvars[v]], na.rm = TRUE)
+      plotProfiles(ip = ips[[k]],
+         data = subset(PBC910, groupMed == k),
+         var = yvars[v], tvar = "month", ylim = YLIM,
+         gvar = "groupHPD", col = if (v <= 2) GCOL else GCOL2, bg = GCOL,
+         xlab = "Time (months)",
+         ylab = ifelse(k == 1, ylabs[v], ""),
+         yaxt = ifelse(k == 1, "s", "n"),
+         lines = (v <= 2), points = (v == 3), cex.points = 1.4,
+         auto.layout = FALSE, main = "")
+      lines(tpred, fit[[fit.yvars[v]]][, k], col = clCOL[k], lwd = 3)
+      if (v == 1) title(main = paste("Group", k), cex.main = 1.5)
+    }
+  }
```

3.8. Selection of a number of mixture components

The final part of our example analysis shall be devoted to the selection of a number of mixture components using the approaches outlined in Section 2.4.

Penalized expected deviance

For a fitted model, the value of the penalized expected value (15) together with related quantities is available as a vector element `PED` of an object of class `GLMM_MCMClist`:

```
R> print(mod$PED)
```

```
D.expect   p(opt)      PED  wp(opt)     wPED
 14088.3     74.5  14162.8     74.8  14163.1
```

The same values were also seen on top of the output from `print(mod)` in Section 3.6. In the output above, `D.expect` is the estimated value of $\mathsf{E}\{D(\boldsymbol{\theta}) \,|\, \mathbf{Y} = \mathbf{y}\}$ from Equation (15), based on both sampled chains stored in a `GLMM_MCMClist` object `mod`. Further, `p(opt)` and `wp(opt)` are the values of the penalty term p_{opt} estimated by the importance sampling as generally outlined by Plummer (2008). Both `p(opt)` and `wp(opt)` are weighted averages of

certain quantities based on generated importance samples, where `p(opt)` exploits the unity weights whereas `wp(opt)` the weights proposed by Plummer (2008). Nevertheless, as argued by him, for regular models the difference between `p(opt)` and `wp(opt)` becomes vanishingly small (as it is the case in our example) with increasing sample size. Therefore, in many practical situations it does not really matter which method is used to calculate the penalty term p_{opt}. Finally, `PED` and `wPED` are the PED values calculated using Equation (15) while `p(opt)` and `wp(opt)`, respectively, is used as the value of p_{opt}.

Deviance samples

In Section 2.4 we further mentioned that the selection of a number of mixture components can also be based on the posterior distribution of the deviances. The two parallel samples from this posterior distribution which we later use to calculate either the posterior probability (16) or the values of the cdf (18), are available as `Deviance1` and `Deviance2` vector elements of an object of class `GLMM_MCMClist`. For example, the first ten values of the first sample are:

```
R> print(mod$Deviance1[1:10], digits = 9)

 [1] 14086.4445 14095.8681 14113.7117 14097.6682 14109.1799 14102.2900
 [7] 14110.1603 14098.9010 14109.3791 14096.8414
```

Note that the `mod$Deviance1` vector provides the same posterior sample as the `mod[[1]]$Deviance` vector (see Section 3.5) and analogously `mod$Deviance2` vector provides the same sample as the `mod[[2]]$Deviance` vector. Nevertheless, when comparing the output above from that on page 17 where we printed the first ten elements of a vector `mod[[1]]$Deviance`, a negligible difference is seen. This arises from the fact that the Laplace approximation around the mode of the integrand located by the Newton-Raphson algorithm is used to calculate numerically the integrals in (8) needed for the deviance evaluation and the initial values for the Newton-Raphson algorithm are different for calculation of `mod[[1]]$Deviance` and `mod$Deviance1`, respectively. Analogously, the values in `mod$Deviance2` are slightly different from those in `mod[[2]]$Deviance`.

Comparison of models with different numbers of components

To use the penalized expected deviance, or any other characteristic of the fitted model for the selection of a number of mixture components, models with different values of K must be first fitted. This can be done by running the following code where we run the posterior MCMC simulation for models with $K = 1, 2, 3, 4$. To use only a necessary amount of the operating memory we keep from each model only the deviance samples (to be stored in the `lists` `Devs1` and `Devs2`) and the values of the penalized expected deviance and the related quantities (to be stored in a `data.frame` `PED`).

```
R> Devs1 <- Devs2 <- list()
R> PED <- data.frame()
R> for (K in 1:4){
+    cat("Calculating K = ", K, "\n=======================\n", sep="")
+    if (K == 2){
+      modK <- mod
```

```
+    }else{
+      set.seed(20042005 + K)
+      modK <- GLMM_MCMC(y = PBC910[, c("lbili", "platelet", "spiders")],
+         dist = c("gaussian", "poisson(log)", "binomial(logit)"),
+         id = PBC910[, "id"],
+         x = list(lbili    = "empty",
+                  platelet = "empty",
+                  spiders  = PBC910[, "month"]),
+         z = list(lbili    = PBC910[, "month"],
+                  platelet = PBC910[, "month"],
+                  spiders  = "empty"),
+         random.intercept = rep(TRUE, 3),
+         prior.b = list(Kmax = K),
+         nMCMC = c(burn = 100, keep = 1000, thin = 10, info = 100),
+         parallel = FALSE)
+    }
+    Devs1[[K]] <- modK[["Deviance1"]]
+    Devs2[[K]] <- modK[["Deviance2"]]
+    PED <- rbind(PED, modK[["PED"]])
+    colnames(PED) <- names(modK[["PED"]])
+
+    rm(list = "modK")
+  }
```

Consequently, the PED values for the four models are as follows, leading us to conclude that the best model having the lowest value of the penalized expected deviance is that with $K = 2$ mixture components:

```
R> print(PED, digits = 6)

  D.expect   p(opt)     PED  wp(opt)    wPED
1  14242.0  36.0238 14278.0  35.8741 14277.8
2  14088.3  74.5441 14162.8  74.8084 14163.1
3  14057.3 131.5205 14188.8 132.7313 14190.0
4  14032.7 179.1014 14211.8 277.4600 14310.2
```

The values of the posterior probabilities (16) and (17) recommended for the model comparison by Aitkin *et al.* (2009) are easily obtained by using another **mixAK** function `summaryDiff()` applied to the posterior samples of the model deviances. In the following, let $D_K(\boldsymbol{\theta})$ denote the deviance of a model with K mixture components. We first compare models with $K = 2$ and $K = 1$:

```
R> sD21 <- summaryDiff(c(Devs1[[2]], Devs2[[2]]), c(Devs1[[1]], Devs2[[1]]))
R> print(sD21, digits = 4)

$summary
  Mean   2.5%    50%  97.5%
```

```
-153.7 -177.4 -154.2 -129.4

$Pcut
P(diff < -4.39)     P(diff < 0)
              1               1
```

The output first provides posterior summary statistics for the difference between $D_2(\boldsymbol{\theta}) - D_1(\boldsymbol{\theta})$ and then the values of the posterior probabilities (17) and (16), respectively. As both of these probabilities are practically equal to one, it is clear that a model with $K = 2$ is better than that with $K = 1$ and hence there is a strong evidence that at least two clusters are present in the data at hand. Using a simple loop (code not shown), analogous values for comparison of each pair of models were calculated:

```
R> sDiff <- data.frame()
R> for (K1 in 1:3){
+    for (K2 in (K1 + 1):4){
+      tmp1 <- summaryDiff(c(Devs1[[K2]], Devs2[[K2]]),
+                          c(Devs1[[K1]], Devs2[[K1]]))
+      tmp2 <- as.data.frame(matrix(c(tmp1$Pcut, tmp1$summary), nrow = 1))
+      colnames(tmp2) <- c(names(tmp1$Pcut), names(tmp1$summary))
+      rownames(tmp2) <- paste(K2, "-", K1)
```

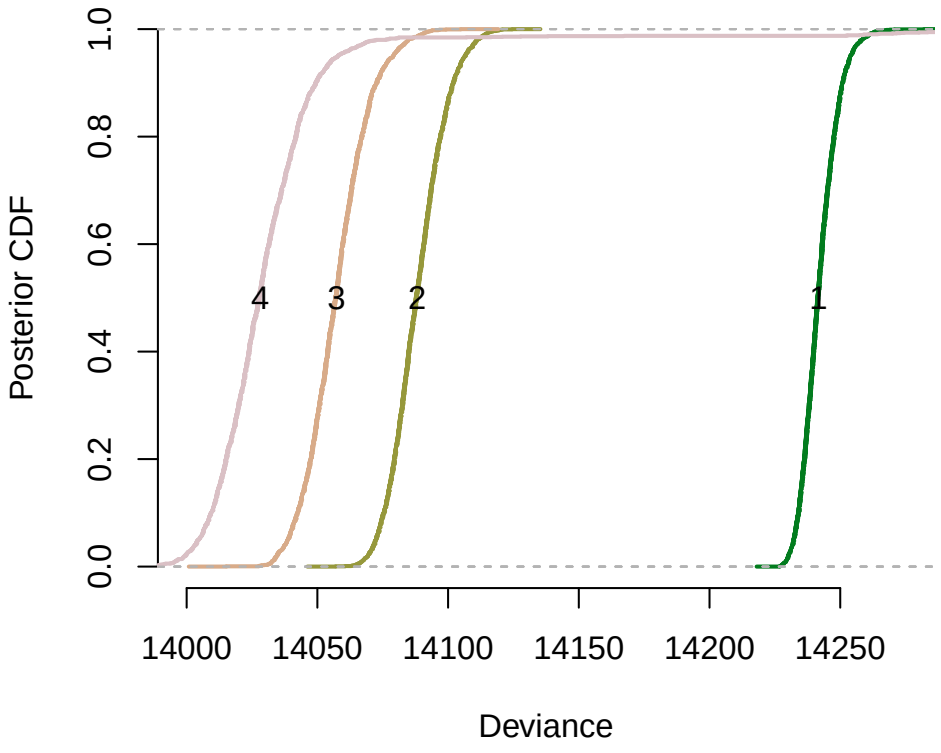

Figure 7: Posterior cumulative distribution functions of the observed data deviances for models with $K = 1, 2, 3, 4$.

```
+      sDiff <- rbind(sDiff, tmp2)
+    }
+  }
R> print(sDiff)

      P(diff < -4.39) P(diff < 0)   Mean   2.5%    50%    97.5%
2 - 1           1.000       1.000 -153.7 -177.4 -154.2 -129.401
3 - 1           1.000       1.000 -184.7 -210.6 -184.8 -155.038
4 - 1           0.988       0.988 -209.3 -245.7 -214.1 -170.617
3 - 2           0.952       0.976  -31.0  -62.7  -31.1   -0.121
4 - 2           0.981       0.982  -55.6  -95.5  -60.3  -13.360
4 - 3           0.884       0.914  -24.6  -66.1  -28.9   16.531
```

If the approach of Aitkin *et al.* (2009) is used for the model selection, then we would choose a model with $K = 3$ components. The posterior probability (17) shown in the first column of the output above is clearly higher than the value of 0.9 advocated by Aitkin *et al.* (2009) to declare model with $K = 3$ components being evidently better that a model with $K = 2$ components. Nevertheless, the value of (17) drops below 0.9 when comparing $K = 4$ to $K = 3$ components.

Finally, the graphical comparison of the cdf's of the deviances of the competing models suggested by Aitkin (2010) is provided by Figure 7 prepared using the following code:

```
R> COL <- terrain_hcl(4, c = c(65, 15), l = c(45, 80), power = c(0.5, 1.5))
R> plot(c(14000, 14275), c(0, 1), type="n",
+       xlab="Deviance", ylab="Posterior CDF")
R> for (K in 1:4){
+    medDEV <- median(c(Devs1[[K]], Devs2[[K]]))
+    ECDF <- ecdf(c(Devs1[[K]], Devs2[[K]]))
+    plot(ECDF, col = COL[K], lwd = 2, add = TRUE)
+    text(medDEV + 0.5, 0.5, labels = K)
+  }
```

Figure 7 shows a huge improvement of the model with respect to its deviance when moving from the $K = 1$ model to the $K = 2$ model. The variability of the posterior distribution of the deviance in a model with $K = 2$ components is practically the same as with $K = 1$. Nevertheless, the $K = 2$ deviance posterior distribution is clearly shifted to left compared to $K = 1$. Almost the same conclusion can be drawn when comparing models with $K = 3$ and $K = 2$ components, and also $K = 4$ and $K = 3$ components.

4. Conclusions and outlook

In particular, on a practical example, this paper provides an overview of capabilities of the **mixAK** package suitable for clustering based on multivariate continuous and discrete longitudinal data. It was not possible to describe here all the options and arguments of the respective functions. Nevertheless, a detailed description of these is available as help pages in the installed package. Additional technical details can also be found in the package vignettes.

Further, this manuscript presents the **mixAK** package mainly as a clustering tool. Nevertheless, the package can also be used in situations when purely a regression analysis with multivariate longitudinal outcomes is of interest. In this case, Equation (9) suggests that the (multivariate) mixture GLMM can be considered as a version of the (multivariate) GLMM being robustified against misspecification of the random effects distribution by assuming a normal mixture there rather than conventionally used normal distribution.

The functionality of the **mixAK** package can be extended in the future in several other directions to provide even more flexible clustering tools. First, we may restrict the component covariance matrices $\mathbb{D}_1, \ldots, \mathbb{D}_K$ in (5) to be common as it is often assumed in mixture applications. Other restrictions imposed on the component covariance matrices, e.g., those considered by the R package **HDclassif**, can be thought over as well, of course. This would allow the user to consider a more parsimonious model underlying the clustering procedure. Second, possible additional flexibility of the clustering procedure might be achieved by assuming that not only the distributional components of the random effects vectors of the underlying generalized linear mixed model are cluster specific but that also the fixed effects vectors $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_R$ varies across clusters. Further, a normal distribution in (5) could be replaced by, e.g., the multivariate t-distribution or even more flexible multivariate skew t-distribution (Azzalini and Capitanio 2003) to have a model being able to capture outlying observations.

In this manuscript, we have shown analysis with outcomes of three types (dichotomous, count, and continuous) for which three benchmark GLMM's, mentioned at the end of Section 2.1, were assumed. Currently, those are the only GLMM's implemented by the **mixAK** package. One important type of outcome, namely a genuine categorical (multinomial) response, is thus not yet covered by the package. A possible step towards the ability to include also the multinomial response could consist of incorporating a mixed-effects version of any of routinely used regression models for such type of outcome in assumption (A1), e.g., the baseline-category logit model (Agresti 2002, Chapter 7). Nevertheless, this goes far beyond the scope of this paper.

Finally, it is worth mentioning that the methods implemented in the **mixAK** package were primarily developed having classical (bio)statistical applications in mind. Notably, even though multivariate outcomes ($R > 1$) are considered, the presented methods and their **mixAK** implementation, if not properly adjusted, are likely not suitable for the analysis of ultradimensional outcomes (R being huge) that are often encountered, e.g., in the context of bioinformatics.

Computational details

The output shown in this article was obtained using R version 3.1.1 (2014-07-10), **mixAK** 3.8, and the following contributed packages which are the dependencies or imports of the package **mixAK**: **colorspace** 1.2-4 (Ihaka, Murrell, Hornik, Fisher, and Zeileis 2013; Zeileis, Hornik, and Murrell 2009), **lme4** 1.1-7 (Bates, Maechler, Bolker, and Walker 2014), **fastGHQuad** 0.1-1 (Blocker 2011), **mnormt** 1.5-1 (Azzalini and Genz 2014). Finally, routines from **coda** 0.16-1 (Plummer *et al.* 2006) were used in this manuscript.

Acknowledgments

We would like to thank Mr. Steven Riley for his help with English corrections of the manuscript. Last but not the least, we thank two anonymous referees and an Associate Editor for very detailed and stimulative comments to the earlier versions of this manuscript that led to its considerable improvement.

Financial support of the first author from the Interuniversity Attraction Poles Programme (IAP-network P7/06), Belgian Science Policy Office, is gratefully acknowledged. The second author was supported by the Czech Science Foundation grant GAČR P403/12/1557.

This document was prepared using **Sweave** (Leisch 2002).

References

Agresti A (2002). *Categorical Data Analysis*. Second edition. John Wiley & Sons, Hoboken. ISBN 0-471-36093-7.

Aitkin M (2010). *Statistical Inference: An Integrated Bayesian/Likelihood Approach*. Chapman & Hall/CRC, Boca Raton. ISBN 978-1-4200-9343-8.

Aitkin M, Liu CC, Chadwick T (2009). "Bayesian Model Comparison and Model Averaging for Small-Area Estimation." *The Annals of Applied Statistics*, **3**(1), 199–221. doi:10.1214/08-AOAS205.

Andrews JL, McNicholas PD (2013). ***teigen**: Model-Based Clustering and Classification with the Multivariate t-Distribution*. R package version 2, URL http://CRAN.R-project.org/package=teigen.

Auder B, Lebret R, Iovleff S, Langrognet F (2014). ***Rmixmod**: An Interface for MIXMOD*. R package version 2.0.2, URL http://CRAN.R-project.org/package=Rmixmod.

Azzalini A, Capitanio A (2003). "Distributions Generated by Perturbation of Symmetry with Emphasis on a Multivariate Skew t-Distribution." *Journal of the Royal Statistical Society B*, **65**(2), 367–389. doi:10.1111/1467-9868.00391.

Azzalini A, Genz A (2014). ***mnormt**: The Multivariate Normal and t Distributions*. R package version 1.5-1, URL http://azzalini.stat.unipd.it/SW/Pkg-mnormt.

Bates D, Maechler M, Bolker B, Walker S (2014). ***lme4**: Linear Mixed-Effects Models using S4*. R package version 1.1-7, URL http://CRAN.R-project.org/package=lme4.

Benaglia T, Chauveau D, Hunter DR, Young D (2009). "**mixtools**: An R Package for Analyzing Finite Mixture Models." *Journal of Statistical Software*, **32**(6), 1–29. URL http://www.jstatsoft.org/v32/i06/.

Bergé L, Bouveyron C, Girard S (2012). "**HDclassif**: An R Package for Model-Based Clustering and Discriminant Analysis of High-Dimensional Data." *Journal of Statistical Software*, **46**(6), 1–29. URL http://www.jstatsoft.org/v46/i06/.

Blocker AW (2011). ***fastGHQuad**: Fast Rcpp Implementation of Gauss-Hermite Quadrature*. R package version 0.1-1, URL http://CRAN.R-project.org/package=fastGHQuad.

Bock HH (1996). "Probabilistic Models in Cluster Analysis." *Computational Statistics & Data Analysis*, **23**(1), 5–28. doi:10.1016/0167-9473(96)88919-5.

Celeux G, Forbes F, Robert CP, Titterington DM (2006). "Deviance Information Criteria for Missing Data Models (with Discussion)." *Bayesian Analysis*, **1**(4), 651–706.

Dickson ER, Grambsch PM, Fleming TR, Fisher LD, Langworthy A (1989). "Prognosis in Primary Biliary-Cirrhosis – Model for Decision-Making." *Hepatology*, **10**(1), 1–7. doi:10.1002/hep.1840100102.

Fraley C, Raftery AE (2002). "Model-Based Clustering, Discriminant Analysis, and Density Estimation." *Journal of the American Statistical Association*, **97**(458), 611–631. doi:10.1198/016214502760047131.

Fraley C, Raftery AE, Murphy TB, Scrucca L (2012). "**mclust** Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation Technical Report No. 597." *Technical report*, Department of Statistics, University of Washington.

Gelfand AE, Sahu SK, Carlin BP (1995). "Efficient Parametrisations for Normal Linear Mixed Models." *Biometrika*, **82**(3), 479–499. doi:10.1093/biomet/82.3.479.

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.

Ihaka R, Murrell P, Hornik K, Fisher JC, Zeileis A (2013). ***colorspace**: Color Space Manipulation*. R package version 1.2-4, URL http://CRAN.R-project.org/package=colorspace.

Komárek A (2009). "A New R Package for Bayesian Estimation of Multivariate Normal Mixtures Allowing for Selection of the Number of Components and Interval-Censored Data." *Computational Statistics & Data Analysis*, **53**(12), 3932–3947. doi:10.1016/j.csda.2009.05.006.

Komárek A (2014). ***mixAK**: Multivariate Normal Mixture Models and Mixtures of Generalized Linear Mixed Models Including Model Based Clustering*. R package version 3.8, URL http://CRAN.R-project.org/package=mixAK.

Komárek A, Hansen BE, Kuiper EMM, van Buuren HR, Lesaffre E (2010). "Discriminant Analysis Using a Multivariate Linear Mixed Model with a Normal Mixture in the Random Effects Distribution." *Statistics in Medicine*, **29**(30), 3267–3283. doi:10.1002/sim.3849.

Komárek A, Komárková L (2013). "Clustering for Multivariate Continuous and Discrete Longitudinal Data." *The Annals of Applied Statistics*, **7**(1), 177–200. doi:10.1214/12-AOAS580.

Leisch F (2002). "Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), *COMPSTAT 2002 – Proceedings in Computational Statistics*, pp. 575–580. Physica-Verlag, Heidelberg.

Macdonald P, Du J (2012). ***mixdist**: Finite Mixture Distribution Models*. R package version 0.5-4, URL http://CRAN.R-project.org/package=mixdist.

McNicholas PD, Jampani KR, Subedi S (2012). ***longclust**: Model-Based Clustering and Classification for Longitudinal Data*. R package version 1.1, URL http://CRAN.R-project.org/package=longclust.

Plummer M (2008). "Penalized Loss Functions for Bayesian Model Comparison." *Biostatistics*, **9**(3), 523–539. doi:10.1093/biostatistics/kxm049.

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/.

Proust-Lima C, Philipps V, Diakite A, Liquet B (2013). ***lcmm**: Estimation of Latent Class Mixed Models, Joint Latent Class Mixed Models and Mixed Models for Curvilinear Outcomes*. R package version 1.6.3, URL http://CRAN.R-project.org/package=lcmm.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Rubin DB (1976). "Inference and missing data." *Biometrika*, **63**(3), 581–592. doi:10.1093/biomet/63.3.581.

Spiegelhalter DJ, Best NG, Carlin BP, van der Linde A (2002). "Bayesian Measures of Model Complexity and Fit (with Discussion)." *Journal of the Royal Statistical Society B*, **64**(4), 583–639. doi:10.1111/1467-9868.00353.

Stephens M (2000). "Dealing with Label Switching in Mixture Models." *Journal of the Royal Statistical Society B*, **62**(4), 795–809. doi:10.1111/1467-9868.00265.

Zeileis A, Hornik K, Murrell P (2009). "Escaping RGBland: Selecting Colors for Statistical Graphics." *Computational Statistics & Data Analysis*, **53**(9), 3259–3270.

Zhong W, Ma P (2012). ***MFDA**: Model Based Functional Data Analysis*. R package version 1.1-4, URL http://CRAN.R-project.org/package=MFDA.

A. Prior hyperparameters and MCMC initial values

In this appendix, we show more details concerning those elements of the object which results from the call to the `GLMM_MCMC` function related to the assumed prior distribution and the initial values of the model parameters used to start the MCMC simulation. Furthermore, we show how to specify the user-defined values of the prior hyperparameters and the MCMC initial values. Everything will be exemplified on the object `mod` obtained by running the code shown on page 13.

As mentioned in Section 3.4, the `class` of the object `mod` returned by the `GLMM_MCMC()` function is `GLMM_MCMClist`. It is a `list` containing the following elements:

```
R> names(mod)
```

```
 [1] ""                    ""                    "PED"
 [4] "D"                   "popt"                "wpopt"
 [7] "inv.D"               "inv.popt"            "inv.wpopt"
[10] "sumISw"              "Deviance1"           "Deviance2"
[13] "Deviance_repl1_ch1" "Deviance_repl1_ch2" "Deviance_repl2_ch1"
[16] "Deviance_repl2_ch2"
```

The first two (unnamed) components of `mod`, objects `mod[[1]]` and `mod[[2]]` are again `lists`, `class` of which is set to `GLMM_MCMC`, with information pertaining to the first and the second sampled chain, respectively. The names of their elements are listed by the following output (shown for `mod[[1]]` only).

```
R> names(mod[[1]])
```

```
 [1] "iter"                "nMCMC"               "dist"
 [4] "R"                   "p"                   "q"
 [7] "fixed.intercept"     "random.intercept"    "lalpha"
[10] "dimb"                "prior.alpha"         "prior.b"
[13] "prior.eps"           "init.alpha"          "state.first.alpha"
[16] "state.last.alpha"    "prop.accept.alpha"   "init.b"
[19] "state.first.b"       "state.last.b"        "prop.accept.b"
[22] "scale.b"             "freqK_b"             "propK_b"
[25] "init.eps"            "state.first.eps"     "state.last.eps"
[28] "poster.mean.y"       "poster.mean.profile" "poster.mean.w_b"
[31] "poster.mean.mu_b"    "poster.mean.Li_b"    "poster.mean.Sigma_b"
[34] "poster.mean.Q_b"     "poster.comp.prob_u"  "poster.comp.prob_b"
[37] "summ.Deviance"       "summ.alpha"          "summ.b.Mean"
[40] "summ.b.SDCorr"       "summ.sigma_eps"      "Deviance"
[43] "Cond.Deviance"       "K_b"                 "w_b"
[46] "mu_b"                "Li_b"                "Q_b"
[49] "Sigma_b"             "gammaInv_b"          "order_b"
[52] "rank_b"              "mixture_b"           "alpha"
[55] "sigma_eps"           "gammaInv_eps"        "relabel_b"
[58] "Cpar"                "poster.comp.prob"    "comp.prob_b"
[61] "quant.comp.prob_b"   "comp.prob"           "quant.comp.prob"
```

In the following, we explain in more detail some of these elements related to the assumed prior distribution and the initial values for the MCMC, those are in particular:

- `scale.b`, see Section A.1,
- `prior.b`, `prior.alpha`, `prior.eps`, see Section A.2,
- `init.b`, `init.alpha`, `init.eps`, see Section A.3.

At the same time, we show how to change their default values by proper use of the arguments `scale.b`, `prior.b`, `prior.alpha`, `prior.eps`, `init.b`, `init2.b`, `init.alpha`, `init2.alpha`, `init.eps`, `init2.eps` of the `GLMM_MCMC` function.

A.1. Scaling of random effects distribution during the sampling

With respect to the implemented MCMC algorithm, we start by pointing out that due to possibility of improving the mixing and its numerical stability, the shifted-scaled random effects

$$\mathbf{b}_i^* = \mathbf{S}^{-1}(\mathbf{b}_i - \mathbf{s}), \qquad i = 1, \ldots, N, \tag{25}$$

and corresponding shifted-scaled mixture means and covariance matrices

$$\left.\begin{array}{l} \boldsymbol{\mu}_k^* = \mathbf{S}^{-1}(\boldsymbol{\mu}_k - \mathbf{s}) \\ \mathbf{D}_k^* = \mathbf{S}^{-1}\mathbf{D}_k\mathbf{S}^{-1} \end{array}\right\} k = 1, \ldots, K, \tag{26}$$

are primarily sampled instead of directly sampling the actual values $\mathbf{b}_1, \ldots, \mathbf{b}_N$ of the random effects, and their mixture means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$. Their sampled values are calculated using the inverse relationships from the sampled values of $\mathbf{b}_1^*, \ldots, \mathbf{b}_N^*$, $\boldsymbol{\mu}_1^*, \ldots, \boldsymbol{\mu}_K^*$, and $\mathbf{D}_1^*, \ldots, \mathbf{D}_K^*$. In (25) and (26), $\mathbf{s}$ is a pre-specified shift vector, and $\mathbf{S}$ is a pre-specified diagonal scale matrix. As it is explained in Komárek and Komárková (2013, Appendix A), it is useful to choose the values of $\mathbf{s}$ and $\mathbf{S}$ such that the shifted-scaled random effects $\mathbf{b}_1^*, \ldots, \mathbf{b}_N^*$ have approximately zero means and unit variances. If not specified by the user, the `GLMM_MCMC` function determines automatically such values of $\mathbf{S}$ and $\mathbf{s}$. These are stored in the `scale.b` components of both `mod[[1]]` and `mod[[2]]` and are always the same for both sampled chains.

```
R> print(mod[[1]]$scale.b)
```

```
$shift
[1]  0.31516  0.00765  5.52621 -0.00663 -2.74948

$scale
(Intercept)          z1 (Intercept)          z1 (Intercept)
     0.8645      0.0201      0.3486      0.0157      3.2284
```

```
R> print(mod[[2]]$scale.b)
```

That is,

$$\mathbf{s} = \big(s_1, \ldots, s_5\big)^\top \doteq \big(0.31516,\ 0.00765,\ 5.52621,\ -0.00663,\ -2.74954\big)^\top,$$
$$\mathbf{S} = \mathsf{diag}\big(S_1, \ldots, S_5\big) \doteq \mathsf{diag}\big(0.8645,\ 0.0201,\ 0.3486,\ 0.0157,\ 3.2285\big).$$

The user is able to set his/her own values of the shift vector and the scale matrix by using the `scale.b` argument in the `GLMM_MCMC` function call. For example, setting $\mathbf{s}$ to a vector of zeros and $\mathbf{S}$ to a unit matrix is achieved by

```
scale.b = list(shift = rep(0, 5),
               scale = rep(1, 5))
```

A.2. Prior hyperparameters

We continue in exploration of the resulting object `mod` by checking the particular values of the hyperparameters of the prior distribution which is in full details described in Komárek

and Komárková (2013, Appendix A). In particular, we now explore the elements `prior.b`, `prior.alpha` and `prior.eps` of the objects `mod[[1]]` and `mod[[2]]`. We also introduce the eponymous arguments of the `GLMM_MCMC` function which allow the user to change the default values of the prior hyperparameters.

Element `prior.b`

First, the element `prior.b` of the objects `mod[[1]]` and `mod[[2]]` provides the hyperparameters of the prior distribution for the mixture related parameters which are the mixture weights $\mathbf{w} = \left(w_1, \ldots, w_K\right)^\top$, the shifted-scaled mixture means $\boldsymbol{\mu}_1^*, \ldots, \boldsymbol{\mu}_K^*$, and scaled mixture covariance matrices $\mathbf{D}_1^*, \ldots, \mathbf{D}_K^*$. The number of assumed mixture components, K, is also given here.

```
R> print(mod[[1]]$prior.b)

$Kmax
[1] 2

$priorK
[1] "fixed"

$priormuQ
[1] "independentC"

$delta
[1] 1


$ce
c1 c2
 0  0


$zeta
[1] 6

$hD
(Intercept)     z1  (Intercept)     z1  (Intercept)
      0.278  0.278         0.278  0.278         0.278
```

```
R> print(mod[[2]]$prior.b)

$distribution
[1] "normal"

$lambda
[1] 0

$xi
   m1 m2 m3 m4 m5
j1  0  0  0  0  0
j2  0  0  0  0  0

$D
     m1 m2 m3 m4 m5
j1.1 36  0  0  0  0
j1.2  0 36  0  0  0
j1.3  0  0 36  0  0
j1.4  0  0  0 36  0
j1.5  0  0  0  0 36
j2.1 36  0  0  0  0
j2.2  0 36  0  0  0
j2.3  0  0 36  0  0
j2.4  0  0  0 36  0
j2.5  0  0  0  0 36

$gD
[1] 0.2 0.2 0.2 0.2 0.2
```

```
$gdf
[1] 1

$hdf
[1] 0.005
```

Some parts of the above `lists`, namely `distribution`, `priorK`, `lambda`, `ce`, `gdf`, `hdf` are redundant in this situation and are included in the object only for compatibility with other related functions from the **mixAK** package. Component `priormuQ` informs us that a semiconjugate independent Normal and Wishart prior, see Komárek and Komárková (2013, Appendices A.8 and A.9) is assumed for the mixture means and the inverted covariance matrices. Alternatively, a natural-conjugate Normal-Wishart prior (see Komárek 2009, Section 2.2) can be considered. That is here, the shifted-scaled mixture means $\boldsymbol{\mu}_1^*$ and $\boldsymbol{\mu}_2^*$ are apriori independent and normally distributed, i.e., $\boldsymbol{\mu}_k^* \sim \mathcal{N}(\boldsymbol{\xi}_b, \mathbf{C}_b)$, $k = 1, 2$. The value of the hyperparameter $\boldsymbol{\xi}_b$ (repeated $K = 2$-times) is shown in the rows of the matrix `xi`, the value of the hyperparameter matrix $\mathbf{C}_b$ (again repeated $K = 2$-times), is shown in the blocks of the matrix `D`, i.e.,

$$\boldsymbol{\mu}_k^* \sim \mathcal{N}\big((0, \ldots, 0)^\top, \mathsf{diag}(36, \ldots, 36)\big), \qquad k = 1, 2.$$

Further, the inverted scaled mixture covariance matrices $\mathbf{D}_1^*$ and $\mathbf{D}_2^*$ are apriori independent Wishart distributed, i.e., $\mathbf{D}_k^{*-1} \sim \mathcal{W}(\zeta_b, \boldsymbol{\Xi}_b)$, where $\boldsymbol{\Xi}_b = \mathsf{diag}(\gamma_{b,1}, \ldots, \gamma_{b,5})$ and $\gamma_{b,l}^{-1} \sim \mathcal{G}(g_{b,l}, h_{b,l})$, $l = 1, \ldots, 5$. The selected value of the hyperparameter ζ_b is reflected by the value of `zeta`, the values of the hyperparameters $g_{b,l}$ and $h_{b,l}$ are shown by components `gD` and `hD`, respectively. That is, for our application,

$$\begin{aligned} \mathbf{D}_k^{*-1} &\sim \mathcal{W}\big(6, \mathsf{diag}(\gamma_{b,1}, \ldots, \gamma_{b,5})\big), \qquad k = 1, 2, \\ \gamma_{b,l}^{-1} &\sim \mathcal{G}(0.2,\, 0.278), \qquad l = 1, \ldots, 5. \end{aligned}$$

Finally, the component `delta` indicates a value of the parameter δ in the Dirichlet prior assumed for the mixture weights (Komárek and Komárková 2013, Appendix A.7). That is, in our application, apriori

$$(w_1,\, w_2) \sim \mathcal{D}(1,\, 1).$$

The user gets a full control over the choice of just mentioned hyperparameters by replacing the `prior.b = list(Kmax = 2)` statement in the `GLMM_MCMC` function call by a more detailed

```
prior.b = list(Kmax = 2, priormuQ = "independentC",
               delta = 1, xi = rep(0, 5), D = diag(rep(36, 5)),
               zeta = 6, gD = rep(0.2, 5), hD = rep(0.278, 5))
```

Element `prior.alpha`

Further, the element `prior.alpha` of the objects `mod[[1]]` and `mod[[2]]` provides the hyperparameters of the prior distribution for the vector of fixed effects $\boldsymbol{\alpha} = \big(\boldsymbol{\alpha}_1^\top, \ldots, \boldsymbol{\alpha}_R^\top\big)^\top$. In our application, only a single fixed effect $\boldsymbol{\alpha} = \alpha_3$ which is the slope from the logit model for the (third) longitudinal variable `spiders`, is included in the model. A normal prior distribution $\mathcal{N}(\boldsymbol{\xi}_\alpha, \mathbf{C}_\alpha)$ with a diagonal covariance matrix $\mathbf{C}_\alpha$ is assumed for (generally a vector) $\boldsymbol{\alpha}$ (Komárek and Komárková 2013, Appendix A.11). We now check particular values of $\boldsymbol{\xi}_\alpha$, $\mathbf{C}_\alpha$.

```
R> print(mod[[1]]$prior.alpha)
```

```
R> print(mod[[2]]$prior.alpha)
```

```
$mean                $var
alpha1.mean          alpha1.var
          0               10000
```

That is, apriori $\alpha_3 \sim \mathcal{N}(0, 10\,000)$. Note that `alpha1` in the output indicates that it corresponds to the first component of in general a vector of the fixed effects $\boldsymbol{\alpha}$. Nevertheless, for notational clarity, subscript 3 was used for α in model (19) to indicate that it pertains to the third marker. The values of the prior mean $\boldsymbol{\xi}_\alpha$ and a diagonal of the prior covariance matrix $\mathbf{C}_\alpha$ might be set by inclusion of the `prior.alpha` argument in the `GLMM_MCMC` function call. For example, the $\mathcal{N}(0, 10\,000)$ prior for α_3 is also achieved by using

```
prior.alpha = list(mean = 0, var = 10000)
```

Element `prior.eps`

Finally, the element `prior.eps` of the objects `mod[[1]]` and `mod[[2]]` provides the hyperparameters of the prior distribution for the subvector of a vector of dispersion parameters $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_R)^\top$ which are unknown and not fixed by the choice of a particular exponential family distribution. In our example, the only unknown dispersion parameter is ϕ_1 which is the residual variance of the linear mixed model assumed for the (first) longitudinal variable `lbili`. It is in general apriori assumed that different dispersion parameters are independent following an inverse Gamma distribution with a random scale parameter following a Gamma hyperprior, see (Komárek and Komárková 2013, Appendix A.10). In our example, $\phi_1^{-1} \sim \mathcal{G}(\zeta_{\phi,1}/2, \gamma_{\phi,1}^{-1}/2)$ where $\gamma_{\phi,1}^{-1}$ is random with a $\mathcal{G}(g_{\phi,1}, h_{\phi,1})$ hyperprior. We examine particular values of the hyperparameters $\zeta_{\phi,1}$, $g_{\phi,1}$, $h_{\phi,1}$ selected by the `GLMM_MCMC` routine using the guidelines given in Komárek and Komárková (2013, Appendix A).

```
R> print(mod[[1]]$prior.eps)
```

```
$zeta      $g       $h
zeta1       g1       h1
    2      0.2     2.76
```

```
R> print(mod[[2]]$prior.eps)
```

That is, $\zeta_{\phi,1} = 2$, $g_{\phi,1} = 0.2$, $h_{\phi,1} = 2.76$. The same values can be explicitly chosen by the user by adding the argument `prior.eps` in the `GLMM_MCMC` function call as

```
prior.eps = list(zeta = 2, g = 0.2, h = 2.76)
```

A.3. Initial values

To start the MCMC simulation, initial values for the model parameters $\boldsymbol{\theta} = (\mathbf{w}^\top, \boldsymbol{\xi}_1^\top, \ldots, \boldsymbol{\xi}_K^\top, \boldsymbol{\xi}^\top)^\top$, for the latent quantities which are the random effect values $\mathbf{b}_1, \ldots, \mathbf{b}_N$, and the component allocations $u_1, \ldots, u_N$ and also for the random hyperparameters $\gamma_{b,1}, \ldots, \gamma_{b,5}$ (see `prior.b` paragraph of Section A.2) and $\gamma_{\phi,1}$ (see `prior.eps` paragraph of Section A.2) must be given. Reasonable initial values are automatically selected by the function `GLMM_MCMC` and are stored as `init.b`, `init.alpha` and `init.eps` components of the objects `mod[[1]]` and `mod[[2]]`, pertaining to the first and the second sampled chain respectively. This automatic

selection is based on the results obtained from fitting separately the R GLMM's from assumption (A1) in Section 2.1 while assuming a classical one-component normal distribution for the random effects using the method of maximum-likelihood (ML) by the mean of the `lmer` or `glmer` functions from the R package **lme4** (Bates *et al.* 2014). In a sequel, let $\boldsymbol{\alpha}^0_{ML}$, $\boldsymbol{\beta}^0_{ML} = \left(\beta^0_{ML,1}, \ldots, \beta^0_{ML,5}\right)^\top$, $\mathbf{d}^0_{ML} = \left(d_{ML,1}, \ldots, d_{ML,5}\right)^\top$, $\sigma^0_{ML,1}$ be vectors of `lmer`/`glmer` ML estimates of the fixed effects, the means of the random effects, the standard deviations of the random effects, and the residual standard deviation (from the Gaussian model for the logarithmic bilirubin), respectively. Further, let $\mathbf{B}^0_{ML}$ be a $N \times 5$ matrix with `lmer`/`glmer` based empirical Bayes estimates of the individual random effects. Below, we explore in more detail the initial values and describe in more detail the mechanism for their default generation.

Element `init.b`

First, the element `init.b` contains the initial values for the mixture related parameters $\mathbf{w}$, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, $\mathbf{D}_1, \ldots, \mathbf{D}_K$, and also for the random effect values $\mathbf{b}_1, \ldots, \mathbf{b}_N$, and the component allocations $u_1, \ldots, u_N$. These for the first chain are seen in R having invoked the following code.

```
R> print(mod[[1]]$init.b)

$b
       b1      b2   b3        b4     b5
1 -0.0458 0.01739 5.38 -0.022628  1.836
2  0.2412 0.01120 5.05 -0.016395 -0.490
3  0.4898 0.01828 5.37  0.004224  1.839
4  0.8811 0.02079 4.89 -0.016732 -0.505
5 -0.2104 0.00407 5.70 -0.000319 -3.751
...

$K         $w           $mu                    $Sigma
[1] 2       w1  w2          m1 m2 m3 m4 m5           m1 m2 m3 m4 m5
           0.5 0.5       j1 -1 -1 -1 -1 -1       j1.1  1  0  0  0  0
                         j2  1  1  1  1  1       j1.2  0  1  0  0  0
                                                 j1.3  0  0  1  0  0
                                                 j1.4  0  0  0  1  0
                                                 j1.5  0  0  0  0  1
                                                 j2.1  1  0  0  0  0
                                                 j2.2  0  1  0  0  0
                                                 j2.3  0  0  1  0  0
                                                 j2.4  0  0  0  1  0
                                                 j2.5  0  0  0  0  1

$Li
Li1.1.1 Li1.2.1 Li1.3.1 Li1.4.1 Li1.5.1 Li1.2.2 Li1.3.2 Li1.4.2 Li1.5.2
      1       0       0       0       0       1       0       0       0
Li1.3.3 Li1.4.3 Li1.5.3 Li1.4.4 Li1.5.4 Li1.5.5 Li2.1.1 Li2.2.1 Li2.3.1
      1       0       0       1       0       1       1       0       0
Li2.4.1 Li2.5.1 Li2.2.2 Li2.3.2 Li2.4.2 Li2.5.2 Li2.3.3 Li2.4.3 Li2.5.3
```

```
      0       0       1       0       0       0       1       0       0
Li2.4.4 Li2.5.4 Li2.5.5
      1       0       1

$gammaInv
gammaInv1 gammaInv2 gammaInv3 gammaInv4 gammaInv5
        6         6         6         6         6

$df
 df1  df2
1000 1000

$r
 r1  r2  r3  r4  r5  r6  r7  r8  r9 r10 r11 r12 r13 r14 r15
  2   1   1   1   1   1   1   1   1   1   1   2   1   1   1
 ...
```

Partially different initial values for the second sampled chain can be seen by invoking (output not shown):

```
R> print(mod[[2]]$init.b)
```

First, `mod[[*]]$init.b$b` is a matrix with initial values of the individual random effects $\mathbf{b}_1, \ldots, \mathbf{b}_N$ in rows. For the first chain (see output above), this is by default equal to the empirical Bayes estimates $\mathbf{B}^0_{ML}$ obtained from the initial `lmer`/`glmer` fits. For the second chain, the initial values of the random effects are the values from the matrix $\mathbf{B}^0_{ML}$ perturbed by normal random variables with a zero mean and the standard deviation equal to the 0.1 multiple of an appropriate element of the vector $\mathbf{d}^0_{ML}$ (random effects standard deviations from the initial `lmer`/`glmer` fits). Namely, the initial values of the random effect vectors in the second chain of the first five subjects are as follows.

```
R> print(mod[[2]]$init.b$b[1:5,])
```

```
       b1      b2   b3       b4     b5
1 -0.1337 0.01788 5.41 -0.02147  1.797
2  0.3134 0.01357 5.03 -0.01637 -0.494
3  0.3802 0.02096 5.39  0.00386  1.659
4  0.9556 0.02134 4.89 -0.01804  0.192
5 -0.0974 0.00384 5.70 -0.00071 -3.665
```

Second, `mod[[*]]$init.b$K` stores the information on the number of mixture components K which is constantly equal to two in our example.

Third, `mod[[*]]$init.b$w` gives the initial values of the mixture weights $\mathbf{w} = \big(w_1, w_2\big)^\top$. For both chains, these are by default all equal to $1/K$. That is, $w_1 = w_2 = 1/2$ in our case.

The initial value for the shifted-scaled mixture means $\boldsymbol{\mu}^*_1$ and $\boldsymbol{\mu}^*_2$ are shown in rows of a matrix `mod[[*]]$init.b$mu`. By default, initials for the lth components, $l = 1, \ldots, 5$, of $\boldsymbol{\mu}^*_1, \ldots, \boldsymbol{\mu}^*_K$

in the first chain are chosen equidistantly on intervals starting and ending at

$$\left.\begin{aligned}\mu_l^{low} &= \frac{\beta^0_{ML,l} - s_l}{S_l} - 3\frac{d^0_{ML,l}}{S_l} + \frac{6d^0_{ML,l}}{(K+1)\,S_l} \\ \mu_l^{upp} &= \frac{\beta^0_{ML,l} - s_l}{S_l} + 3\frac{d^0_{ML,l}}{S_l} - \frac{6d^0_{ML,l}}{(K+1)\,S_l}\end{aligned}\right\} l = 1, \ldots, 5. \tag{27}$$

Due to the fact that in our illustration also the shift vector $\mathbf{s}$ and the scale matrix $\mathbf{S}$ were chosen automatically (see Section A.1) which leads to $\mathbf{s} = \boldsymbol{\beta}^0_{ML}$ and $\mathbf{S} = \mathsf{diag}(\mathbf{d}^0_{ML})$, the initial values of the shifted-scaled mixture means with $K = 2$ are $\boldsymbol{\mu}^*_1 = \big(-1, -1, -1, -1, -1, -1\big)^\top$ and $\boldsymbol{\mu}^*_2 = \big(1, 1, 1, 1, 1, 1\big)^\top$. To create the initial values for the second chain, the same procedure is applied while in (27) replacing for each $l = 1, \ldots, 5$, $\beta^0_{ML,l}$ by a normal $\mathcal{N}\big(\beta^0_{ML,l}, \mathrm{SE}(\beta^0_{ML,l})\big)$ random variable, and replacing $d^0_{ML,l}$ by $d^1_{ML,l} = U \cdot d^0_{ML,l}$, where U is a random variable with the uniform distribution on interval (0.9, 1.1). This leads to the following initial values of the shifted-scaled mixture means for the second chain:

```
R> print(mod[[2]]$init.b$mu)
```

```
      m1     m2     m3    m4    m5
j1 -1.10 -0.942 -1.108 -1.01 -1.15
j2  1.01  1.054  0.985  1.04  1.01
```

The initial values for the scaled mixture covariance matrices $\mathbf{D}^*_1$ and $\mathbf{D}^*_2$ are shown as blocks of `mod[[*]]$init.b$Sigma`. By default, for the first chain, the initial values are

$$\mathbf{D}^*_k = \mathsf{diag}\big((d^0_{ML,1}/S_1)^2, \ldots, (d^0_{ML,5}/S_5)^2\big), \qquad k = 1, \ldots, K, \tag{28}$$

which in our case leads to $\mathbf{D}^*_1 = \mathbf{D}^*_2 = I_5$. To get the default initial values for the second chain, values of $d^0_{ML,l}$, $l = 1, \ldots, 5$ in (28) are replaced by $d^1_{ML,l}$ which leads to $\mathbf{D}^*_1 = \mathbf{D}^*_2 = \mathsf{diag}(1.11, 0.996, 1.10, 1.05, 1.16)$. We can see this in R using

```
R> print(mod[[2]]$init.b$Sigma)
```

```
       m1    m2  m3   m4   m5
j1.1 1.11 0.000 0.0 0.00 0.00
j1.2 0.00 0.996 0.0 0.00 0.00
j1.3 0.00 0.000 1.1 0.00 0.00
j1.4 0.00 0.000 0.0 1.05 0.00
j1.5 0.00 0.000 0.0 0.00 1.16
j2.1 1.11 0.000 0.0 0.00 0.00
j2.2 0.00 0.996 0.0 0.00 0.00
j2.3 0.00 0.000 1.1 0.00 0.00
j2.4 0.00 0.000 0.0 1.05 0.00
j2.5 0.00 0.000 0.0 0.00 1.16
```

Further, `mod[[*]]$init.b$Li` shows the lower triangles of the Cholesky factors of the inverted initial scaled covariance matrices $\mathbf{D}^{*-1}_1$, $\mathbf{D}^{*-1}_2$ stacked in a vector.

Furthermore, `mod[[*]]$init.b$gammaInv` are the initial values for the hyperparameters $\gamma_{b,1}^{-1}, \ldots, \gamma_{b,5}^{-1}$ (see the `prior.b` paragraph of Section A.2). For the first chain, these are by default equal to $\zeta_b \left(d_{ML,l}^0 / S_l\right)^2$, leading to $\gamma_{b,l}^{-1} = 6$, $l = 1, \ldots, 5$ in our application. For the second chain, the initials equal $\zeta_b \left(d_{ML,l}^1 / S_l\right)^2$, $l = 1, \ldots, 5$, leading to $\gamma_{b,1}^{-1} = 6.64$, $\gamma_{b,2}^{-1} = 5.98$, $\gamma_{b,3}^{-1} = 6.57$, $\gamma_{b,4}^{-1} = 6.31$, $\gamma_{b,5}^{-1} = 6.96$, which is seen in R using

```
R> print(mod[[2]]$init.b$gammaInv)

gammaInv1 gammaInv2 gammaInv3 gammaInv4 gammaInv5
     6.64      5.98      6.57      6.31      6.96
```

The value of the element `mod[[*]]$init.b$df` does not have any influence on the calculation and it is included in the object only for compatibility with some other functionality of the **mixAK** package.

Finally, `mod[[*]]$init.b$r` is a vector of the initial values of the component allocations $u_1, \ldots, u_N$. For each subject i, $i = 1, \ldots, N$, the initial value of u_i is by default equal to $g(i)$ for which the density of the conditional distribution $\mathbf{B}_i \,\big|\, U_i = g(i)$, Equation 5, is maximal at the initial values of the remaining parameters.

Element `init.alpha`

The element `mod[[*]]$init.alpha` contains the initial values for the vector of the fixed effects $\boldsymbol{\alpha}$. It is by default equal to $\boldsymbol{\alpha}_{ML}^0$ from the initial `lmer`/`glmer` fits for the first chains, i.e., $\boldsymbol{\alpha} = \alpha_3 = 0.0256$. For the second chain the initial value of each element of the fixed effect vector $\boldsymbol{\alpha}$ is by default equal to the normal random variable with the mean given by the corresponding element of the $\boldsymbol{\alpha}_{ML}^0$ vector and the standard deviation equal to the standard error of the corresponding element of the $\boldsymbol{\alpha}_{ML}^0$ vector, leading to $\boldsymbol{\alpha} = \alpha_3 = 0.0324$ in our application. In R:

```
R> print(mod[[1]]$init.alpha)

alpha1
0.0256
```

```
R> print(mod[[2]]$init.alpha)

alpha1
0.0324
```

Element `init.eps`

Finally, `mod[[*]]$init.eps` holds the initial values for the parameters related to the GLMM dispersion parameters, in our case the residual variance ϕ_1 from the linear mixed model for the first marker `lbili` and its random hyperparameter $\gamma_{\phi,1}^{-1}$ (see `prior.eps` paragraph of Section A.1). The initial value of the residual standard deviation $\sigma_1 = \sqrt{\phi_1}$ from the model for logarithmic bilirubin is for the first chain by default equal to $\sigma_{ML,1}^0$ which is in our application equal to 0.317. The initial of the random hyperparameter $\gamma_{\phi,1}^{-1}$ for the first chain equals by default $\zeta_{\phi,1} \left(\sigma_{ML,1}^0\right)^2$, i.e., $\gamma_{\phi,1}^{-1} = 0.202$. For the second chain, the initial value of the parameter σ_1 equals $U \cdot \sigma_{ML,1}^0$, and the initial value of the hyperparameter $\gamma_{\phi,1}^{-1}$ equals $\zeta_{\phi,1} \left(U \cdot \sigma_{ML,1}^0\right)^2$, where U is the random variable with the uniform distribution on (0.9, 1.1). This leads to the second set of initials being $\sigma_1 = 0.301$, $\gamma_{\phi,1}^{-1} = 0.181$. To see the initial values in R, we invoke

```
R> print(mod[[1]]$init.eps)
$sigma
sigma1
 0.317

$gammaInv
gammaInv1
    0.202
```

```
R> print(mod[[2]]$init.eps)
$sigma
sigma1
 0.301

$gammaInv
gammaInv1
    0.181
```

User-defined initial values

The user is also able to define his/her own initial values for all or a subset of model parameters by defining the `lists` with the same structure as `mod[[1]]$init.b`, `mod[[2]]$init.b`, `mod[[1]]$init.alpha`, `mod[[2]]$init.alpha`, `mod[[1]]$init.eps`, `mod[[2]]$init.eps`, respectively, and using these `lists` as additional arguments `init.b`, `init2.b` (where only either `Sigma` or `Li` component is sufficient to specify the initial values for mixture covariance matrices), `init.alpha`, `init2.alpha`, `init.eps`, `init2.eps` in the call to `GLMM_MCMC` function. At the same time, the user does not have to specify the initial values for all sets of parameters as missing components are initialized using the default procedures described above.

Additionally, the objects `mod[[1]]` and `mod[[2]]` contain components:

- `state.first.b`,
- `state.last.b`,
- `state.first.alpha`,
- `state.last.alpha`,
- `state.first.eps`,
- `state.last.eps`,

which have the same structure as the corresponding `init.*` components (the `state.first.b` and `state.last.b` components contain additionally an element `Q` which holds the lower triangles of the inverted scaled mixture covariance matrices stacked in a vector). Finally, the `state.first.*` components contain the values of the chain at the first saved (after burn-in) MCMC iteration, the `state.last.*` components contain the last saved values of the chain. All of them can be directly supplied as corresponding `init.*` arguments to the `GLMM_MCMC` function when one wishes to re-start the MCMC simulation either from the end of the original burn-in period or from the end of currently finished MCMC simulation. To illustrate this and also the explicit specification of the hyperparameters of the prior distribution, shift vector $\mathbf{s}$ and scale matrix $\mathbf{S}$, we generate a sample of 1 000 values, again with 1:10 thinning, starting from the last MCMC iteration saved in the `mod` object.

```
R> set.seed(20072011)
R> modContinue <- GLMM_MCMC(y = PBC910[, c("lbili", "platelet", "spiders")],
+      dist = c("gaussian", "poisson(log)", "binomial(logit)"),
+      id = PBC910[, "id"],
+      x = list(lbili    = "empty",
+               platelet = "empty",
+               spiders  = PBC910[, "month"]),
+      z = list(lbili    = PBC910[, "month"],
```

```
+               platelet = PBC910[, "month"],
+               spiders  = "empty"),
+      random.intercept = rep(TRUE, 3),
+      scale.b = list(shift = c( 0.31516,  0.00765, 5.52621,
+                                -0.00663, -2.74954),
+                     scale = c(0.8645, 0.0201, 0.3486, 0.0157, 3.2285)),
+      prior.b = list(Kmax = 2, priormuQ = "independentC",
+                     delta = 1, xi = rep(0, 5), D = diag(rep(36, 5)),
+                     zeta = 6, gD = rep(0.2, 5), hD = rep(0.278, 5)),
+      prior.alpha = list(mean = 0, var = 10000),
+      prior.eps = list(zeta = 2, g = 0.2, h = 2.76),
+      init.b  = mod[[1]]$state.last.b,
+      init2.b = mod[[2]]$state.last.b,
+      init.alpha  = mod[[1]]$state.last.alpha,
+      init2.alpha = mod[[2]]$state.last.alpha,
+      init.eps = mod[[1]]$state.last.eps,
+      init2.eps = mod[[2]]$state.last.eps,
+      nMCMC = c(burn = 0, keep = 1000, thin = 10, info = 100),
+      parallel = TRUE)
```

B. Posterior samples

This appendix shows details concerning the storage of the posterior samples in the object from the call to the `GLMM_MCMC` function. Analogously to Appendix A, everything will be exemplified on the object `mod` obtained by running the code shown on page 13. As it is indicated in Section 3.5, the two generated posterior samples of the model parameters and some additional derived quantities are stored as certain elements of `mod[[1]]` (the first sampled chains) and `mod[[2]]` (the second sampled chain): `w_b`, `mu_b`, `Sigma_b`, `Q_b`, `Li_b`, `alpha`, `sigma_eps`, `gammaInv_b`, `gammaInv_eps`, `mixture_b`, `Deviance`, `Cond.Deviance`. All of them are vectors or matrices where each value of each row corresponds to one MCMC iteration and each column to one parameter or an element of a vector parameter. In the reminder of this part, we explore in more detail the values of the first sampled chain which are kept in the `mod[[1]]` object. Getting the values of the second chain from the `mod[[2]]` object is then analogous. In the illustrative code, we shall print in most cases only the first three sampled values stored in the first three rows of the respective matrix.

B.1. Mixture parameters

With respect to the mixture related parameters (mixture weights $\mathbf{w}$, means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$). Remember that $K = 2$ in our example. We first check the element `w_b` holding the sampled values of the mixture weights $\mathbf{w}$:

```
R> print(mod[[1]]$w_b[1:3,])
```

```
        w1    w2
[1,] 0.638 0.362
```

```
[2,] 0.564 0.436
[3,] 0.545 0.455
```

Similarly, we find the element `mu_b` with the sampled values of the shifted-scaled mixture means $\boldsymbol{\mu}_1^*, \ldots, \boldsymbol{\mu}_K^*$:

```
R> print(mod[[1]]$mu_b[1:3,])
```

```
     mu.1.1  mu.1.2 mu.1.3  mu.1.4 mu.1.5 mu.2.1 mu.2.2 mu.2.3  mu.2.4
[1,] -0.663 -0.0600  0.175 -0.0171 -0.330  1.070 -0.221 -0.273 -0.0418
[2,] -0.662 -0.2093  0.192  0.0824 -0.514  0.735  0.329 -0.395  0.2469
[3,] -0.738 -0.0698  0.238  0.0515 -0.334  0.605  0.189 -0.344  0.2224
     mu.2.5
[1,]  0.511
[2,]  0.521
[3,]  0.677
```

The first five columns contain the mean vector $\boldsymbol{\mu}_1^*$ for the first mixture component, the remaining five columns contain the mean vector $\boldsymbol{\mu}_2^*$ of the second mixture component. Further, the element `Sigma_b` contains the sampled values of the lower triangles of the scaled mixture covariance matrices $\mathbf{D}_1^*, \mathbf{D}_2^*$:

```
R> print(mod[[1]]$Sigma_b[1:3,])
```

```
     Sigma1.1.1 Sigma1.2.1 Sigma1.3.1 Sigma1.4.1 Sigma1.5.1 Sigma1.2.2
[1,]      0.212     0.0584    -0.0751    0.00354      0.273      0.206
[2,]      0.142     0.1112    -0.0582   -0.09732      0.117      0.293
[3,]      0.139     0.0430    -0.0107   -0.06589     -0.136      0.152
     Sigma1.3.2 Sigma1.4.2 Sigma1.5.2 Sigma1.3.3 Sigma1.4.3 Sigma1.5.3
[1,]    0.00142    -0.0795    -0.0144      0.791    -0.0567    -0.0664
[2,]    0.07909    -0.1037     0.1962      0.583    -0.0116     0.1382
[3,]   -0.01655    -0.0190     0.0665      0.704     0.0674     0.1120
     Sigma1.4.4 Sigma1.5.4 Sigma1.5.5 Sigma2.1.1 Sigma2.2.1 Sigma2.3.1
[1,]      0.531      0.224      1.244      0.624    -0.0831      0.277
[2,]      0.405     -0.117      1.207      0.676    -0.1997      0.379
[3,]      0.353      0.166      0.887      0.682     0.1028      0.094
     Sigma2.4.1 Sigma2.5.1 Sigma2.2.2 Sigma2.3.2 Sigma2.4.2 Sigma2.5.2
[1,]  -0.000169   -0.00384       2.38    -0.0653     0.0559      0.248
[2,]  -0.081421   -0.00740       2.75    -0.0554     0.3833     -0.141
[3,]  -0.203883    0.14186       2.03    -0.2473     0.1260     -0.121
     Sigma2.3.3 Sigma2.4.3 Sigma2.5.3 Sigma2.4.4 Sigma2.5.4 Sigma2.5.5
[1,]       1.77     0.2375     -0.227       2.20     -0.271      0.621
[2,]       1.73    -0.0879     -0.340       1.74     -0.171      0.897
[3,]       1.19    -0.2495      0.191       1.92     -0.424      0.795
```

The first 15 columns comprise the lower triangles of the sampled matrices $\mathbf{D}_1^*$, the remaining 15 columns comprise the lower triangles of the sampled matrices $\mathbf{D}_2^*$. Similarly, elements

`Q_b` and `Li_b` (not printed) contain the sampled values of the lower triangles of the inverted mixture covariance matrices $\mathbf{D}_1^{*-1}, \mathbf{D}_2^{*-1}$ and their Cholesky factors, respectively. All above mentioned mixture related parameters are saved as sampled, i.e., without applying any relabeling algorithm.

To get the (relabeled) samples of the mixture parameters (on the original – data – scale) and possibly some derivatives like standard deviations and correlation coefficients based on the mixture covariance matrices, a function `NMixChainComp()` can be used. The chains with relabeled mixture weights are obtained using:

```
R> wSamp <- NMixChainComp(mod[[1]], relabel = TRUE, param = "w_b")
R> print(wSamp[1:3,])

        w1    w2
[1,] 0.638 0.362
[2,] 0.564 0.436
[3,] 0.545 0.455
```

Setting the argument `param` to a value of `"mu_b"` provides the (relabeled) sample of the mixture means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$:

```
R> muSamp <- NMixChainComp(mod[[1]], relabel = TRUE, param = "mu_b")
R> print(muSamp[1:3,])

      mu1.1   mu1.2 mu1.3    mu1.4 mu1.5 mu2.1   mu2.2 mu2.3    mu2.4
[1,] -0.258 0.00645  5.59 -0.00690 -3.82 1.240 0.00322  5.43 -0.00729
[2,] -0.257 0.00345  5.59 -0.00534 -4.41 0.951 0.01426  5.39 -0.00277
[3,] -0.323 0.00625  5.61 -0.00583 -3.83 0.838 0.01144  5.41 -0.00315
      mu2.5
[1,] -1.101
[2,] -1.069
[3,] -0.563
```

Analogously, the `param` values of `"var_b"`, `"sd_b"` and `"cor_b"` lead to the relabeled samples of mixture variances, standard deviations and correlations, respectively, derived from the mixture covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$:

```
R> varSamp <- NMixChainComp(mod[[1]], relabel = TRUE, param = "var_b")
R> print(varSamp[1:3,])

     var1.1   var1.2 var1.3   var1.4 var1.5 var2.1   var2.2 var2.3
[1,]  0.158 8.30e-05 0.0962 1.30e-04  12.97  0.466 0.000958  0.215
[2,]  0.106 1.18e-04 0.0708 9.92e-05  12.58  0.505 0.001109  0.211
[3,]  0.104 6.13e-05 0.0855 8.66e-05   9.24  0.510 0.000819  0.145
       var2.4 var2.5
[1,] 0.000539   6.47
[2,] 0.000425   9.35
[3,] 0.000470   8.28
```

```
R> sdSamp <- NMixChainComp(mod[[1]], relabel = TRUE, param = "sd_b")
R> print(sdSamp[1:3,])

     sd1.1   sd1.2 sd1.3   sd1.4 sd1.5 sd2.1  sd2.2 sd2.3  sd2.4 sd2.5
[1,] 0.398 0.00911 0.310 0.01141  3.60 0.683 0.0309 0.463 0.0232  2.54
[2,] 0.326 0.01086 0.266 0.00996  3.55 0.711 0.0333 0.459 0.0206  3.06
[3,] 0.323 0.00783 0.292 0.00930  3.04 0.714 0.0286 0.381 0.0217  2.88


R> corSamp <- NMixChainComp(mod[[1]], relabel = TRUE, param = "cor_b")
R> print(corSamp[1:3,])

     cor1.2.1 cor1.3.1 cor1.4.1 cor1.5.1 cor1.3.2 cor1.4.2 cor1.5.2
[1,]    0.280  -0.1835   0.0106    0.532  0.00352   -0.240  -0.0285
[2,]    0.545  -0.2023  -0.4055    0.283  0.19151   -0.301   0.3300
[3,]    0.295  -0.0341  -0.2969   -0.386 -0.05059   -0.082   0.1810
     cor1.4.3 cor1.5.3 cor1.5.4 cor2.2.1 cor2.3.1  cor2.4.1 cor2.5.1
[1,]  -0.0874  -0.0669    0.276  -0.0683    0.264 -0.000144 -0.00618
[2,]  -0.0238   0.1648   -0.167  -0.1464    0.350 -0.075167 -0.00950
[3,]   0.1351   0.1417    0.297   0.0873    0.104 -0.178338  0.19269
     cor2.3.2 cor2.4.2 cor2.5.2 cor2.4.3 cor2.5.3 cor2.5.4
[1,]  -0.0319   0.0244   0.2038   0.1205   -0.217   -0.232
[2,]  -0.0254   0.1754  -0.0897  -0.0507   -0.272   -0.137
[3,]  -0.1588   0.0639  -0.0951  -0.1649    0.196   -0.343
```

Finally, setting the `param` argument to `"Sigma_b"`, `"Q_b"` and `"Li_b"` provides (relabeled) samples of the lower triangles of the mixture covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$, their inversions $\mathbf{D}_1^{-1}, \ldots, \mathbf{D}_K^{-1}$ and Cholesky decompositions of the inverted mixture covariance matrices $\mathbf{D}_1^{-1}, \ldots, \mathbf{D}_K^{-1}$, respectively. As illustration, we create a relabeled sample of the mixture covariance matrices $\mathbf{D}_1, \ldots, \mathbf{D}_K$ and print the first three values.

```
R> DSamp <- NMixChainComp(mod[[1]], relabel = TRUE, param = "Sigma_b")
R> print(DSamp[1:3,])

     Sigma1.1.1 Sigma1.2.1 Sigma1.3.1 Sigma1.4.1 Sigma1.5.1 Sigma1.2.2
[1,]      0.158   0.001014   -0.02262   0.000048      0.762   8.30e-05
[2,]      0.106   0.001931   -0.01755  -0.001317      0.327   1.18e-04
[3,]      0.104   0.000746   -0.00322  -0.000892     -0.379   6.13e-05
     Sigma1.3.2 Sigma1.4.2 Sigma1.5.2 Sigma1.3.3 Sigma1.4.3 Sigma1.5.3
[1,]   9.96e-06  -2.50e-05  -0.000935     0.0962  -0.000309    -0.0747
[2,]   5.53e-04  -3.26e-05   0.012715     0.0708  -0.000063     0.1555
[3,]  -1.16e-04  -5.97e-06   0.004307     0.0855   0.000368     0.1260
     Sigma1.4.4 Sigma1.5.4 Sigma1.5.5 Sigma2.1.1 Sigma2.2.1 Sigma2.3.1
[1,]   1.30e-04    0.01134      12.97      0.466   -0.00144     0.0836
[2,]   9.92e-05   -0.00590      12.58      0.505   -0.00347     0.1141
[3,]   8.66e-05    0.00839       9.24      0.510    0.00178     0.0283
     Sigma2.4.1 Sigma2.5.1 Sigma2.2.2 Sigma2.3.2 Sigma2.4.2 Sigma2.5.2
```

```
[1,]  -2.29e-06    -0.0107   0.000958  -0.000457   1.76e-05    0.01604
[2,]  -1.10e-03    -0.0206   0.001109  -0.000388   1.20e-04   -0.00913
[3,]  -2.76e-03     0.3959   0.000819  -0.001731   3.96e-05   -0.00784
     Sigma2.3.3 Sigma2.4.3 Sigma2.5.3 Sigma2.4.4 Sigma2.5.4 Sigma2.5.5
[1,]      0.215    0.00130     -0.256   0.000539   -0.01369       6.47
[2,]      0.211   -0.00048     -0.382   0.000425   -0.00864       9.35
[3,]      0.145   -0.00136      0.215   0.000470   -0.02141       8.28
```

B.2. GLMM related parameters

The sampled values of the GLMM related parameters (fixed effects $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_3$ and square roots of dispersion parameters $\phi_1, \ldots, \phi_R$) are kept in the `alpha` and `sigma_eps` elements of `mod[[1]]` and `mod[[2]]` objects. In our application, $\boldsymbol{\alpha} = \alpha_3$ (slope from the logit model for the variable `spiders`) and the first three sampled values from the first chain are as follows.

```
R> print(mod[[1]]$alpha[1:3,])
```

```
[1] 0.00725 0.04123 0.01868
```

Further, $\boldsymbol{\phi} = \phi_1$ is the residual variance from the linear mixed model for the variable `lbili` and the first three sampled values of $\sigma_1 = \sqrt{\phi_1}$ from the first chain are the following.

```
R> print(mod[[1]]$sigma_eps[1:3,])
```

```
[1] 0.320 0.313 0.304
```

B.3. Random hyperparameters

The sampled values of the random hyperparameters $\gamma_{b,1}^{-1}, \ldots, \gamma_{b,5}^{-1}$ introduced in the `prior.b` paragraph of Section A.2 are kept in the component `gammaInv_b` of the `mod[[1]]` and `mod[[2]]` objects. Similarly, the sampled values of the random hyperparameter $\gamma_{\phi,1}^{-1}$ introduced in the `prior.eps` paragraph of Section A.2 are found in the `gammaInv_eps` element (code to print their values not shown).

B.4. Moments of random effects

Element `mixture_b` of the objects `mod[[1]]` and `mod[[2]]` contains the posterior samples of some characteristics of the unconditional distribution of random effects $\mathbf{B}_1, \ldots, \mathbf{B}_N$ which following Equations (1) and (5) is a normal mixture, i.e., for all $i = 1, \ldots, N$:

$$\mathbf{B}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_q(\boldsymbol{\mu}_k, \mathbf{D}_k). \tag{29}$$

The mean and the variance of the distribution (29) are given by Equations (10) and (11) which can also be written in a more detailed form as

$$\boldsymbol{\beta} = \mathsf{E}\big(\mathbf{B}_i;\, \boldsymbol{\theta}\big) = \sum_{k=1}^{K} w_k \boldsymbol{\mu}_k, = \mathbf{s} + \mathbf{S} \sum_{k=1}^{K} w_k \boldsymbol{\mu}_k^*, \tag{30}$$

$$\begin{aligned}
\mathbf{D} &= \mathsf{VAR}\big(\mathbf{B}_i;\, \boldsymbol{\theta}\big) \\
&= \sum_{k=1}^{K} w_k \Big\{ \mathbf{D}_k \,+\, \big(\boldsymbol{\mu}_k - \sum_{j=1}^{K} w_j \boldsymbol{\mu}_j\big)\big(\boldsymbol{\mu}_k - \sum_{j=1}^{K} w_j \boldsymbol{\mu}_j\big)^\top \Big\}, \\
&= \mathbf{S} \Big[\sum_{k=1}^{K} w_k \Big\{ \mathbf{D}_k^* \,+\, \big(\boldsymbol{\mu}_k^* - \sum_{j=1}^{K} w_j \boldsymbol{\mu}_j^*\big)\big(\boldsymbol{\mu}_k^* - \sum_{j=1}^{K} w_j \boldsymbol{\mu}_j^*\big)^\top \Big\} \Big] \mathbf{S}^\top.
\end{aligned} \tag{31}$$

Note that the vector $\boldsymbol{\beta}$ express the mean effect (in a total population) of covariates a subject specific effect of which is expressed by the random effects. The first three sampled values of the first chain stored in the element `mixture_b` of the object `mod[[1]]` are

```
R> print(mod[[1]]$mixture_b[1:3,])
```

```
  b.Mean.1 b.Mean.2 b.Mean.3 b.Mean.4 b.Mean.5 b.SD.1 b.Corr.2.1
1    0.284  0.00528     5.53 -0.00704    -2.83  0.888    -0.0557
2    0.270  0.00817     5.50 -0.00422    -2.95  0.800     0.1448
3    0.206  0.00862     5.52 -0.00461    -2.34  0.789     0.1691
  b.Corr.3.1 b.Corr.4.1 b.Corr.5.1 b.SD.2 b.Corr.3.2 b.Corr.4.2 b.Corr.5.2
1    -0.1132   -0.00702      0.456 0.0200   -0.00558    -0.0279     0.0452
2    -0.0695   -0.03671      0.392 0.0241   -0.04429     0.1094     0.1343
3    -0.1710   -0.07595      0.342 0.0203   -0.15619     0.0553     0.0433
  b.SD.3 b.Corr.4.3 b.Corr.5.3 b.SD.4 b.Corr.5.4 b.SD.5
1  0.380     0.0450   -0.17830 0.0167     0.0349   3.51
2  0.377    -0.0637   -0.17598 0.0156    -0.0857   3.73
3  0.351    -0.0976    0.00163 0.0162    -0.0549   3.38
```

The columns `b.Mean.*` hold a posterior sample for the elements of the vector $\boldsymbol{\beta}$, Equation (30), the columns `b.SD.*` and `b.Corr.*.*` hold posterior samples for standard deviations and correlation coefficients, respectively, derived from the covariance matrix $\mathbf{D}$, Equation (31).

B.5. Model deviance

Finally, also the posterior sample of the observed data model deviance $L(\boldsymbol{\theta})$ (Equation 12) is available as the component `Deviance` of the objects `mod[[1]]` and `mod[[2]]` (first ten values of the deviance sample based on the first chain are printed):

```
R> print(mod[[1]]$Deviance[1:10], digits = 6)
```

```
 [1] 14086.4 14095.9 14113.7 14097.7 14109.2 14102.3 14110.2 14098.9
 [9] 14109.4 14096.8
```

Note that the integral in (8) needed to calculate the deviance at each MCMC iteration cannot be evaluated analytically and the function `GLMM_MCMC` uses the Laplace approximation to calculate numerically its value.

Affiliation:

Arnošt Komárek
Dept. of Probability and Mathematical Statistics
Faculty of Mathematics and Physics
Charles University in Prague
Sokolovská 83
CZ–186 75 Praha 8 – Karlín, Czech Republic
E-mail: arnost.komarek@mff.cuni.cz
URL: http://msekce.karlin.mff.cuni.cz/~komarek/

Lenka Komárková
Dept. of Exact Methods
Faculty of Management
University of Economics in Prague
Jarošovská 1117
CZ–377 01 Jindřichův Hradec, Czech Republic
E-mail: komarkol@fm.vse.cz