robustlmm: An R Package for Robust Estimation of Linear Mixed-Effects Models

Manuel Koller
University of Bern

Abstract

As any real-life data, data modeled by linear mixed-effects models often contain outliers or other contamination. Even little contamination can drive the classic estimates far away from what they would be without the contamination. At the same time, datasets that require mixed-effects modeling are often complex and large. This makes it difficult to spot contamination. Robust estimation methods aim to solve both problems: to provide estimates where contamination has only little influence and to detect and flag contamination.

We introduce an R package, **robustlmm**, to robustly fit linear mixed-effects models using the Robust Scoring Equations estimator. The package's functions and methods are designed to closely equal those offered by **lme4**, the R package that implements classic linear mixed-effects model estimation in R. The robust estimation method in **robustlmm** is based on the random effects contamination model and the central contamination model. Contamination can be detected at all levels of the data. The estimation method does not make any assumption on the data's grouping structure except that the model parameters are estimable. **robustlmm** supports hierarchical and non-hierarchical (e.g., crossed) grouping structures. The robustness of the estimates and their asymptotic efficiency is fully controlled through the function interface. Individual parts (e.g., fixed effects and variance components) can be tuned independently.

In this tutorial, we show how to fit robust linear mixed-effects models using **robustlmm**, how to assess the model fit, how to detect outliers, and how to compare different fits. If you use the software, please cite this article as published in the Journal of Statistic Software (Koller 2016).

Keywords: robust statistics, mixed-effects model, hierarchical model, ANOVA, R, crossed, random effect.

1. Introduction

Linear mixed-effects models are powerful tools to model data with multiple levels of random variation, sometimes called variance components. Data with multiple levels of random variation may have contamination or outliers on any of these levels. To detect and deal with contamination, we developed a method that fits linear mixed-effects models robustly, using the Robust Scoring Equations estimator (Koller and Stahel 2015; Koller 2013). We have implemented the methods in the R-package **robustlmm** (Koller 2015) that we introduce here.

The variability introduced at the random effects level generally affects multiple observations

simultaneously. In a one-way anova dataset, for example, a group level random effect influences the observed value of all the observations that belong to the corresponding group. If this group level random effect were an outlier with respect to the other group levels, this would lead to a whole group of outliers on the level of observations (see, e.g., plate g in Figure 1). When using classic estimation methods, even one such outlier might inflate the between-group variability estimate and distort the results (see example discussed in Section 4). In such a case it would be natural to assume that the group's random effect (or mean) is an outlier rather than all observations are outliers in the same direction. This concept of allowing potential contamination on different sources of variability leads to the "random effects contamination model". With this model, we make the assumption of long-tailed or "gross error" distributions for the random effects as well and not just for the random errors. The effect of the contamination is then propagated via the design matrices to the actual observations.

Levels of random variability can be hierarchical or crossed, or both, depending on the grouping structure in the data. This implies that the effect of a single outlier on the random effects level is not always as straight forward as in the above mentioned one-way anova example. The effect may be different for each observation as the result of an outlier for a single observation is combined with all the other random effects that affect this observation. This complex relationship between the source of contamination and what is effectively realized in the data can make it very hard or even impossible to spot contamination. This is where robust methods step in and help clear the picture.

Basing the robust estimator on the "random effects contamination model" allows not only multiple sources of contamination, it also avoids unnecessary assumptions about the data's grouping structure. The only assumption on the grouping structure, that is also required for classic estimation, is that the model parameters are estimable. Other contamination models usually assume that contamination is introduced and dealt with at the lowest level only – the level of the observations. In mixed-effects models, observations generally correlate with one another, and robust methods must respect these correlations. These dependencies between observations require other contamination models to make strict assumptions about the grouping structure. The random effects contamination model assumes that contamination occurs directly at the source of random variability, before the grouping structure is introduced, thus circumventing the complexity introduced by the data structure and avoids unnecessary assumptions.

Classic estimation of linear mixed-effects models is mainly provided by two functions in R (Table 1). The function `lme` in the R package **nlme** (Pinheiro, Bates, DebRoy, Sarkar, and R Core Team 2016) supports a variety of random effects and error level covariance structures. It is designed for hierarchical data structures, so incorporating crossed random effects is not straightforward. The function `lmer` from the **lme4** package (Bates, Mächler, Bolker, and Walker 2015) is not limited in that respect: it supports arbitrary grouping structures and efficiently deals with large data by making heavy use of memory-saving sparse representations of matrices. Special random effects and error level covariance structures like, e.g., compound symmetry or AR(1) correlation models, are, however, not yet supported. Linear quantile mixed effects estimation is implemented in the `lqmm` function from the **lqmm** package (Geraci

R package	Function	Approach	Details / Assumptions
nlme	`lme`	classic	optimized for nested hierarchical structures; allows special random effects covariance structures
lme4	`lmer`	classic	no assumptions on grouping structure; correlated and uncorrelated random effects within levels
lqmm	`lqmm`	quantile-based	allows median-type estimates; one grouping level with or without correlation between random effects
heavy	`heavyLme`	t distributions	one grouping level; correlated random effects
—	`lmeRob`	reformulation as multivariate problem, then MM-estimation	balanced nested hierarchical structures; uncorrelated random effects within levels
rlme	`rlme`	rank based	unbalanced nested hierarchical structures (2 or 3 levels); random intercepts only; does not support balanced data
robustlmm	`rlmer`	huberization of likelihood and DAS-Scale estimation	no assumptions on grouping structure; correlated and uncorrelated random effects within levels

Table 1: Overview of classic and robust estimation methods available in R. See also the CRAN Task View on robust statistical methods (Mächler 2016).

2014). This is not a robust method per se, but allows for median-based estimation. The function supports only one grouping level but allows the correlation structure of the random effects to be specified.

For robust estimation of linear mixed-effects models, there exists a variety of specialized implementations in R, all using different approaches to the robustness problem. Most of them are available on the Comprehensive R Archive Network (CRAN) as R packages. Except the method presented in this paper, all other methods are applicable only for certain grouping structures, see Table 1 for an overview. The function `heavyLme` in the **heavy** package (Osorio 2016) implements mixed-effects models using t distributions. However, it allows for a single grouping factor only, which limits the method to two-level data. As both, the residual errors and the random effects are modeled with a t distribution, the method can capture outliers on both the subject and the observational level. The degrees of freedom for the two t distributions are fixed to be the same. Hence, it is not possible to have a differing treatment of outliers on the two levels. Multiple random effects are fitted with a correlation parameter, uncorrelated random effects are not supported. The function `lmeRob` implements the method by Copt and Victoria-Feser (2006). It is not available on CRAN but from the authors upon request. They reformulate the mixed-effects problem as multivariate problem and apply multivariate MM-estimation. This approach requires the grouping structure to be nested and the data to be

balanced. Observations are down-weighted at the highest group level, so the high breakdown point of 50% applies to the number of groups that can be contaminated, not to the number of observations. The implementation only supports uncorrelated random effects within levels. The function `rlme` in the **rlme** R package implements nested hierarchical mixed-effects models using a rank-based approach (Bilgic, Susmann, and McKean 2014). The function supports only simple random intercepts, and solutions might not be unique.

This article is a tutorial for **robustlmm**, an implementation of the Robust Scoring Equations estimator to fit mixed-effects models for the statistical computing environment R (R Core Team 2016). The R package **robustlmm** is available on CRAN at `https://cran.r-project.org/package=robustlmm` under the GPL-v2 license.

In the next section we provide background on **robustlmm**'s underlying estimating equations and algorithms. In Section 3, we describe how **robustlmm** is implemented. In Section 4, we work an example and demonstrate how to do a full statistical analysis. Pointers to further information are given in Section 5. Details, tables of tuning parameters and formulas are contained in the Appendix.

2. Background

2.1. Model equations and assumptions

We work with the general linear mixed-effects model in matrix form and, following Bates (2010), with spherical random effects. The spherical random effects are obtained from the regular random effects by a transformation such that they have a covariance matrix that equals a scaled identity matrix. This transformation enables variance components to be estimated as exactly zero. The model equations are:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{U}_b(\boldsymbol{\theta})\boldsymbol{b}^* + \boldsymbol{U}_e\boldsymbol{\varepsilon}^* \,, \\ \boldsymbol{b}^* \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2\boldsymbol{I}_q\right) \,, &\quad \boldsymbol{\varepsilon}^* \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2\boldsymbol{I}_n\right) \,, \quad \boldsymbol{b}^* \perp \boldsymbol{\varepsilon}^* \,, \end{aligned} \tag{1}$$

where $\boldsymbol{y}$ is the response vector of length n, $\boldsymbol{\beta}$ is the fixed effects vector of length p with design matrix $\boldsymbol{X}$, and $\boldsymbol{b}^*$ is the spherical random effects vector of length q with design matrix $\boldsymbol{Z}$. The relation between the regular and the spherical random effects is $\boldsymbol{b} = \boldsymbol{U}_b(\boldsymbol{\theta})\boldsymbol{b}^*$. The lower triangular matrix $\boldsymbol{U}_b(\boldsymbol{\theta})$ is parameterized by the vector $\boldsymbol{\theta}$. The covariance matrix of the random effects is $\boldsymbol{V}_b(\boldsymbol{\theta}) = \boldsymbol{U}_b(\boldsymbol{\theta})\boldsymbol{U}_b(\boldsymbol{\theta})^\top$. The matrix $\boldsymbol{U}_e$ is assumed to be a diagonal matrix of known weights.

As mentioned in the introduction, we do not assume anything about the structure of the data (i.e., the design matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$), though we do make the usual assumption which the model parameters are estimable. We do assume the covariance matrix of the random effects $\boldsymbol{V}_b(\boldsymbol{\theta})$ to be block-diagonal. This assumption excludes problems that cannot be written in block-diagonal form, like geostatistical problems with spatial dependence encoded in $\boldsymbol{V}_b(\boldsymbol{\theta})$ (see the **georob** package (Papritz 2016) for robust methods to deal with this special case).

To reduce the complexity of the algorithms, our implementation makes additional assumptions about the covariance matrices of the random effects and the residual errors that are not required by the theory *per se*. Blocks of $\boldsymbol{V}_b(\boldsymbol{\theta})$ of size 2×2 and larger are assumed to be unstructured, i.e., unconstrained covariance matrices (other structures such as *compound symmetry* are not supported). In the remainder of the text, we will call blocks of size 1×1 *diagonal* and larger blocks *unstructured*. Finally, the residual error covariance matrix is assumed to be a diagonal matrix with only one unknown scaling parameter.

2.2. Robustness approach

Robustness is achieved by robustification of the scoring equations. The scoring equations are the derivatives of the log-likelihood. To fit the model (1), either the log-likelihood can be maximized, or the roots of the scoring equations can be found. Robust estimating equations are derived from the scoring equations by replacing the residuals and predicted spherical random effects with bounded functions. These bounded functions ensure that a single term (error or random effect) only has bounded influence on the estimating equations. To get robust and efficient estimating equations of σ and $\boldsymbol{\theta}$, we apply the Design Adaptive Scale approach by Koller and Stahel (2011). The robust estimating equations are provided in Appendix B. A detailed derivation and evaluation of the robust method is given in Koller (2013) and Koller and Stahel (2015).

The robustified estimating equations no longer correspond to any likelihood or pseudo-likelihood. Thus, information criteria like AIC and tests based on the log-likelihood statistic are unavailable for the robust method we present here.

2.3. Weighting functions, robustness weights and tuning

Tuning (adjusting robustness properties of the resulting estimates) is done by adjusting parameters that control the form of the bounded functions in the robust estimating equations. In M-estimation terminology, these bounded functions are called *ψ-functions*. They are the derivatives of a *ρ-function* (see Maronna, Martin, and Yohai (2006) for exact definitions). The Huber function, a function that is quadratic around zero and linear for values outside $\pm k$, is a ρ-function (the corresponding ψ-function is shown in Figure 3). The parameter k is called the *tuning parameter*. Larger values yield more efficient, but less robust estimates (for $k = \infty$ one recovers the REML-estimates), whereas smaller values yield more robust but less efficient estimates. A popular choice is to fix the asymptotic efficiency at 95% of the classic estimates ($k = 1.345$ for the Huber function).

Replacing terms by bounded functions thereof down-weights terms with a large absolute value. In the robustness literature, these weights are called *robustness weights*. Observations or random effects with low robustness weights are classified as outliers by the robust method. For a given ψ-function, the robustness weights are defined as

$$w_{\cdot}(v) = \begin{cases} \psi_{\cdot}(v)/x & \text{if } x \neq 0\,, \\ 1 & \text{if } x = 0\,. \end{cases} \tag{2}$$

where we replace the . in $w_.$ and $\psi_.$ by e or b to specify the terms to which the functions are applied (e for errors/residuals; b for random effects). To gain robustness for all estimates, estimating equations for covariance parameters have to be treated differently from fixed and random effects, although the weighting functions for similar terms are related. We therefore distinguish the weighting functions used for estimating σ and $\boldsymbol{\theta}$ with a superscript $^{(\sigma)}$ in equations. (In **robustlmm**, the functions are objects of class `psi_func`. The arguments are called `rho.e`, `rho.b`, `rho.sigma.e` and `rho.sigma.b`.)

The robustness weights defined in (2) yield robust estimates of the fixed effects and predicted values for the random effects for all ρ-functions with a bounded derivative, and also for convex ρ-functions like the Huber function. For estimates of scaling factors (σ and $\boldsymbol{\theta}$ for diagonal-only blocks of $\boldsymbol{V}_b(\boldsymbol{\theta})$), the requirements to get robust estimates are more strict. These are not robust when convex ρ-functions are used. To get robust estimates for scaling factors, we need to use ρ-functions so that $w_.^{(\sigma)}(v)v^2$ is bounded for $v \to \pm\infty$. When convex ρ-functions are used to estimate the fixed and random effects, a natural choice for a ρ-function to estimate the scaling factors is the one that corresponds to the squared robustness weights, i.e.,

$$w_.^{(\sigma)}(v) = w_.(v)^2 \,. \tag{3}$$

Note the similarity to Huber's Proposal 2. (The function `psi2propII` can be used to transform a ρ-function to the corresponding ρ-function that yields squared robustness weights.)

Squared robustness weights are not required for block-diagonal parts of $\boldsymbol{V}_b(\boldsymbol{\theta})$. Instead of M-scale type estimating equations, the unstructured blocks require methods similar to multivariate M-estimators for estimating covariance matrices. Multivariate M-estimators, as introduced by Stahel (1987), use a derived set of ψ-functions that also yield bounded influence estimates for convex ρ-functions. (This derivation is handled internally in **robustlmm**.)

The use of different ρ-functions in the estimating equations for σ and $\boldsymbol{\theta}$ ensures the resulting estimates to be robust, but lowers the efficiency of the estimates $\widehat{\sigma}$ and $\widehat{\boldsymbol{\theta}}$. This might be acceptable for problems in which the scale parameter σ is considered a nuisance parameter, but in mixed-effects modeling one is usually interested in estimating the variance components and does not regard them as nuisance terms. If desired, the efficiency of the estimates of σ and $\boldsymbol{\theta}$ can be increased by increasing the tuning parameters of $\psi_e^{(\sigma)}$ and $\psi_b^{(\sigma)}$. Tables of tuning parameters for popular ψ-functions are provided in the Appendix.

2.4. Estimation algorithms

The models are fit with a nested iterative reweighting algorithm. If there are no initial estimates, then the classic estimates are used as initial estimates. The outer loop is updating $\widehat{\boldsymbol{\theta}}$ until it converges. For each new value of $\widehat{\boldsymbol{\theta}}$, we update $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{b}}^*$ and then $\widehat{\sigma}$. This algorithm converges to a local solution of the estimating equations. For convex ρ-functions and squared robustness weights, the solution can be expected to be unique aside from pathological, easily discarded solutions. A detailed description of the algorithm is given in Appendix C.

3. Implementation

The **robustlmm** package is built upon the **lme4** package, more specifically the `lmer` function. The structure of the objects and the methods are implemented to be as similar as possible to the ones of **lme4** with robustness specific extensions where needed. The object returned by `rlmer` is of class `rlmerMod`. Even though this class is close to the corresponding class `lmerMod` returned by `lmer`, `rlmerMod` does not extend `lmerMod`. This is for two reasons. First, methods for classic estimates are in general not applicable to robust estimates without changes. Second, class inheritance would require a lot of maintenance when the corresponding code in **lme4** is changed. While computational methods of the **lme4** package are implemented in C++, the **robustlmm** package is implemented in pure R.

The main function of the package is `rlmer`, its name hinting at the fact that it is a robust version of the `lmer` function. Besides additional arguments to control the robustness of the fit, the usage of `rlmer` is identical to `lmer`. Most of the functions available for objects returned by `lmer` are also available for objects returned by `rlmer`, e.g., `predict` or `getME`. The `getME` function is a universal accessor function for quantities derived from the fitted object (see `help("getME")`). The function `anova` requires the log-likelihood statistic and is therefore unavailable. The simulation functions `simulate` and `bootMer` have not yet been implemented. The functions to create diagnostic plots, `dotplot`, `plot` and `qqmath` for objects returned by `ranef`, as well as `dotplot` and `plot` for objects returned by `coef`, are available and identical to the those from **lme4**. In addition to the mentioned plot methods, we have added a plot method `plot.rlmerMod` for objects returned by `rlmer` and `lmer`. It creates a Tukey-Anscombe plot, a QQ-plot of the residuals and the random effects as well as scatterplots of the random effects.

4. Usage

4.1. The `Penicillin` data

We illustrate the use of the **robustlmm** package on a dataset originally published by Davies and Goldsmith (1972) and later used by Bates (2010). Davies and Goldsmith (1972) describe it as data coming from an investigation to...

> ...assess the variability between samples of penicillin by the *B. subtilis* method. In this test method a bulk-inoculated nutrient agar medium is poured into a Petri dish of approximately 90 mm. diameter, known as a plate. When the medium has set, six small hollow cylinders or pots (about 4 mm. in diameter) are cemented onto the surface at equally spaced intervals. A few drops of the penicillin solutions to be compared are placed in the respective cylinders, and the whole plate is placed in an incubator for a given time. Penicillin diffuses from the pots into the agar, and this produces a clear circular zone of inhibition of growth of the organisms, which can be readily measured. The diameter of the zone is related in a known way to the concentration of penicillin in the solution.

The description implies that it is a balanced two-way anova dataset with two types of random

effects: *sample* with six levels and *plate* with 24 levels. The random effects are completely crossed. The dataset is distributed as the `Penicillin` dataset in **lme4**.

To emphasize the effect of the robust method, we modified the dataset slightly (as we did in Koller and Stahel (2015)). We scaled down the first plate's observation values, and we moved one observation in plate f down to the lowest original observation. The modified dataset is shown in Figure 1.

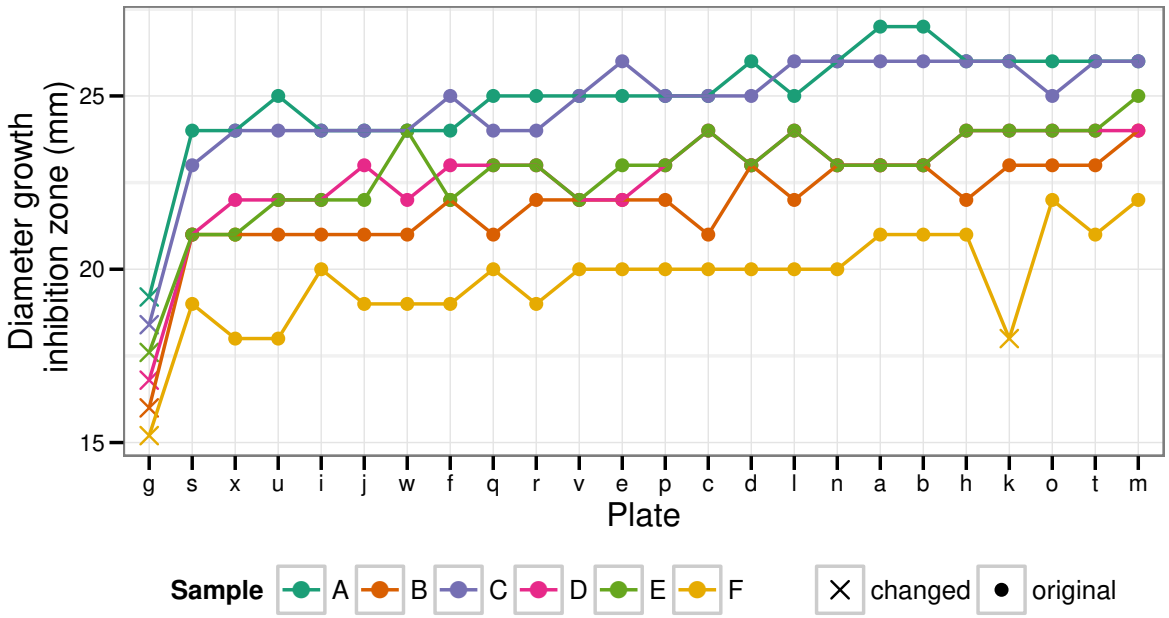

Figure 1: Diameters of growth inhibition zones of 6 samples applied to each of 24 agar plates to assess penicillin concentration. The lines join the observations of the same sample. The plates have been reordered by their diameter values. The observations marked by × have been modified to introduce some contamination.

4.2. Fitting the model and assessing the model fit

We start by loading the R package and the modified `Penicillin` dataset.

```
R> require("robustlmm")
R> warning("Current dir: ", system.file("", package = "robustlmm"), " has contents: ",
+          paste(list.files(system.file("", package = "robustlmm")), collapse = ", "))
R> warning("doc dir: ", system.file("doc", package = "robustlmm"), " has contents: ",
+          paste(list.files(system.file("doc", package = "robustlmm")), collapse = ", "))
R> filename <- system.file("doc/Penicillin.R", package = "robustlmm", mustWork = TRUE)
R> warning("Filename: ", filename)
R> source(filename)
```

The contaminated dataset is now available in the data.frame `PenicillinC`. The dataset con-

sists of four columns: the response `diameter`; two factors `plate` and `sample` that describe the origin of the observations and `contaminated`, which indicates whether or not an observation has been modified.

```
R> str(PenicillinC)
```

```
'data.frame':        144 obs. of  4 variables:
 $ diameter    : num  27 23 26 23 ...
 $ plate       : Factor w/ 24 levels "g","s","x","u",..: 18 18 18 18 ...
 $ sample      : Factor w/ 6 levels "A","B","C","D",..: 1 2 3 4 ...
 $ contaminated: Factor w/ 2 levels "changed","original": 2 2 2 2 ...
```

We fit the model using the function `rlmer`. The name of the function suggests that it is a robust variant of a popular function to fit classic mixed-effects models in R: the function `lmer` of the R package **lme4**. The specification of models is the same for both functions. In a single formula, we specify the fixed and random terms of the model. Fixed terms are added in the usual R formula notation, and random terms are specified in parentheses. Random effects are defined in conjunction with a grouping factor. The grouping factor is separated from the random effect by a pipe symbol "`|`". We now fit the classic and robust models:

```
R> fm  <-  lmer(diameter ~ (1|plate) + (1|sample), PenicillinC)
R> rfm <- rlmer(diameter ~ (1|plate) + (1|sample), PenicillinC)
```

As usual, we get information about the fit using the `summary` function:

```
R> summary(rfm)
```

```
Robust linear mixed model fit by DAStau
Formula: diameter ~ (1 | plate) + (1 | sample)
   Data: PenicillinC

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.8253 -0.6206  0.0811  0.5807  3.2244

Random effects:
 Groups   Name        Variance Std.Dev.
 plate    (Intercept) 0.9209   0.9596
 sample   (Intercept) 4.4921   2.1195
 Residual             0.3282   0.5729
Number of obs: 144, groups: plate, 24; sample, 6

Fixed effects:
```

```
            Estimate Std. Error t value
(Intercept)  22.9970     0.9112   25.24

Robustness weights for the residuals:
 124 weights are ~= 1. The remaining 20 ones are summarized as
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.279   0.693   0.861   0.787   0.911   0.996

Robustness weights for the random effects:
 25 weights are ~= 1. The remaining 5 ones are
    1     2     3    24    30
0.226 0.919 0.994 0.869 0.915

Rho functions used for fitting:
  Residuals:
    eff: smoothed Huber (k = 1.345, s = 10)
    sig: smoothed Huber, Proposal 2 (k = 1.345, s = 10)
  Random Effects, variance component 1 (plate):
    eff: smoothed Huber (k = 1.345, s = 10)
    vcp: smoothed Huber, Proposal 2 (k = 1.345, s = 10)
  Random Effects, variance component 2 (sample):
    eff: smoothed Huber (k = 1.345, s = 10)
    vcp: smoothed Huber, Proposal 2 (k = 1.345, s = 10)
```

The output is close to the output from `lmer`. The first part of the summary provides information about the model fit; the estimates of $\boldsymbol{V}_b(\boldsymbol{\theta})$ and σ; followed by the estimated fixed effects and derived statistics and (if applicable) the correlation of the fixed effects. The second part gives information about the robustness weights and a list of the ρ-functions employed.

As we can read off the summary, the default ρ-function is the smoothed Huber ρ-function, which is a smoothed variant of the regular Huber function (the exact definition is given in the Appendix). The functions for estimating σ and $\boldsymbol{\theta}$ are suffixed by "Proposal 2", indicating that squared robustness weights, with a high robustness but a low efficiency, are used by default (diagonal blocks only, for unstructured blocks the regular ρ-function is used by default).

The summary also tells us that the lowest robustness weights for both the observations and the groups are about 0.2. We can get a full named list of the robustness weights by using the calls `getME(rfm, "w_e")` and `getME(rfm, "w_b")` for the observations and the groups, respectively.

With the call `plot(rfm)`, we can create simple residual analysis plots including normal QQ-plots of the predicted random effects. The resulting plots are shown in Figure 2. The darker color indicates observations with a low robustness weight $w_.$. From the plot as well as from the summary shown above, we can see that the observations that were changed have been detected by the method, i.e., they have received a low robustness weight. There are also

other observations that were assigned a rather low robustness weight and we would do good to investigate them.

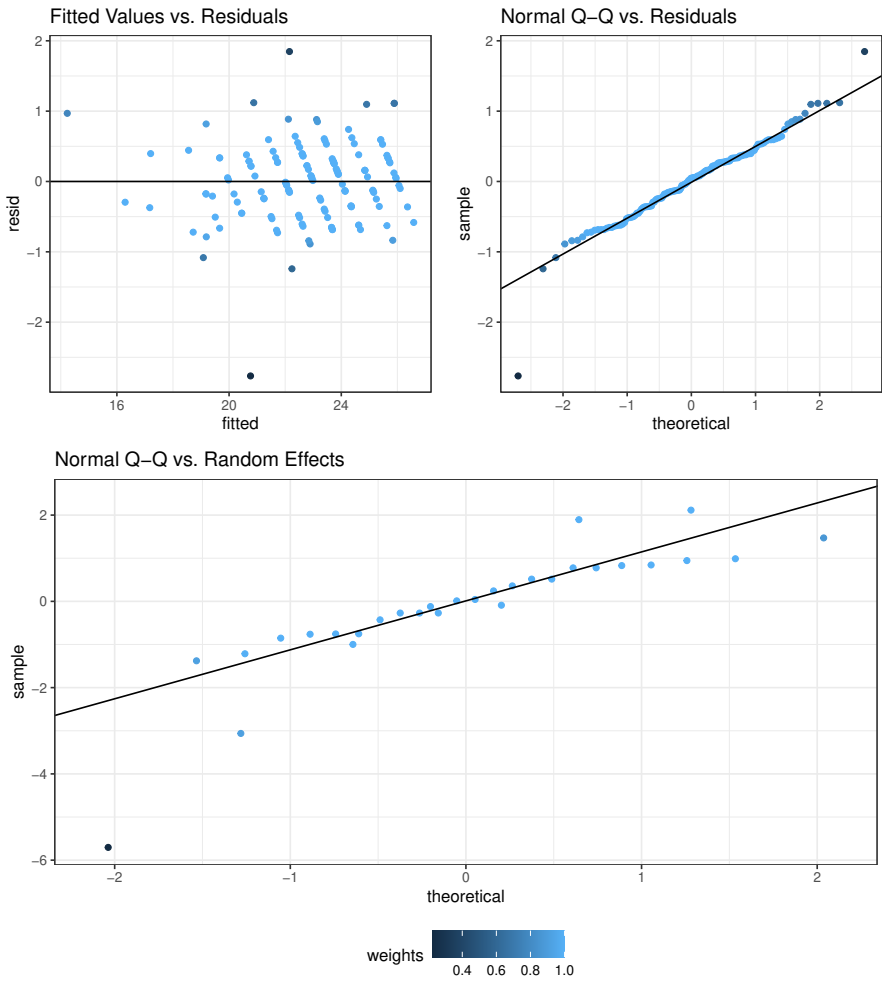

Figure 2: Residual analysis plots for robust fit `rfm`. The coloring of the points gives information about the corresponding robustness weights.

4.3. Tuning the fit

As already mentioned in Section 2.3, the estimates of σ and $\boldsymbol{\theta}$ have a low efficiency if the same tuning parameters are used as for estimating the fixed and random effects. To get a higher efficiency, we have to increase the tuning parameter of $\rho_e^{(\sigma)}$ and $\rho_b^{(\sigma)}$. Tables of tuning parameters are provided in the Appendix. We can change the default tuning parameter of objects of class `psi_func` with the function `chgDefaults`. The smoothed Huber function is a convex ρ-function. Hence, we need to use squared robustness weights to get robust estimates of the residual error scale and the variance components (see Section 2.3). We can convert any ρ-function to the corresponding one with squared robustness weights using the function `psi2propII`. The latter function also allows the tuning parameters to be changed

simultaneously, so that a second call to `chgDefaults` can be saved. We use the `update` function to fit a model with a higher efficiency for the estimates of σ and $\boldsymbol{\theta}$:

```
R> rfm2 <- update(rfm, rho.sigma.e = psi2propII(smoothPsi, k = 2.28),
+    rho.sigma.b = psi2propII(smoothPsi, k = 2.28))
```

Note that the update function both uses the call information from the given object `rfm`, and, if applicable, also uses estimates from the object as initial values for the fitting procedure.

To specify different ρ-functions for different random effects, the arguments `rho.b` and `rho.sigma.b` accept as input a list of ρ-functions. The order in the list of the random effects must be the same as the order listed in the output of `summary`. To fit only the element of $\boldsymbol{\theta}$ that corresponds to the "sample" random effect with higher efficiency, we use

```
R> rsb <- list(psi2propII(smoothPsi), psi2propII(smoothPsi, k = 2.28))
R> rfm3 <- update(rfm2, rho.sigma.b = rsb)
```

Note the missing second argument `k` when generating the first element of the list `rsb`. In that case, the default tuning parameter $k = 1.345$ is used.

To compare the estimates of the various fits we did so far, we can use the function `compare`. We set the argument `show.rho.functions` to `FALSE` to avoid a lengthy display of the ρ-functions here. To enhance the comparison, we also fit the model to the original, uncontaminated data with the classic, non-robust, method.

```
R> fmUncontam <- update(fm, data = Penicillin)
R> compare(fmUncontam, fm, rfm, rfm2, rfm3, show.rho.functions = FALSE)
```

	fmUncontam	fm	rfm	rfm2	rfm3
Coef					
(Intercept)	23 (0.809)	22.8 (0.85)	23 (0.911)	23 (0.848)	23 (0.849)
VarComp					
(Intercept) | plate	0.847	1.409	0.960	0.939	0.961
(Intercept) | sample	1.932	1.955	2.119	1.967	1.967
sigma	0.55	0.609	0.573	0.566	0.566
REML	331	377			

The resulting table gives the estimates and standard errors in parentheses. The `REML` row gives the restricted maximum likelihood statistic. As mentioned earlier, this statistic is unavailable for robust fits. The output of the `compare` function can also be passed to `xtable` from the **xtable** package (Dahl 2016) to create LaTeX or HTML-tables.

How to choose tuning parameters

When choosing tuning parameters for `rlmer`, one has to balance robustness and efficiency. In the examples discussed above, this means that by setting the tuning parameters too high, the estimates might break down and the resulting estimates are misleading. On the other hand, it is not good practice to set the tuning parameters very low as this will produce inefficient, i.e., imprecise, estimates. An approach as outlined above, fitting the model with low as well as with high efficiency, is better. In general, parameters that are not considered nuisance parameters should be estimated with high efficiency if possible.

When comparing fits with low to fits with high efficiency and robust to classic fits, one should first compare the parameter estimates (keeping in mind their precision or confidence intervals). If there are any relevant differences, then a study of the robustness weights should give insight as to which observations cause the difference. It is important not to just remove and, thus, ignore outliers. Whenever possible, the reason why an outlier is far from the bulk of observations should be determined. Outliers that are not merely due to recording errors usually carry information that can help to improve the model.

Applying this method to the example shown above, we can see that the one contaminated plate clearly inflates the classic, non-robust estimates for the standard deviation of the "plate" random effect. The robust method can detect this contamination and reduce its effect, leading to an estimate that is only slightly inflated. The comparison of the robust fits with lower and higher efficiency shows that the contamination is not strong enough to cause a breakdown of the fit with higher efficiency but lower robustness. Also, the higher efficiency for the estimate of the standard deviation of the "sample" random effect leads to a better estimate that is closer to the classic fits. Finally, the robust estimates of the standard deviation of the random errors are closer to the original classic fit of the uncontaminated data.

4.4. Controlling the fitting procedure

To diagnose problems with the fitting procedure, use the argument `verbose`. The argument takes values from 1 to 5, and gives more verbose output for higher values. If the method is not converging, increasing the maximum number of allowed iterations (argument `max.it`) or the tolerance (`rel.tol`) below which convergence is declared can help to achieve convergence.

To specify starting values for the fitting procedure, use the `init` argument. The `init` argument expects a list with four items: `fixef`, the fixed effects, `u`, the spherical random effects, `sigma`, the error scale σ, and `theta`, the vector $\boldsymbol{\theta}$ parameterizing the random effects covariance matrix $\boldsymbol{V}_b(\boldsymbol{\theta})$.

To shorten fitting times at the beginning of an analysis, use the argument `method = "DASvar"`. This method is faster as it uses simple direct approximations instead of numerical integrals to compute the scale and covariance parameters using the Design Adaptive Scale approach (see Koller and Stahel (2011) and Appendix B). The `DASvar` method yields approximate results only. For covariance matrices of the random effects $\boldsymbol{V}_b(\boldsymbol{\theta})$ with unstructured blocks of size three and larger, the method `DASvar` is the only method currently available.

5. Further information

Detailed information on the properties of the robust method and validation using simulation studies can be found in compact form in Koller and Stahel (2015) and more detailed in Koller (2013). Bates (2010) is a general introduction to mixed modeling using the R package **lme4** (Bates *et al.* 2015). Because **lme4** and **robustlmm** are similar, this is also a good starting point for using **robustlmm**.

We avoided the topic of robust testing for linear mixed-effects models in this tutorial. The usual caveats of testing in mixed models apply for the methods presented here. Bootstrap presents itself as a simple (but data structure dependent) way to get p values. One has to be careful, though, since the ordinary bootstrap quantiles are not robust (Salibián-Barrera, Van Aelst, and Willems 2008; Singh 1998).

Development of **robustlmm** is hosted on GitHub at `https://github.com/kollerma/robustlmm`. Any issues with the package can also be reported there.

Acknowledgments

The author would like to thank Kali Tal for providing editorial help with the manuscript. The author would also like to thank two anonymous reviewers for their helpful comments and suggestions on how to improve the paper. Finally, the author would like to thank Niels Hagenbuch for his comments and proof-reading.

References

Bates DM (2010). "**lme4**: Mixed-Effects Modeling with R." https://lme4.r-forge.r-project.org/book/.

Bates DM, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using **lme4**." *Journal of Statistical Software*, **67**(1), 1–48. doi:10.18637/jss.v067.i01.

Bilgic Y, Susmann H, McKean J (2014). ***rlme**: Rank-based Estimation and Prediction in Random Effects Nested Models*. R package version 0.4, URL https://CRAN.R-project.org/package=rlme.

Chervoneva I, Vishnyakov M (2011). "Constrained S-Estimators for Linear Mixed Effects Models with Covariance Components." *Statistics in Medicine*, **30**(14), 1735–1750. doi:10.1002/sim.4169.

Copt S, Victoria-Feser M (2006). "High-Breakdown Inference for Mixed Linear Models." *Journal of the American Statistical Association*, **101**(473), 292–300. doi:10.1198/016214505000000772.

Dahl DB (2016). ***xtable**: Export Tables to LaTeX or HTML*. R package version 1.8-2, URL https://CRAN.R-project.org/package=xtable.

Davies OL, Goldsmith PL (eds.) (1972). *Statistical Methods in Research and Production*. 4th edition. Hafner.

Demidenko E (2004). *Mixed Models: Theory and Applications*. John Wiley & Sons. doi:10.1002/0471728438.

Geraci M (2014). "Linear Quantile Mixed Models: The **lqmm** Package for Laplace Quantile Regression." *Journal of Statistical Software*, **57**(13), 1–29. doi:10.18637/jss.v057.i13.

Hampel F, Ronchetti E, Rousseeuw P, Stahel W (1986). *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons.

Koller M (2013). "Robust Estimation of Linear Mixed Models." Diss., ETH Zürich, Nr. 20997, 2013, URL https://doi.org/10.3929/ethz-a-007632241.

Koller M (2015). ***robustlmm**: Robust Linear Mixed Effects Models*. R package version 2.1, URL https://CRAN.R-project.org/package=robustlmm.

Koller M (2016). "**robustlmm**: An R Package for Robust Estimation of Linear Mixed-Effects Models." *Journal of Statistical Software*, **75**(6), 1–24. doi:10.18637/jss.v075.i06.

Koller M, Stahel WA (2011). "Sharpening Wald-Type Inference in Robust Regression for Small Samples." *Computational Statistics & Data Analysis*, **55**(8), 2504–2515. doi:10.1016/j.csda.2011.02.014.

Koller M, Stahel WA (2015). "Robust Estimation of General Mixed Effects Models." To be submitted.

Mächler M (2016). "CRAN Task View: Robust Statistical Methods." Version 2016-08-29, URL https://CRAN.R-project.org/view=Robust.

Maronna RA, Martin RD, Yohai VJ (2006). *Robust Statistics, Theory and Methods*. John Wiley & Sons. doi:10.1002/0470010940.

Osorio F (2016). ***heavy**: Package for Outliers Accommodation Using Heavy-Tailed Distributions*. R package version 0.38, URL https://cran.r-project.org/package=heavy.

Papritz A (2016). ***georob**: Robust Geostatistical Analysis of Spatial Data*. R package version 0.3-1, URL https://CRAN.R-project.org/package=georob.

Pinheiro JC, Bates DM, DebRoy S, Sarkar D, R Core Team (2016). ***nlme**: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-128, URL https://CRAN.R-project.org/package=nlme.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Salibián-Barrera M, Van Aelst S, Willems G (2008). "Fast and Robust Bootstrap." *Statistical Methods & Applications*, **17**(1), 41–71. doi:10.1007/s10260-007-0048-6.

Singh K (1998). "Breakdown Theory for Bootstrap Quantiles." *The Annals of Statistics*, **26**(5), 1719–1732. doi:10.1214/aos/1024691354.

Stahel W (1987). "Estimation of a Covariance Matrix with Location: Asymptotic Formulas and Optimal B-robust Estimators." *Journal of Multivariate Analysis*, **22**(2), 296–312. doi:10.1016/0047-259x(87)90092-3.

A. The smoothed Huber function and tables of tuning constants

The *smoothed Huber ψ-function* is defined as

$$\psi(x,k,s) = \begin{cases} x & |x| \le c \\ \text{sign}(x)\left(k - \frac{1}{(|x|-d)^s}\right) & \text{otherwise} \end{cases}, \tag{4}$$

where $c = k - s^{\frac{-s}{s+1}}$ and $d = c - s^{\frac{1}{s+1}}$. We recommend using a value of $s = 10$. The asymptotic properties of the regular Huber function and the smoothed Huber function are almost identical when this value is used. We can therefore safely use the same tuning parameter k for both ψ-functions. The two ψ-functions are compared in Figure 3. Tuning constants for this and the lqq ψ-function (Koller and Stahel 2011) are shown in Tables 2 to 5.

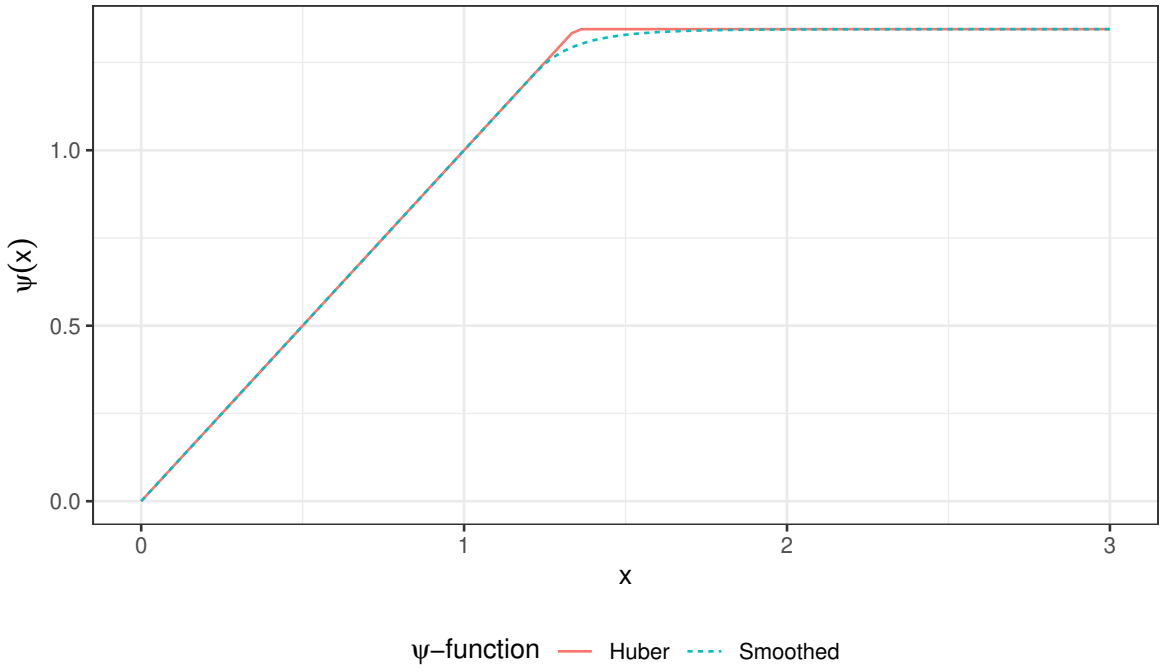

Figure 3: Comparison of the Huber and the smoothed Huber ψ-function for $k = 1.345$ and $s = 10$.

Efficiency	k for $\hat{\mu}$	k for $\hat{\sigma}$	k for $\hat{\sigma}$, Prop. II
0.80	0.53	0.50	1.49
0.85	0.73	0.71	1.69
0.90	0.98	1.08	1.94
0.95	1.345	1.66	2.28

Table 2: Tuning parameters k for scale estimates such that they reach the same asymptotic efficiency as the location estimate. For the Huber ψ-function.

	Dimension s					
	2	3	4	5	6	7
b_η	5.66	6.41	7.14	7.87	8.58	9.28
b_τ	5.15	5.55	5.91	6.25	6.55	6.84
b_μ	1.5	1.63	1.73	1.81	1.87	1.9

Table 3: Tuning parameters for the optimal B-estimator to yield 95% efficiency, non-diagonal case. For the Huber ψ-function.

Efficiency	cc for $\hat{\mu}$	cc for $\hat{\sigma}$
0.80	(0.946, 0.631)	(1.414, 0.942)
0.85	(1.058, 0.705)	(1.57, 1.05)
0.90	(1.214, 0.809)	(1.79, 1.19)
0.95	(1.474, 0.982)	(2.19, 1.46)

Table 4: Tuning parameters for lqq ψ-function for the location and scale estimates such that they reach the given asymptotic efficiency. The third parameter is always taken to be 1.5.

	Dimension s				
	2	3	4	5	6
cc_η	(6.44, 4.29)	(7.23, 4.82)	(8.01, 5.34)	(8.77, 5.85)	(9.52, 6.35)
cc_τ	(5.95, 3.97)	(6.41, 4.27)	(6.82, 4.55)	(7.2, 4.8)	(7.55, 5.03)
cc_μ	(1.63, 1.09)	(1.77, 1.18)	(1.88, 1.26)	(1.99, 1.32)	(2.08, 1.39)

Table 5: Tuning parameters for the lqq weight function to yield 95% efficiency, non-diagonal case. The third parameter is always taken to be 1.5.

B. Robust estimating equations

This Appendix summarises the more extensive derivation of the robust estimating equations found in Koller (2013).

B.1. Fixed and random effects

Let $k(j)$ be a function that maps random effect j to the corresponding block k, then the squared Mahalanobis distances of the estimated random effects are

$$\boldsymbol{d} = \left(d\left(\boldsymbol{b}^*_{k(j)}/\sigma\right)\right)_{j=1,\dots,q}, \quad \text{where} \quad d(\boldsymbol{b}^*_k) = \boldsymbol{b}^{*\top}_k \boldsymbol{b}^*_k .$$

We may then define the robustness weight for the jth random effect as $w_b(d_j)$. We use standard (location and linear regression) robustness weights:

$$w_b(d) = \begin{cases} \psi_b\left(\sqrt{d}\right)/\sqrt{d} & \text{if } d \neq 0 , \\ \psi'_b(0) & \text{if } d = 0 . \end{cases}$$

It is convenient to represent the robustness weights as (diagonal) weighting matrix,

$$\mathbf{W}_b(\boldsymbol{d}) = \mathbf{Diag}\left(w_b\left(d_{k(j)}\right)\right)_{j=1,\dots,q} .$$

The robust estimating equations are then

$$\begin{aligned} \boldsymbol{X}^\top \boldsymbol{U}_e^{-\top} \boldsymbol{\psi}_e(\widehat{\boldsymbol{\varepsilon}}^*/\sigma) &= 0 , \\ \boldsymbol{U}_b^\top \boldsymbol{Z}^\top \boldsymbol{U}_e^{-\top} \boldsymbol{\psi}_e(\widehat{\boldsymbol{\varepsilon}}^*/\sigma) - \boldsymbol{\Lambda}_b \mathbf{W}_b\left(\widehat{\boldsymbol{d}}\right)\widehat{\boldsymbol{b}}^*/\sigma &= 0 , \end{aligned} \tag{5}$$

where $\boldsymbol{\Lambda}_b = \mathbf{Diag}(\lambda_e/\lambda_{b,j})_{j=1,\dots,q}$ is a diagonal matrix with elements depending on the block size $s_{k(j)}$, $\lambda_e = \mathbb{E}_0[\psi'_e(\varepsilon^*)]$ and $\lambda_{b,j} = \widetilde{\lambda}\left(s_{k(j)}\right)$,

$$\widetilde{\lambda}(s) = \mathbb{E}_0\left[\frac{\partial}{\partial b^*_1}\left(w_b\left(\boldsymbol{b}^{*\top}\boldsymbol{b}^*\right)\boldsymbol{b}^*_1\right)\right], \quad \boldsymbol{b}^* \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{I}_s) .$$

B.2. Scale

We apply the Design Adaptive Scale approach following Koller and Stahel (2011). We get

$$\sum_{i=1}^n \tau^2_{e,i} w^{(\sigma)}_e\left(\frac{\widehat{\varepsilon}^*_i}{\tau_{e,i}\widehat{\sigma}}\right)\left[\left(\frac{\widehat{\varepsilon}^*_i}{\tau_{e,i}\widehat{\sigma}}\right)^2 - \kappa^{(\sigma)}_e\right] = 0 , \tag{6}$$

where the superscript $\cdot^{(\sigma)}$ is used to distinguish the weighting functions used for the scale and covariance parameters from the ones used for the fixed effects. Just as in the linear regression

case, we define $\tau_{e,i}$ as the value that zeroes the expectation of the i-th summand in (6). The expectation is

$$\mathbb{E}\left[w_e^{(\sigma)}\left(\frac{\widehat{\varepsilon}_i^*}{\tau_{e,i}\widehat{\sigma}}\right)\left(\frac{\widehat{\varepsilon}_i^*}{\tau_{e,i}\widehat{\sigma}}\right)^2 - \kappa_e^{(\sigma)} w_e^{(\sigma)}\left(\frac{\widehat{\varepsilon}_i^*}{\tau_{e,i}\widehat{\sigma}}\right)\right] = 0\,, \tag{7}$$

where the distribution of the residuals is approximated using a linear expansion of $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{b}}^*$ around their true values (Koller 2013, Appendix C), and $\kappa_e^{(\sigma)}$ is

$$\kappa_e^{(\sigma)} = \mathbb{E}_0\left[w_e^{(\sigma)}(\varepsilon)\varepsilon^2\right] \Big/ \mathbb{E}_0\left[w_e^{(\sigma)}(\varepsilon)\right]\,.$$

The weighting functions used for the scale estimates are the squared robustness weights used to estimate the fixed and random effects, $w_e^{(\sigma)}(x) = (\psi_e^{(\sigma)}(x)/x)^2$, $w_e^{(\sigma)}(0) = \psi_e^{(\sigma)\prime}(0)$, for convex ρ-functions. For redescending ρ-functions, it is unnecessary to use the squared robustness weights. Using the same weights as for the fixed and random effects still gives robust estimates (assuming $\psi(x)x$ is bounded). When the squared weights are used, a different set of tuning parameters must be used to estimate the scale and covariance parameters. Tables of tuning parameters can be found in Appendix A.

B.3. Covariance parameters

For the covariance parameters, we have to treat the diagonal and the block-diagonal case of $\boldsymbol{U}_b$ separately.

Diagonal case

In the case of diagonal $\boldsymbol{U}_b(\boldsymbol{\theta})$, estimating $\widehat{\boldsymbol{\theta}}$ is essentially a scale estimation problem on $\widehat{\boldsymbol{b}}^*$. It can be robustified just like the estimating equation for $\hat{\sigma}$, see Equation 6. For a model with only one random effects term, the robust estimating equations are

$$\sum_{j=1}^{q} \tau_{b,j}^2 w_b^{(\sigma)}\left(\frac{\widehat{b}_j^*}{\tau_{b,j}\widehat{\sigma}}\right)\left[\left(\frac{\widehat{b}_j^*}{\tau_{b,j}\widehat{\sigma}}\right)^2 - \kappa_b^{(\sigma)}\right] = 0\,, \tag{8}$$

with $\tau_{b,j}$ such that

$$\mathbb{E}\left[w_b^{(\sigma)}\left(\frac{\widehat{b}_j^*}{\tau_{b,j}\widehat{\sigma}}\right)\left[\left(\frac{\widehat{b}_j^*}{\tau_{b,j}\widehat{\sigma}}\right)^2 - \kappa_b^{(\sigma)}\right]\right] = 0\,,$$

and normalizing constant

$$\kappa_b^{(\sigma)} = \mathbb{E}_0\left[w_b^{(\sigma)}(b^*)b^{*2}\right] \Big/ \mathbb{E}_0\left[w_b^{(\sigma)}(b^*)\right]\,.$$

Generalization to multiple random effects terms is straightforward. We get one such equation for each of the random effects terms.

Block-diagonal case

For block-diagonal $\boldsymbol{U}_b(\boldsymbol{\theta})$ we have to take care of the block structure. The normalizing constant $\tau_{b,j}^2$ must be replaced by a matrix $\boldsymbol{T}_{b,k}$ defined for each block k. Analogous to the estimator for the covariance matrix and location problem, we must use two different weight functions: one for the size of the matrix $w_b^{(\tau)}$ and another one for the shape $w_b^{(\eta)}$. For details, we refer to Stahel (1987) and Hampel, Ronchetti, Rousseeuw, and Stahel (1986, Chapter 5). As in the cited references, we introduce a third weight function $w_b^{(\delta)}$ to simplify notation. For block types with dimension $s > 1$, let

$$w_b^{(\delta)}(d) = \Big(dw_b^{(\eta)}(d) - \Big(d - s\kappa_b^{(\tau)}\Big)w_b^{(\tau)}\Big(d - s\kappa_b^{(\tau)}\Big)\Big)\Big/ s \;,$$

where $\kappa_b^{(\tau)}$ is defined such that

$$\mathbb{E}\Big[\Big(u - s\kappa_b^{(\tau)}\Big)w_b^{(\tau)}\Big(u - s\kappa_b^{(\tau)}\Big)\Big] = 0 \quad \text{for } u \sim \chi_s^2.$$

Remark. The optimal B-robust estimator derived in Stahel (1987) is given by $w_b^{(\tau)}(d) = \min(1/b_\tau, 1/d)$ and $w_b^{(\eta)}(d) = \min(1/b_\eta, 1/d)$. Other weight functions may be chosen, as long as $\psi(d) = dw(d)$ is a ψ-function. For $w_b^{(\tau)}$ and $w_b^{(\eta)}$ given above, this would be the Huber ψ-function. For low dimensions s, one may choose $w_b^{(\tau)} = w_b^{(\eta)}$. In higher dimensions, the efficiency loss for the estimated size is negligible. Hence, a smaller tuning parameter may be chosen for $w_b^{(\eta)}$. For $s = 2$ and Huber or smoothed Huber ψ-functions (see Appendix A), the squared tuning parameter of $\rho_e^{(\sigma)}$ for $w_b^{(\tau)}$may be used to get approximately the same efficiency for $\widehat{\boldsymbol{\theta}}$ as for $\widehat{\sigma}$. Tables of tuning parameters for higher dimensions for the Huber and the lqq ψ-functions can be found in Appendix A.

Before we can state the robust estimating equation for the block-diagonal case, we need one more definition. Let

$$\boldsymbol{Q}_\ell(\boldsymbol{\theta}) = \boldsymbol{U}_b(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{U}_b(\boldsymbol{\theta})}{\partial \theta_l} \;.$$

The robust estimating equation in the block-diagonal case can then be defined as follows. For $l = 1, \ldots, r$,

$$\begin{aligned}\sum_{k=1}^{K}\Big[w_b^{(\eta)}\Big(d\Big(\boldsymbol{T}_{b,k}^{-1/2}\widehat{\boldsymbol{b}}_k^*/\widehat{\sigma}\Big)\Big)\widehat{\boldsymbol{b}}_k^{*\top}\boldsymbol{Q}_{\ell,k}(\widehat{\boldsymbol{\theta}})\widehat{\boldsymbol{b}}_k^*/\widehat{\sigma}^2 \\ - w_b^{(\delta)}\Big(d\Big(\boldsymbol{T}_{b,k}^{-1/2}\widehat{\boldsymbol{b}}_k^*/\widehat{\sigma}\Big)\Big)\operatorname{tr}\Big(\boldsymbol{T}_{b,k}\boldsymbol{Q}_{\ell,k}(\widehat{\boldsymbol{\theta}})\Big)\Big] = 0 \;,\end{aligned} \tag{9}$$

where $\boldsymbol{Q}_{\ell,k}(\widehat{\boldsymbol{\theta}})$ is the $s \times s$ submatrix of $\boldsymbol{Q}_\ell(\widehat{\boldsymbol{\theta}})$ which acts on block k and $\boldsymbol{T}_{b,k}^{-1/2}$ is the inverse of any square root of the $s \times s$ matrix $\boldsymbol{T}_{b,k}$. As in the diagonal case, we define the matrix $\boldsymbol{T}_{b,k}$ such that each summand has expectation zero. For $l = 1, \ldots, r$,

$$\mathbb{E}\Big[w_b^{(\eta)}\Big(d\Big(\boldsymbol{T}_{b,k}^{-1/2}\widehat{\boldsymbol{b}}_k^*/\sigma\Big)\Big)\widehat{\boldsymbol{b}}_k^{*\top}\boldsymbol{Q}_{\ell,k}(\widehat{\boldsymbol{\theta}})\widehat{\boldsymbol{b}}_k^*/\sigma^2 - w_b^{(\delta)}\Big(d\Big(\boldsymbol{T}_{b,k}^{-1/2}\widehat{\boldsymbol{b}}_k^*/\sigma\Big)\Big)\operatorname{tr}\Big(\boldsymbol{T}_{b,k}\boldsymbol{Q}_{\ell,k}(\widehat{\boldsymbol{\theta}})\Big)\Big] = 0\,.$$

Remarks. The symmetric matrix $\boldsymbol{T}_{b,k}$ is fully defined for unstructured covariance matrices only, where $r = s(s+1)/2$. For other covariance matrix structures we can replace $\boldsymbol{T}_{b,k}$ by the variance of the linear approximation of $\boldsymbol{b}^*$.

Since in the classic case, the linear approximations for $\widehat{\boldsymbol{b}}^*$ and $\widehat{\boldsymbol{\varepsilon}}^*$ are exact, the estimating equation (9) reduces to the REML estimating equations. A similar argument is valid for the estimating equation (6) for $\hat{\sigma}$.

C. Estimation algorithm

The algorithm for finding the simultaneous roots of the estimating equations (5), (6) and (8) (and/or (9)) can be split into four general steps. They are:

1. Compute initial estimates.
2. For given $\widehat{\boldsymbol{\theta}}$, $\hat{\sigma}$, find $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{b}}^*$ that solve (5).
3. Keeping the intermediate solutions $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{b}}^*$ fixed, find $\hat{\sigma}$ such that (6) is fulfilled.
4. Check the estimating equations for $\widehat{\boldsymbol{\theta}}$, (8) and/or (9), for convergence. If they are not fulfilled, update $\widehat{\boldsymbol{\theta}}$ in some way and go back to step 2.

The algorithms for the four steps can be chosen independently from each other. We discuss the four steps below.

When this algorithm stops, it has found a simultaneous solution of all the estimating equations, except in the block-diagonal case where some, but not all, components corresponding to a block might lie on the border of the parameter space. For those parameters, the estimating equations won't be satisfied. To avoid incorrect solutions, it is crucial that the estimates for $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{b}}^*$ are updated for each new candidate of $\widehat{\boldsymbol{\theta}}$ and that the initial estimate for $\boldsymbol{\theta}$ is large enough. Otherwise, the algorithm might wrongly set all components of $\boldsymbol{\theta}$ corresponding to one block to zero or close to zero.

This is illustrated for a simple one-way anova in Figure 4. The expected sum of squares vanishes for $\widehat{\boldsymbol{\theta}} = \mathbf{0}$ in the classic case. In the robust case, the expectation does not vanish, but there is a solution close to zero. This is an artifact of the linear approximation used to compute the expectation. As long as convex ρ-functions are used, the classic estimates are generally a good choice of initial estimates. Zero components of the initial $\widehat{\boldsymbol{\theta}}$ should be set to one at the start of the algorithm.

C.1. Step 1: Initial estimates

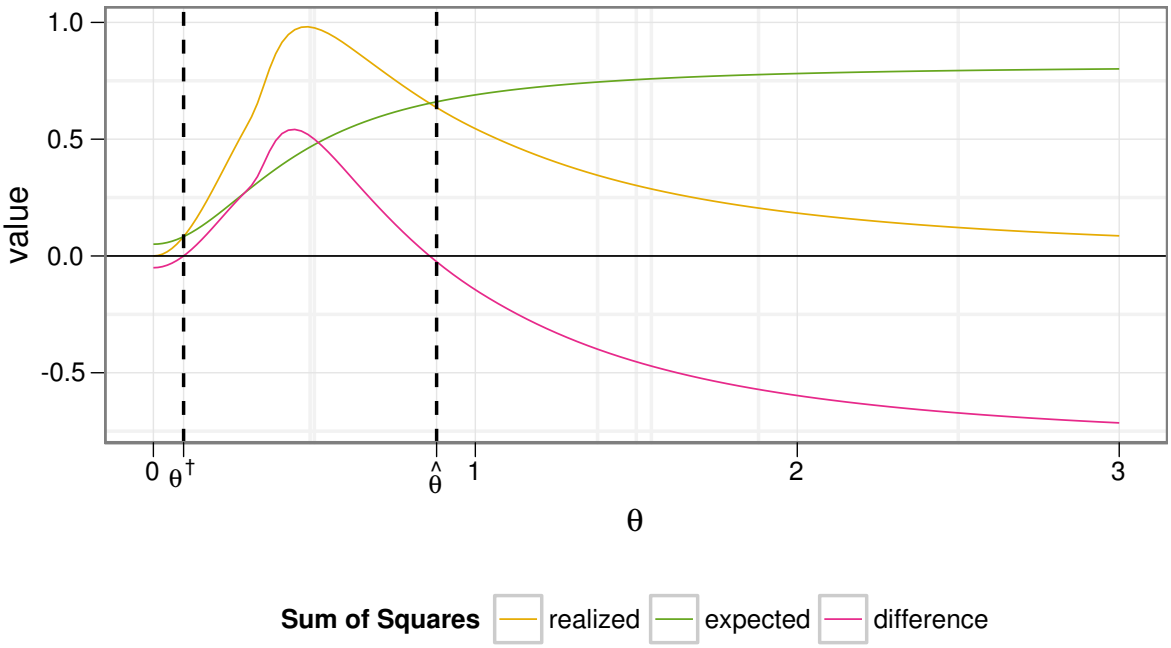

Figure 4: Sum of Squares of the spherical random effects for a balanced one-way anova. The smoothed Huber function was used for both ρ_e and ρ_b. The estimating equations (8) are solved at the points where the two curves cross. The solutions are highlighted by black dashed lines, $\hat{\theta}$ is the correct solution, $\theta^\dagger$ the wrong one.

The methods described here work for convex ρ-functions as well as for redescender ρ-functions. If redescender ρ-functions are used, the algorithm as defined here converges to a local solution. It is up to the initial estimator to provide starting values that ensure the algorithm converges to the right local solution, whatever the right solution is. In case of MM-estimates for the fixed effects model, the initial S-estimate makes sure that the final estimate has the desired high breakdown point. The same would certainly also be desirable in case of mixed-effects models. However, to the best of our knowledge, there exists currently no such estimator. The S-estimators by Copt and Victoria-Feser (2006) and Chervoneva and Vishnyakov (2011) do not seem suitable, since they are based on a different contamination model and are not as general as the method proposed here.

If convex ρ-functions are used, this difficulty does not exist. Apart from the artificial solution $\widehat{\boldsymbol{\theta}}$ close to zero (see Figure 4) which is easily distinguishable from the true solution, we conjecture that the solutions are unique as they are for the Proposal 2 case in the location-scale problem (Koller 2013). We therefore consider it safe to use the classic solutions as initial estimates when convex ρ-functions are used.

Redescender ρ-functions have the advantage that they can assign a weight of zero to some observations or random effects levels. This makes it possible that such observations have no influence on the estimates. When convex ρ-functions are used, an observation practically

always has an influence on the estimates, since a weight of zero is only reached in the limit, when the residual or the random effect level approaches plus or minus infinity. If one is interested in eliminating the influence of observations, then one might consider the following. First, compute the fit using a convex ρ-function. Then use the results as starting value for fitting using a redescender ρ-function in a second step. In the absence of good initial estimators for redescender ρ-functions, this approach might be used to get at least some of the desirable properties of redescender ρ-functions.

C.2. Step 2: Fixed and random effects

For given $\boldsymbol{\theta}$ and σ, the estimation of the fixed and random effects can be done using iteratively reweighted least squares.

Let $\mathbf{W}_e$ be defined analogously to $\mathbf{W}_b$, i.e.,

$$\mathbf{W}_e = \mathbf{Diag}(w_e(\varepsilon_i^*/\sigma))_{i=1,\ldots,n} ,$$

where

$$w_e(\varepsilon^*) = \begin{cases} \psi_e(\varepsilon^*)/\varepsilon^* & \text{if } \varepsilon^* \neq 0 , \\ \psi_e'(0) & \text{if } \varepsilon^* = 0 . \end{cases}$$

Then insert these weights into (5) and expand $\widehat{\boldsymbol{\varepsilon}}^*$ to get the following linear system of equations,

$$\begin{bmatrix} \boldsymbol{X}^\top \boldsymbol{U}_e^{-\top} \mathbf{W}_e \boldsymbol{U}_e^{-1} \boldsymbol{X} & \boldsymbol{X}^\top \boldsymbol{U}_e^{-\top} \mathbf{W}_e \boldsymbol{U}_e^{-1} \boldsymbol{Z} \boldsymbol{U}_b \\ \boldsymbol{U}_b^\top \boldsymbol{Z}^\top \boldsymbol{U}_e^{-\top} \mathbf{W}_e \boldsymbol{U}_e^{-1} \boldsymbol{X} & \boldsymbol{U}_b^\top \boldsymbol{Z}^\top \boldsymbol{U}_e^{-\top} \mathbf{W}_e \boldsymbol{U}_e^{-1} \boldsymbol{Z} \boldsymbol{U}_b + \boldsymbol{\Lambda}_b \mathbf{W}_b \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\boldsymbol{b}}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}^\top \boldsymbol{U}_e^{-\top} \mathbf{W}_e \boldsymbol{y} \\ \boldsymbol{U}_b^\top \boldsymbol{Z}^\top \boldsymbol{U}_e^{-\top} \mathbf{W}_e \boldsymbol{y} \end{bmatrix} .$$

By alternating between computing $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{b}}^*$ for a given set of weights and updating the weights for a given set of estimates, we get a simple and efficient algorithm for computing the fixed and random effects.

We start the algorithm with either a predefined set of weights or set all the weights to one. When the relative change of the estimates is small enough, the algorithm can stop.

C.3. Step 3: Variance parameter

Equation 6 can be written as

$$\widehat{\sigma}^2 = \frac{\sum_{i=1}^n w_e^{(\sigma)}\left(\frac{\widehat{\varepsilon}_i^*}{\tau_{e,i}\widehat{\sigma}}\right)\widehat{\varepsilon}_i^{*2}}{\kappa_e^{(\sigma)} \sum_{i=1}^n \tau_{e,i}^2 w_e^{(\sigma)}\left(\frac{\widehat{\varepsilon}_i^*}{\tau_{e,i}\widehat{\sigma}}\right)} .$$

This suggests a simple two-step algorithm, namely alternating between computing $\widehat{\sigma}$ using the above formula and updating the weights given $\widehat{\sigma}$. This algorithm is quick and reliable, especially if the overall algorithm has almost converged and $\widehat{\sigma}$ only changes little between iterations of $\widehat{\boldsymbol{\theta}}$.

A similar procedure can also be derived for the computation of $\tau_{e,i}$. Solving (7) for $\tau_{e,i}$ yields

$$\tau_{e,i}^2 = \mathbb{E}\left[w_e^{(\sigma)}\left(\frac{\widehat{\varepsilon}_i^*}{\tau_{e,i}\widehat{\sigma}}\right)\left(\frac{\widehat{\varepsilon}_i^*}{\widehat{\sigma}}\right)^2\right] \Big/ \mathbb{E}\left[\kappa_e^{(\sigma)} w_e^{(\sigma)}\left(\frac{\widehat{\varepsilon}_i^*}{\tau_{e,i}\widehat{\sigma}}\right)\right],$$

which again suggests to use the same two-step procedure as lined out above. The values $\tau_{e,i}$ have to be recomputed for every new value of $\widehat{\boldsymbol{\theta}}$, preferably using the values of the last candidate $\widehat{\boldsymbol{\theta}}$ as starting values.

C.4. Step 4: Covariance parameters

In the following, we will assume that we have only one block type. The algorithms mentioned below can be easily generalized to multiple block types. One iteration then consists of computing the updates for every block type separately before applying all of them together.

In case of diagonal $\boldsymbol{U}_b(\boldsymbol{\theta})$, $\widehat{\boldsymbol{\theta}}$ may be computed using the analogue of the algorithm for Step 3. This has proven to be much more efficient and robust compared to using a generic root solving procedure.

The same is true in the non-diagonal case. Nevertheless, if we assume a special covariance structure, the only options are generic root solving procedures such as Newton-Raphson. The Newton-Raphson algorithm, however, can be quite unstable and often does not converge for problems with many parameters.

In the case of unstructured covariance matrices, there exists a better algorithm of EM-type. Let the function $\mathbf{L}(\boldsymbol{A})$ return the lower triangular Cholesky factor of $\boldsymbol{A}$, and $\mathbf{L}^{-1}$ return the inverse of the factor. Then, for unstructured covariance matrices and in terms of the first block $\boldsymbol{U}_{b,1}$ of $\boldsymbol{U}_b$, the update is

$$\boldsymbol{U}_{b,1}\left(\widehat{\boldsymbol{\theta}}^{[\text{it}]}\right) = \boldsymbol{U}_{b,1}\left(\widehat{\boldsymbol{\theta}}^{[\text{it}-1]}\right)\frac{1}{\sigma}\,\mathbf{L}\left(\sum_{k=1}^{K}\widehat{w}_{b,k}^{(\eta)}\widehat{\boldsymbol{b}}_k^*\widehat{\boldsymbol{b}}_k^{*\top}\right)\mathbf{L}^{-1}\left(\sum_{k=1}^{K}\widehat{w}_{b,k}^{(\delta)}\boldsymbol{T}_{b,k}\right), \tag{10}$$

where the superscript in square brackets denotes the iteration. The right-hand side is computed using $\widehat{\boldsymbol{\theta}}^{[\text{it}-1]}$, the value from the last iteration, and $\widehat{w}_{b,k}^{(.)}$ is the corresponding k-th robustness weight.

Remark. To see that (10) is indeed a sensible update, we have to first rewrite the r scalar valued estimating equations into one matrix valued estimating equation. We may write (8) as

$$\sum_{k=1}^{K}\left[\operatorname{tr}\left(\left(\widehat{w}_{b,k}^{(\eta)}\widehat{\boldsymbol{b}}_k^*\widehat{\boldsymbol{b}}_k^{*\top}/\widehat{\sigma}^2 - \widehat{w}_{b,k}^{(\delta)}\boldsymbol{T}_{b,k}\right)\boldsymbol{Q}_{\ell,k}(\widehat{\boldsymbol{\theta}})\right)\right] = 0 \quad \text{for } l = 1,\ldots,r\,.$$

When assuming an unstructured covariance matrix for the random effects, $\boldsymbol{Q}_{\ell,k}$ has only one non-zero value and does not depend on k. (For other block types, $\boldsymbol{Q}_{\ell,k}$ vanishes, thus decoupling the problem

for different block types.) Since $r = s(s+1)/2$, we may thus write the estimating equation as

$$\sum_{k=1}^{K}\left[\widehat{w}_{b,k}^{(\eta)}\widehat{\boldsymbol{b}}_k^*\widehat{\boldsymbol{b}}_k^{*\top} - \widehat{\sigma}^2\widehat{w}_{b,k}^{(\delta)}\boldsymbol{T}_{b,k}\right] = \boldsymbol{0}\,.$$

The dependence of the robustness weights on $\widehat{\boldsymbol{\theta}}$ will be neglected from now on, thereby reducing the problem to solving a system of linear equations. In terms of the actual random effects, the estimating equation in iteration [it] reads

$$\sum_{k=1}^{K}\left[\widehat{w}_{b,k}^{(\eta)}\boldsymbol{U}_{b,1}^{-1}\left(\widehat{\boldsymbol{\theta}}^{[\mathrm{it}-1]}\right)\widehat{\boldsymbol{b}}_k\widehat{\boldsymbol{b}}_k^{\top}\boldsymbol{U}_{b,1}^{-\top}\left(\widehat{\boldsymbol{\theta}}^{[\mathrm{it}-1]}\right) - \widehat{\sigma}^2\widehat{w}_{b,k}^{(\delta)}\boldsymbol{T}_{b,k}\right] = \boldsymbol{0}\,.$$

As long as the algorithm has not converged, the estimating equation is not fulfilled for $\widehat{\boldsymbol{\theta}}^{[\mathrm{it}-1]}$, but there exists a $\widehat{\boldsymbol{\theta}}^{[\mathrm{it}]}$, such that it is. For

$$\boldsymbol{U}_{b,1}\left(\widehat{\boldsymbol{\theta}}^{[\mathrm{it}]}\right) = \boldsymbol{U}_{b,1}\left(\widehat{\boldsymbol{\theta}}^{[\mathrm{it}-1]}\right)\Delta\boldsymbol{U}_b^{[\mathrm{it}]}\,,$$

where $\Delta\boldsymbol{U}_b^{[\mathrm{it}]}$ is a lower triangular matrix, we have after multiplying the equation by $\Delta\boldsymbol{U}_b^{[\mathrm{it}]}$ from the left and by $\Delta\boldsymbol{U}_b^{[\mathrm{it}]\top}$ from the right,

$$\sum_{k=1}^{K}\left[\widehat{w}_{b,k}^{(\eta)}\widehat{\boldsymbol{b}}_k^*\widehat{\boldsymbol{b}}_k^{*\top} - \widehat{\sigma}^2\widehat{w}_{b,k}^{(\delta)}\Delta\boldsymbol{U}_b^{[\mathrm{it}]}\boldsymbol{T}_{b,k}\,\Delta\boldsymbol{U}_b^{[\mathrm{it}]\top}\right] = \boldsymbol{0}\,.$$

By splitting the left-hand side into two sums, moving the second sum to the right-hand side, and replacing both sides by the corresponding lower-triangular Cholesky factor, we get an equation that can be solved for $\Delta\boldsymbol{U}_b^{[\mathrm{it}]}$ and thus an expression for $\boldsymbol{U}_{b,1}\left(\widehat{\boldsymbol{\theta}}^{[\mathrm{it}]}\right)$, which is exactly update (10) mentioned above.

The resulting algorithm, considering steps 2 to 4 together, is then of EM-type. It converges fairly quickly, except when the solution is zero, i.e., some variance components are dropped. An illustration of the problem and potential improvements to the algorithm can be found in Demidenko (2004, Section 2.12).

Affiliation:

Manuel Koller
Institute for Social and Preventive Medicine
Universität Bern
3012 Bern, Switzerland
E-mail: kollermal@proton.me
URL: https://www.ispm.unibe.ch/