Journal of Data Science 0 (0), 1–23
??? 2026
DOI: 10.6339/26-JDS1246
Computing in Data Science

Mixed-Model Log-Likelihood Evaluation via a Blocked Cholesky Factorization

Douglas Bates[1,*], Phillip Alday[2], and Ajinkya Kokandakar[1]

[1]*Department of Statistics, University of Wisconsin–Madison, U.S.A.*
[2]*Beacon Biosignals, U.S.A.*

Abstract

Bates et al. (2015) described the evaluation of the profiled log-likelihood of a linear mixed-effects model by updating a sparse, symmetric positive-definite matrix and computing its Cholesky factor, as implemented in the *lme4* package for R. Here we present enhancements to the derivation and theoretical presentation of the result and to its implementation using a blocked Cholesky factorization in the *MixedModels.jl* package for Julia (Bezanson et al., 2017). The gain in computational efficiency is primarily due to three factors: (1) the new derivation allows us to compute the penalized sum of squared residuals without computing the conditional estimates of the fixed-effects parameters and the conditional modes of the random effects at each optimization step, (2) the blocked Cholesky representation and careful ordering of the random effects terms reduces the amount of "fill-in" that occurs during the Cholesky factorization, and (3) the multiple dispatch feature of the Julia language allows us to use specialized algorithms for different kinds of matrices instead of relying on generic algorithms during the Cholesky factorization. To show the effectiveness of the blocked Cholesky approach we use it to fit a linear mixed model to over 32 million ratings of movies in the MovieLens `ml-32m` (Harper and Konstan, 2016) data set. The model incorporates random effects for over 200,000 movies and over 80,000 participants. Further enhancements to these computational methods are suggested.

Keywords *fill-reducing permutation; mixed-effects models; partially crossed grouping; profiled log-likelihood; rectangular full-packed storage*

1 Introduction

Mixed effects models are a rich class of statistical models frequently used when the measurement units exhibit grouping structures. They are especially useful when the data exhibit non-nested grouping, for instance when there are two or more unrelated grouping variables such as measurements taken on items and users (see Baayen et al., 2008); or partially nested groupings, for instance when measuring test scores for students over time when the students can change schools (see Fühner et al., 2021). There is a long history of computational tools for fitting mixed effects models, including the *MIXED* procedure in SAS, and the *nlme*, *lme4* and more recently *glmmTMB* packages in the R ecosystem. The extensive use of these models is shown by the fact that Bates et al. (2015), which describes the methodology and use of *lme4*, is among the top 25 most-cited papers of the twenty-first century (Pearson et al., 2025). We present *MixedModels.jl* a new implementation of mixed effects models in the Julia programming language. This imple-

*Corresponding author Email: bates@stat.wisc.edu.

© 2026 The Author(s). Published by the School of Statistics and the Center for Applied Statistics, Renmin University of China. Open access article under the CC BY license.
Received June 1, 2026; Accepted September 8, 2026

mentation relies on two innovations, one analytical and the other computational. At the heart of *MixedModels.jl*, there is a new way of formulating the profiled log-likelihood evaluation that expresses this objective without requiring explicit evaluation of the conditional parameter estimates. On the computational level, the evaluation of the objective requires the Cholesky factor of a large, sparse, positive-definite symmetric matrix, as in *lme4*. This matrix has a blocked structure that is explicitly exploited in *MixedModels.jl* but not in *lme4*. More concretely, the Cholesky factor consists of dense and sparse blocks, and using specific algorithms to handle these blocks differently leads to more computationally efficient algorithms. We exploit the multiple dispatch feature of Julia to do exactly that – use sparse matrix algorithms for the sparse blocks, and level 3 BLAS and LAPACK subroutines for the dense blocks. Together, these contributions allow us to fit linear mixed models to very large datasets.

Bates et al. (2015) provided a formulation for linear mixed-effects models and showed how maximum likelihood estimates of the parameters in the model can be determined by optimizing a profiled objective function, which is negative twice the profiled log-likelihood. The term "profiled" refers to the fact that this objective is a function of some (but not all) of the parameters in the model. As shown in Bates et al. (2015), the profiled objective can be evaluated from the solution of a penalized least squares (PLS) problem. In this paper, we show that it is not necessary to evaluate the solution to the PLS problem explicitly when evaluating the profiled objective. The objective can be evaluated from a factorization that is an intermediate step in solving the PLS problem.

The model matrix, $\mathbf{Z}$, for the random effects in such models is defined in blocks according to the "grouping factors" for random-effects terms in the model formula. The blocked structure of $\mathbf{Z}$ and of its cross-product matrix, $\mathbf{Z}^\top\mathbf{Z}$, (also called the Gram matrix of the columns of $\mathbf{Z}$) are described in great detail in Bates et al. (2015, Section 2) when showing how a model formula and data frame create the structures used to evaluate the profiled objective. However, the implementation of the profiled log-likelihood evaluation in the *lme4* package for R is not explicitly based upon the blocks — *lme4* uses a more general sparse Cholesky factorization as implemented in the *CHOLMOD* library (Chen et al., 2008).

An important part in a general sparse Cholesky factorization, such as implemented in *CHOLMOD*, is determining a *fill-reducing permutation* of the rows and columns of the symmetric, positive-definite matrix to be factored. The main difference between the methods in *MixedModels.jl* and those in *lme4* is that in *MixedModels.jl* it is the blocks that are permuted, if necessary, rather than individual rows and columns. In other words, the particular structure of the blocks in the Gram matrix is exploited in *MixedModels.jl* rather than relying upon general sparse Cholesky techniques that are not able to exploit this structure.

The structure of the blocks in the Gram matrix depends upon the number and types of random-effects terms in the model. A general and efficient implementation of the blocked Cholesky factorization requires the ability to define operations for many different combinations of specialized forms of matrices. Julia (Bezanson et al., 2017) is an ideal environment for this type of calculation because of its efficient implementation of multiple dispatch, its just-in-time compilation, and its rich type system.

The rest of the paper is organized as follows. In Section 2 we recap the derivation of the profiled log-likelihood from Bates et al. (2015), in a slightly different form from that paper, and show that the objective can be evaluated directly from the Cholesky factorization of an augmented positive-definite matrix. That is, explicit evaluation of the solution to the PLS problem is not necessary. In Section 3 we use an example of a fitted mixed-effects model to illustrate our implementation of the blocked Cholesky factorization in *MixedModels.jl*. In Section 4 we use an

example of a mixed-effects model fit to a large data set with partially crossed grouping factors for the random effects to examine how the compute time per evaluation of the objective and the memory footprint of the model depend on various dimensions of the model. We conclude with a few additional comments in Section 5 followed by a short summary and discussion in Section 6.

2 Profiled Log-Likelihood of Linear Mixed-Effects Models

The types of linear mixed-effects models that can be fit by *lme4* or *MixedModels.jl* are based on two vector-valued random variables: the q-dimensional vector of random effects, $\mathcal{B}$, and the n-dimensional response vector, $\mathcal{Y}$. The unconditional distribution of $\mathcal{B}$ and the conditional distribution of $\mathcal{Y}$, given $\mathcal{B} = \mathbf{b}$, are defined as multivariate Gaussian distributions of the form

$$\begin{aligned}(\mathcal{Y}|\mathcal{B} = \mathbf{b}) &\sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b}, \sigma^2\mathbf{I}) \\ \mathcal{B} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\theta)\ ,\end{aligned} \tag{1}$$

where $\mathbf{X}$ is the $n \times p$ model matrix for the fixed-effects terms, $\boldsymbol{\beta}$ is the p-dimensional fixed-effects parameter vector and $\mathbf{Z}$ is the $n \times q$ model matrix for the random effects. The $q \times q$ symmetric variance-covariance matrix $\text{Var}(\mathcal{B}) = \boldsymbol{\Sigma}_\theta$ depends on the *variance-component parameter vector* $\boldsymbol{\theta}$ through a lower triangular *relative covariance factor* $\boldsymbol{\Lambda}_\theta$ as

$$\boldsymbol{\Sigma}_\theta = \sigma^2\boldsymbol{\Lambda}_\theta\boldsymbol{\Lambda}_\theta^\top\ . \tag{2}$$

(Recall that the lower Cholesky factor is generally written $\mathbf{L}$. In this case the lower Cholesky factor contains parameters and is named with the corresponding Greek letter, $\boldsymbol{\Lambda}$.)

The matrices $\mathbf{Z}$, $\boldsymbol{\Sigma}_\theta$ and $\boldsymbol{\Lambda}_\theta$ can be very large but have special sparsity structures, as described in Section 3.

In many descriptions of linear mixed models, computational formulas are written in terms of the *precision matrix*, $\boldsymbol{\Sigma}_\theta^{-1}$. Such formulas will become unstable as $\boldsymbol{\Sigma}_\theta$ approaches singularity. And it can do so; singular (i.e. non-invertible) $\boldsymbol{\Sigma}_\theta$ can and do occur in practice. Moreover, during the course of the numerical optimization by which the parameter estimates are determined, it is frequently the case that the log-likelihood or the REML criterion will need to be evaluated at values of $\boldsymbol{\theta}$ that produce a singular $\boldsymbol{\Sigma}_\theta$. Because of this we will take care to use computational methods that can be applied even when $\boldsymbol{\Sigma}_\theta$ is singular and are stable as $\boldsymbol{\Sigma}_\theta$ approaches singularity.

According to Equation (2), $\boldsymbol{\Sigma}$ depends on both σ and θ, and we should write it as $\boldsymbol{\Sigma}_{\sigma,\theta}$. However, we will blur that distinction and continue to write $\text{Var}(\mathcal{B}) = \boldsymbol{\Sigma}_\theta$.

Another technicality is that the *common scale parameter*, σ, could, in theory, be zero. However, the only way for its estimate, $\widehat{\sigma}$, to be zero is for the fitted values from the fixed-effects only, $\mathbf{X}\widehat{\boldsymbol{\beta}}$, to be exactly equal to the observed data. This occurs only with data that have been (incorrectly) simulated without error. In practice, we can safely assume that $\sigma > 0$. However, the estimated $\boldsymbol{\Lambda}_\theta$, like $\boldsymbol{\Sigma}_\theta$, can be singular.

The computational methods in *lme4* and in *MixedModels.jl* are based on $\boldsymbol{\Lambda}_\theta$ and do not require evaluation of $\boldsymbol{\Sigma}_\theta$. In fact, $\boldsymbol{\Sigma}_\theta$ is explicitly evaluated only at the converged parameter estimates.

To express the log likelihood without the precision matrix, we define the spherical random effects, $\mathcal{U} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_q)$, which in turn determine $\mathcal{B}$ as

$$\mathcal{B} = \boldsymbol{\Lambda}_\theta\mathcal{U}\ . \tag{3}$$

Although it may seem more intuitive to write $\mathcal{U}$ as a linear transformation of $\mathcal{B}$, we cannot do that when $\mathbf{\Lambda}_\theta$ is singular, which is why Equation (3) is in the form shown.

We can easily verify that Equation (3) provides the desired distribution for $\mathcal{B}$. As a linear transformation of a multivariate Gaussian random variable, $\mathcal{B}$ will also be multivariate Gaussian with mean

$$\mathbb{E}\left[\mathcal{B}\right] = \mathbb{E}\left[\mathbf{\Lambda}_\theta \mathcal{U}\right] = \mathbf{\Lambda}_\theta \, \mathbb{E}\left[\mathcal{U}\right] = \mathbf{\Lambda}_\theta \mathbf{0} = \mathbf{0} \ ,$$

and covariance matrix

$$\mathrm{Var}(\mathcal{B}) = \mathbf{\Lambda}_\theta \mathrm{Var}(\mathcal{U}) \mathbf{\Lambda}_\theta^\top = \sigma^2 \mathbf{\Lambda}_\theta \mathbf{\Lambda}_\theta^\top = \mathbf{\Sigma}_\theta \ .$$

Just as we concentrate on how $\boldsymbol{\theta}$ determines $\mathbf{\Lambda}_\theta$, not $\mathbf{\Sigma}_\theta$, we will concentrate on properties of $\mathcal{U}$ rather than $\mathcal{B}$. In particular, we now define the model according to the distributions

$$\begin{aligned} (\mathcal{Y} | \mathcal{U} = \mathbf{u}) &\sim \mathcal{N}(\mathbf{Z}\mathbf{\Lambda}_\theta \mathbf{u} + \mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n) \\ \mathcal{U} &\sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_q) \ . \end{aligned} \tag{4}$$

The joint density function for $\mathcal{Y}$ and $\mathcal{U}$ is the product of the densities of the two distributions in Equation (4). That is

$$f_{\mathcal{Y},\mathcal{U}}(\mathbf{y}, \mathbf{u}) = \frac{1}{\left(2\pi\sigma^2\right)^{(n+q)/2}} \exp\left(\frac{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{\Lambda}_\theta \mathbf{u}\|^2 + \|\mathbf{u}\|^2}{-2\sigma^2} \right) \ . \tag{5}$$

To evaluate the likelihood for the parameters, $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, and σ^2, given the observed response, $\mathbf{y}$, we must evaluate the marginal distribution of $\mathcal{Y}$, whose density is the integral of $f_{\mathcal{Y},\mathcal{U}}(\mathbf{y}, \mathbf{u})$ with respect to $\mathbf{u}$.

This evaluation can be simplified if we rewrite the *penalized sum of squared residuals*, $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{\Lambda}_\theta \mathbf{u}\|^2 + \|\mathbf{u}\|^2$, the numerator of the exponent in Equation (5), as a quadratic form in $\mathbf{u}$, to isolate the dependence on $\mathbf{u}$

$$\begin{aligned} r_\theta^2(\mathbf{u}, \boldsymbol{\beta}) &= \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{\Lambda}_\theta \mathbf{u}\|^2 + \|\mathbf{u}\|^2 \\ &= \left\| \begin{bmatrix} \mathbf{Z}\mathbf{\Lambda}_\theta & \mathbf{X} & \mathbf{y} \\ -\mathbf{I}_q & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} -\mathbf{u} \\ -\boldsymbol{\beta} \\ 1 \end{bmatrix} \right\|^2 \\ &= \begin{bmatrix} -\mathbf{u} \\ -\boldsymbol{\beta} \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{\Lambda}_\theta^\top \mathbf{Z}^\top \mathbf{Z} \mathbf{\Lambda}_\theta + \mathbf{I} & \mathbf{\Lambda}_\theta^\top \mathbf{Z}^\top \mathbf{X} & \mathbf{\Lambda}_\theta^\top \mathbf{Z}^\top \mathbf{y} \\ \mathbf{X}^\top \mathbf{Z} \mathbf{\Lambda}_\theta & \mathbf{X}^\top \mathbf{X} & \mathbf{X}^\top \mathbf{y} \\ \mathbf{y}^\top \mathbf{Z} \mathbf{\Lambda}_\theta & \mathbf{y}^\top \mathbf{X} & \mathbf{y}^\top \mathbf{y} \end{bmatrix} \begin{bmatrix} -\mathbf{u} \\ -\boldsymbol{\beta} \\ 1 \end{bmatrix} \\ &= \left\| \begin{bmatrix} \mathbf{R}_{ZZ} & \mathbf{R}_{ZX} & \mathbf{r}_{Zy} \\ \mathbf{0} & \mathbf{R}_{XX} & \mathbf{r}_{Xy} \\ \mathbf{0} & \mathbf{0} & r_{yy} \end{bmatrix} \begin{bmatrix} -\mathbf{u} \\ -\boldsymbol{\beta} \\ 1 \end{bmatrix} \right\|^2 \\ &= \|\mathbf{r}_{Zy} - \mathbf{R}_{ZX}\boldsymbol{\beta} - \mathbf{R}_{ZZ}\mathbf{u}\|^2 + \|\mathbf{r}_{Xy} - \mathbf{R}_{XX}\boldsymbol{\beta}\|^2 + r_{yy}^2 \ , \end{aligned} \tag{6}$$

using the Cholesky factor of the blocked matrix,

$$\begin{aligned}\boldsymbol{\Omega}_\theta &= \begin{bmatrix} \boldsymbol{\Lambda}_\theta^\top \mathbf{Z}^\top \mathbf{Z} \boldsymbol{\Lambda}_\theta + \mathbf{I} & \boldsymbol{\Lambda}_\theta^\top \mathbf{Z}^\top \mathbf{X} & \boldsymbol{\Lambda}_\theta^\top \mathbf{Z}^\top \mathbf{y} \\ \mathbf{X}^\top \mathbf{Z} \boldsymbol{\Lambda}_\theta & \mathbf{X}^\top \mathbf{X} & \mathbf{X}^\top \mathbf{y} \\ \mathbf{y}^\top \mathbf{Z} \boldsymbol{\Lambda}_\theta & \mathbf{y}^\top \mathbf{X} & \mathbf{y}^\top \mathbf{y} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_{ZZ}^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{R}_{ZX}^\top & \mathbf{R}_{XX}^\top & \mathbf{0} \\ \mathbf{r}_{Zy}^\top & \mathbf{r}_{Xy}^\top & r_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{ZZ} & \mathbf{R}_{ZX} & \mathbf{r}_{Zy} \\ \mathbf{0} & \mathbf{R}_{XX} & \mathbf{r}_{Xy} \\ \mathbf{0} & \mathbf{0} & r_{yy} \end{bmatrix} . \end{aligned} \tag{7}$$

(In practice we evaluate the blocks of the lower Cholesky factor, such as $\mathbf{L}_{ZZ} = \mathbf{R}_{ZZ}^\top$, of this matrix but the equations are slightly easier to write in terms of the upper Cholesky factor.)

Note that the block in the upper left, $\boldsymbol{\Lambda}_\theta^\top \mathbf{Z}^\top \mathbf{Z} \boldsymbol{\Lambda}_\theta + \mathbf{I}$, is positive definite even when $\boldsymbol{\Lambda}_\theta$ is singular, because

$$\mathbf{u}^\top \left(\boldsymbol{\Lambda}_\theta^\top \mathbf{Z}^\top \mathbf{Z} \boldsymbol{\Lambda}_\theta + \mathbf{I} \right) \mathbf{u} = \|\mathbf{Z} \boldsymbol{\Lambda}_\theta \mathbf{u}\|^2 + \|\mathbf{u}\|^2 \ , \tag{8}$$

and the first term is non-negative while the second is positive if $\mathbf{u} \neq \mathbf{0}$.

Thus, the triangular $\mathbf{R}_{ZZ}$, with positive diagonal elements, can be evaluated and its determinant, $|\mathbf{R}_{ZZ}|$, which is the product of its diagonal elements, is also positive. This determinant appears in the marginal density of $\boldsymbol{\mathcal{Y}}$, from which the likelihood of the parameters is evaluated.

To evaluate the likelihood,

$$L(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma | \mathbf{y}) = \int_{\mathbf{u}} f_{\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{U}}}(\mathbf{y}, \mathbf{u}) \, d\mathbf{u} \ , \tag{9}$$

we isolate the part of the joint density that depends on $\mathbf{u}$ and perform a change of variable

$$\mathbf{v} = \mathbf{R}_{ZZ} \mathbf{u} + \mathbf{R}_{ZX} \boldsymbol{\beta} - \mathbf{r}_{Zy} \ , \tag{10}$$

with the *Jacobian matrix*

$$\frac{d\mathbf{v}}{d\mathbf{u}} = \mathbf{R}_{ZZ} \ .$$

From the properties of the multivariate Gaussian distribution

$$\begin{aligned} &\int_{\mathbf{u}} \frac{1}{(2\pi\sigma^2)^{q/2}} \exp\left(-\frac{\|\mathbf{R}_{ZZ}\mathbf{u} + \mathbf{R}_{ZX}\boldsymbol{\beta} - \mathbf{r}_{Zy}\|^2}{2\sigma^2} \right) d\mathbf{u} \\ &\qquad = \int_{\mathbf{v}} \frac{1}{(2\pi\sigma^2)^{q/2}} \exp\left(-\frac{\|\mathbf{v}\|^2}{2\sigma^2} \right) |\mathbf{R}_{ZZ}|^{-1} \, d\mathbf{v} \\ &\qquad = |\mathbf{R}_{ZZ}|^{-1} \ , \end{aligned} \tag{11}$$

from which we obtain the likelihood as

$$L(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma; \mathbf{y}) = \frac{|\mathbf{R}_{ZZ}|^{-1}}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{r_{yy}^2 + \|\mathbf{R}_{XX}(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}})\|^2}{2\sigma^2} \right) \ , \tag{12}$$

where the conditional estimate, $\widehat{\boldsymbol{\beta}}$, given $\boldsymbol{\theta}$, satisfies

$$\mathbf{R}_{XX} \widehat{\boldsymbol{\beta}} = \mathbf{r}_{Xy} \ .$$

Setting $\boldsymbol{\beta} = \widehat{\boldsymbol{\beta}}$ and taking the logarithm provides the estimate of σ^2, given $\boldsymbol{\theta}$, as

$$\widehat{\sigma^2} = \frac{r_{yy}^2}{n} \,, \tag{13}$$

which gives the *profiled log-likelihood*, $\ell(\boldsymbol{\theta}|\mathbf{y}) = \log L(\boldsymbol{\theta}, \widehat{\boldsymbol{\beta}}, \widehat{\sigma})$, on the deviance scale, as

$$-2\ell(\boldsymbol{\theta}|\mathbf{y}) = 2\log(|\mathbf{R}_{ZZ}|) + n\left(1 + \log\left(\frac{2\pi r_{yy}^2(\boldsymbol{\theta})}{n}\right)\right) \,, \tag{14}$$

which is an affine function of the logarithms of the diagonal elements of $\mathbf{R}$.

One of the interesting aspects of this formulation is that it is not necessary to solve for the conditional estimate of $\boldsymbol{\beta}$ or the conditional modes of the random effects when evaluating the log-likelihood. The two values needed for the log-likelihood evaluation, $2\log(|\mathbf{R}_{ZZ}|)$ and r_{yy}^2, are obtained directly from the diagonal elements of the Cholesky factor.

Furthermore, $\boldsymbol{\Omega}_\theta$ and, from that, the Cholesky factor, $\mathbf{R}_\theta$, and the objective to be optimized can be evaluated for a given value of $\boldsymbol{\theta}$ from the Gram matrix of the columns of $\mathbf{Z}$, $\mathbf{X}$ and $\mathbf{y}$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{Z}^\top\mathbf{Z} & \mathbf{Z}^\top\mathbf{X} & \mathbf{Z}^\top\mathbf{y} \\ \mathbf{X}^\top\mathbf{Z} & \mathbf{X}^\top\mathbf{X} & \mathbf{X}^\top\mathbf{y} \\ \mathbf{y}^\top\mathbf{Z} & \mathbf{y}^\top\mathbf{X} & \mathbf{y}^\top\mathbf{y} \end{bmatrix} \,, \tag{15}$$

and $\boldsymbol{\Lambda}_\theta$.

In *MixedModels.jl* the `LinearMixedModel` struct contains a blocked representation of $\mathbf{A}$ in the `A` field and a similarly structured lower-triangular blocked array in the `L` field. Evaluation of the objective simply involves updating a representation of the relative covariance factor $\boldsymbol{\Lambda}_\theta$ from $\boldsymbol{\theta}$, forming $\boldsymbol{\Omega}_\theta$ (Equation (7)) from $\mathbf{A}$ and $\boldsymbol{\Lambda}_\theta$, then evaluating its lower Cholesky factor, $\mathbf{L}_\theta$.

In Equation (15) the blocking structure is according to all the random effects, followed by the fixed-effects followed by the response. However, as described in Bates et al. (2015, Section 3), the structure of $\mathbf{Z}$, $\mathbf{Z}^\top\mathbf{Z}$ and $\boldsymbol{\Lambda}_\theta$ is further subdivided into blocks according to the random-effects terms in the model formula, which we cover in the next section.

3 Random-Effects Terms in Mixed-Model Formulas

The user interface in *MixedModels.jl* is intentionally similar to that of *lme4*. A model is specified, as described in Bates et al. (2015, section 2.1), by a formula of the form

```
@formula(resp ~ FEexpr + (REexpr1 | factor1) + (REexpr2 | factor2) + ...)
```

"where `FEexpr` is an expression determining the columns of the fixed-effects model matrix, $\mathbf{X}$, and the random-effects terms, `(REexpr1 | factor1)`, `(REexpr2 | factor2)`, etc. together determine both the random-effects model matrix $\mathbf{Z}$ and the structure of the relative covariance factor $\boldsymbol{\Lambda}_\theta$."

(In Julia a formula expression such as this must be enclosed in a call to the `@formula` macro.)

The expression on the right hand side of the vertical bar, `|`, in a random-effects term is called the *grouping factor* for those random effects.

3.1 From Formula to Model Matrices

The process of converting a mixed-model formula for a data set into the random-effects model matrix, $\mathbf{Z}$, the Gram matrix, $\mathbf{Z}^\top\mathbf{Z}$, and the mapping from the relative covariance parameter, $\boldsymbol{\theta}$, to the relative covariance factor, $\boldsymbol{\Lambda}_\theta$, is described in considerable detail in Bates et al. (2015, Sections 2.2 and 2.3). In particular, Tables 3 and 4 of Bates et al. (2015) provide the dimensions and the names of some of the components of these matrices and should be kept close at hand when reading this section. For convenience, Table 3 from Bates et al. (2015) has been reproduced with slight changes as Table 1.

Table 1: Sizes of model components.

Symbol	Description	`insteval` example
n	length of the response vector $\mathcal{Y}$	$n = 73{,}421$
p	no. of fixed effects in the model matrix $\mathbf{X}$	$p = 2$
k	no. of random effects terms	$k = 3$ (`s`, `d`, `dept`)
p_i	no. of fixed effects terms in the i^{th} random effect	(1, 1, 2)
ℓ_i	no. of levels of the i^{th} grouping factor	(2972, 1128, 14)
$q_i = p_i\ell_i$	no. of columns of the term-wise model matrix	(2972, 1128, 28)
$q = \sum_i q_i$	no. of columns in the random effects model matrix $\mathbf{Z}$	$q = 4{,}128$
$q + p + 1$	no. of columns (and rows) of the augmented matrix $\mathbf{A}$	4131

To recap some of the material in Bates et al. (2015, Sections 2.2 and 2.3), in a model with k random-effects terms, an $n \times p_i$ model matrix $\mathbf{X}_i, i = 1, \ldots, k$ is evaluated from the expression $\texttt{REterm}_i$ and the data. The $\mathbf{X}_i, i = 1, \ldots, k$ are (implicitly) combined with the indicator matrices, $\mathbf{J}_i, i = 1, \ldots, k$ of the ith grouping factor to produce the sparse model matrix blocks $\mathbf{Z}_i, i = 1, \ldots, k$. If $\ell_i, i = 1, \ldots, k$ is the number of levels of the ith grouping factor then the total number of the random effects for each grouping factor is $q_i = \ell_i p_i$, $i = 1, \ldots, k$ with $q = \sum_{i=1}^{k} q_i$ being the total number of random effects in the model.

Generally the $p_i, i = 1, \ldots, k$ are small. Random effects terms of the form `(1|g)`, i.e. $p_i = 1$, are not uncommon. In such cases, called simple, scalar random-effects terms, the matrix $\mathbf{Z}_i$ is exactly the indicator matrix $\mathbf{J}_i$ and its $\ell_i \times \ell_i$ Gram matrix, $\mathbf{Z}_i^\top\mathbf{Z}_i$, is diagonal with the incidence counts of the levels on the diagonal.

The blocked derivation in Bates et al. (2015) carries over into the model representation in *MixedModels.jl* with one modification — in *lme4* the (implicit) partitioning of $\mathbf{Z}$ into k vertical blocks is according to the random-effects terms in the formula, whereas in *MixedModels.jl* it is according to the grouping factors. If more than one random-effects terms have the same grouping factor they are amalgamated in the model representation.

Making the number of blocks small reduces the complexity of the evaluation of the profiled likelihood while retaining simplicity in the factorization of the first column of blocks (see Section 3.3).

3.1.1 An Example Using Instructor Evaluation Data

The `InstEval` dataset from the *lme4* package is reproduced as `insteval` in *MixedModels.jl*. (The Julia convention is to use lower-case for names of objects and camel-case for names of types and

Table 2: Summary of the `insteval` dataset.

```
using AlgebraOfGraphics, CairoMakie, Printf, SparseArrays
using Arrow, Chairmarks, CSV, DataFrames, LinearAlgebra, MixedModels
dat = DataFrame(MixedModels.dataset("insteval"))
dat.service = (dat.service .== "Y")         # convert service to a Boolean vector
show(
  describe(dat, :min, :max, :mean, :nunique, :eltype);
  row_labels=nothing,
  eltypes=false,
  summary=false,
)
```

variable	min	max	mean	nunique	eltype
s	S0001	S2972		2972	String
d	I0001	I2160		1128	String
dept	D01	D15		14	String
studage	2	8		4	String
lectage	1	6		6	String
service	false	true	0.432887		Bool
y	1	5	3.20574		Int8

for package names.) It consists of nearly 75,000 evaluations (`y`) of instructors (`d` for *Dozent*, German for "instructor") at ETH-Zürich by students (`s`). Other covariates, such as the department (`dept`) that offered the course and whether or not it was a `service` course, are included. Further details are given on the R help page for `lme4::InstEval`.

In Julia we attach the packages to be used, load the data set, convert it to a `DataFrame` and evaluate some descriptive statistics on its columns to produce Table 2, which shows that a total of 2972 students evaluated a total of 1128 instructors in courses from 14 different departments. About 43% of the evaluations were from service courses.

We will use random effects to model the variation due to instructors and due to students, as we regard the individual levels of these factors as being samples from their respective populations. We will also use random effects to model the variation due to departments and due to possible department-to-department variation in the service-course effect. (Technically this study is more of a census of departments than a random sample but random effects are often used when the number of levels of the factor is moderate, to help stabilize the computation.)

The model formula is chosen to illustrate certain aspects of the conversion from a model formula to the internal model representation. In particular, we choose independent random effects for `dept` and for `service` by `dept` to show that distinct terms with the same grouping factor will be amalgamated in the internal representation.

```
form = @formula(y ~ 1 + service + (1|d) + (1|s) + (1|dept) + (0+service|dept))
m1 = fit(MixedModel, form, dat, progress=false) # suppress display of progress bar
print(m1)
```

```
Linear mixed model fit by maximum likelihood
 y ~ 1 + service + (1 | d) + (1 | s) + (1 | dept) + (0 + service | dept)
    logLik     -2 logLik       AIC         AICc          BIC
 -118824.3008  237648.6016  237662.6016  237662.6032  237727.0294

Variance components:
            Column   Variance Std.Dev.   Corr.
s        (Intercept)  0.105296 0.324493
d        (Intercept)  0.262433 0.512282
dept     (Intercept)  0.002580 0.050795
         service      0.023400 0.152972   .
Residual              1.385008 1.176864
 Number of obs: 73421; levels of grouping factors: 2972, 1128, 14

  Fixed-effects parameters:
──────────────────────────────────────────────────────
                 Coef.  Std. Error       z  Pr(>|z|)
──────────────────────────────────────────────────────
(Intercept)   3.27765     0.0235036  139.45    <1e-99
service      -0.0507428   0.0439893   -1.15    0.2487
──────────────────────────────────────────────────────
```

Notice that, in the formula, the random effects term for instructor, `(1|d)`, occurred before that for student, `(1|s)`. In the model the `s` term occurs first because the number of random effects from that term, 2972, is greater than that from `d`, 1128. Also the two terms for `dept` are amalgamated into a single term in the model.

The optimization of the profiled objective required about 175 evaluations of the objective function and took about one and a half seconds on a laptop computer, as shown in this benchmark using the `@be` macro from the *Chairmarks.jl* package.

```
@be fit($MixedModel, $form, $dat; progress=false)  seconds=8
```

```
Benchmark: 7 samples with 1 evaluation
 min    1.121 s (440286 allocs: 70.645 MiB)
 median 1.137 s (440286 allocs: 70.645 MiB, 0.91% gc time)
 mean   1.167 s (440286 allocs: 70.645 MiB, 2.45% gc time)
 max    1.233 s (440286 allocs: 70.645 MiB, 6.88% gc time)
```

The environment in which these benchmarks were performed is described in the supplemental materials. Note that we use the Apple Accelerate implementation of the BLAS/Lapack routines on computers with Apple M-series processors and Intel's MKL BLAS implementation on computers with x86_64 processors. In Julia hot-swapping BLAS implementations is as easy as loading the appropriate package, such as *AppleAccelerate.jl* or *MKL.jl*.

Individual evaluations of the profiled objective, including installing a new covariance parameter vector, updating the blocked Cholesky factorization, and evaluating Equation (14), takes a few milliseconds on the same laptop computer.

Table 3: Block structure of the `A` and `L` matrices in the `insteval` example.

rows	s	d	dept	fixed
2972	Diagonal			
1128	Sparse	Diag/TrRFP		
28	Dense	Sparse/Dense	BlkDiag/Dense	
3	Dense	Dense	Dense	Dense

```
@be objective!($m1, $(m1.optsum.final))
```

```
Benchmark: 17 samples with 1 evaluation
 min    6.168 ms (35 allocs: 880 bytes)
 median 6.202 ms (35 allocs: 880 bytes)
 mean   6.219 ms (35 allocs: 880 bytes)
 max    6.293 ms (35 allocs: 880 bytes)
```

The random effects terms define scalar random effects for `s` and `d` and vector-valued random effects for `dept` in this model, as reflected in the data type of each of the `reterms` of the model

```
show([typeof(rt) for rt in m1.reterms])
```

```
DataType[ReMat{Float64, 1}, ReMat{Float64, 1}, ReMat{Float64, 2}]
```

In the notation of Bates et al. (2015, Section 2) the second parameter of each of these types is $p_i, i = 1, 2, 3$.

The number of levels of the grouping factors are

```
show([length(rt.levels) for rt in m1.reterms])
```

```
[2972, 1128, 14]
```

In the notation of Bates et al. (2015, Section 2) these are $\ell_i, i = 1, 2, 3$.

Finally the numbers of columns in each of the $\mathbf{Z}_i, i = 1, 2, 3$ are

```
show([size(rt, 2) for rt in m1.reterms])
```

```
[2972, 1128, 28]
```

For a summary of the model component sizes see Table 1.

The blocks of the augmented Gram matrix, $\mathbf{A}$, and the lower Cholesky factor, $\mathbf{L}_\theta = \mathbf{R}_\theta^\top$, are summarized in Table 3.

```
BlockDescription(m1)
```

(Only the lower triangle of the symmetric matrix, $\mathbf{A}$, is stored.)

In the theoretical development the blocking structure of the Gram matrix, $\mathbf{A}$, and the Cholesky factor (Equation (7)) is according to random effects, fixed effects and response. In practice the response vector, $\mathbf{y}$, is concatenated (`hcat`) to the fixed effects model matrix, $\mathbf{X}$, and the random effects model matrix, $\mathbf{Z}$, is partitioned according to the grouping factors.

In the block structure summary (Table 3) if a block of $\mathbf{L}$ and its corresponding block of $\mathbf{A}$ have the same structure, the name of that structure is given once. This is the case for all of the blocks except the [2,2] block, which is diagonal in $\mathbf{A}$ but a dense lower triangular matrix, stored in the Rectangular Full Packed format (Gustavson et al., 2010) in $\mathbf{L}$, the [2,3] block, which is sparse in $\mathbf{A}$ but dense in $\mathbf{L}$, and the [3,3] block, which is block-diagonal in $\mathbf{A}$ and triangular dense (full storage) in $\mathbf{L}$.

As described in Bates et al. (2015, Section 3) the relative covariance factor, $\mathbf{\Lambda}_\theta$ is block diagonal with the same blocking structure as the random effects parts of $\mathbf{A}$ or $\mathbf{L}$. The ith diagonal block, $i = 1, \ldots, k$, consists of q_i repetitions on the diagonal of a $\ell_i \times \ell_i$ lower-triangular template matrix $\mathbf{T}_i$. The last diagonal block, corresponding to the fixed-effects and response, is the $(p+1) \times (p+1)$ identity matrix.

For a scalar random effects term (i.e. $\ell_i = 1$) its block in $\mathbf{\Lambda}_\theta$ is a multiple of the $q_i \times q_i$ identity matrix. The blocks of $\mathbf{\Lambda}_\theta$ are never instantiated in the *MixedModels.jl* implementation. There is sufficient flexibility in the multiple dispatch model in Julia that pre-multiplication by $\mathbf{\Lambda}_\theta^\top$ or post-multiplication by $\mathbf{\Lambda}_\theta$ can be performed from the template matrices $\mathbf{T}_i, i = 1, \ldots, k$ and the `refs` property of the grouping factor alone.

The value of $\hat{\boldsymbol{\theta}}$ for model `m1` is

```
show(m1.optsum.final)
```

```
[0.2757268746920012, 0.43529426043944414, 0.04316133870171569, 0.12998270495422298]
```

which maps to the template matrices

```
m1.lambda
```

```
3-element Vector{AbstractMatrix{Float64}}:
 [0.2757268746920012;;]
 [0.43529426043944414;;]
 Diagonal([0.04316133870171569, 0.12998270495422298])
```

3.2 The Rectangular Full Packed (RFP) Format

The diagonal blocks of $\mathbf{A}$ generated by random-effects terms are either diagonal, for scalar random effects, or block diagonal, consisting of ℓ_i small blocks of size $p_i \times p_i$, for vector-valued random effects. As described in Section 3.3, the diagonal or block-diagonal structure of the [1, 1] block of $\mathbf{A}$ carries over to the [1, 1] block of $\mathbf{L}$. However, with more than one grouping factor in a non-nested design, the diagonal blocks of $\mathbf{L}$ below the [1, 1] block get "filled-in" with non-zeros in most or all of the lower triangle. Allocating a $q_i \times q_i$ array for the i'th diagonal block of $\mathbf{L}$ but only using the lower triangle is fine for small q_i but becomes expensive when the number of levels of the i'th grouping factor is large.

The rectangular full-packed (RFP) storage format (Gustavson et al., 2010) for triangular or symmetric/Hermitian matrices, designated `TrRFP` in Table 3, allows for a triangular matrix to

be packed into (roughly) half the amount of storage required for the dense square matrix. For the [2, 2] block in Table 3, a dense square matrix of size 1128×1128 requires about 9.71 MB of storage whereas the RFP format, which is stored in a rectangular array of size 1129×564 requires about 4.86 MB.

The *RectangularFullPacked.jl* package allows access to this format in Julia.

Full expanation of all forms of RFP storage requires considerable detail, for which see Gustavson et al. (2010), but the broad outline for a lower triangular matrix is that the first $n/2$ (rounded up for odd n) columns are stored unchanged in the rectangular backing array. For an even n the number of rows in the backing array is $n + 1$ and the columns of this "trapezoidal" part of the original array are shifted down by one row. The triangular part from the lower right of the original array is transposed and stored in the upper right corner of the backing array.

The details are not as important as the fact that 3/4 of the original triangular array (the trapezoidal part from the first $n/2$ columns) is stored essentially the same way as it would be in full triangular storage and only 1/4 of the original array requires customized access methods.

The LAPACK implementation of the Cholesky factorization for the RFP format (`dpftrf`) is nearly as fast as the Cholesky factorization for a dense square matrix (`dpotrf`). However, the blocked Cholesky factorization requires several operations to evaluate the matrix to be passed to LAPACK's dense Cholesky factorization. The most expensive of these occurs after $\mathbf{L}[2, 1]$ and the diagonal matrix $\mathbf{\Lambda}_2^\top \mathbf{A}[2, 2]\mathbf{\Lambda}_2+\mathbf{I}$ have been evaluated (in the storage for $\mathbf{L}[2, 2]$). It is described as a "symmetric rank-k update" and involves replacing the current $\mathbf{L}[2, 2]$ by $\mathbf{L}[2, 2]-\mathbf{L}[2, 1]\mathbf{L}[2, 1]^\top$ (because of the negative sign in that expression this is actually a "downdate").

In practical examples, $\mathbf{L}[2, 1]$ is sparse and the update is performed by iterating over columns of $\mathbf{L}[2, 1]$. Each pair of nonzeros in each column of $\mathbf{L}[2, 1]$ generates a (scalar) update of one of the elements of $\mathbf{L}[2, 2]$ and this is where the more expensive indexing system in the RFP formulation (relative to storing a square dense matrix) takes its toll. The number of updates done to the triangular part of the RFP representation of $\mathbf{L}[2, 2]$ versus the trapezoidal part depends on ordering of the levels of the second grouping factor. If we order these levels by decreasing incidence then usually a substantial majority of the updates occur in the trapezoidal part, using Cartesian or linear indexing, as opposed to the triangular part with a more complicated indexing scheme.

This re-ordering of the levels of the second grouping factor is a heuristic but, in our experience, a successful heuristic for reducing the time to perform the downdate operation.

Two arguments to the `fit` method for a linear mixed model, `RFPthreshold`, which defaults to 1000, and `sortlevels`, which defaults to `true`, control whether RFP storage will be used for a dense triangular block of $\mathbf{L}$ and whether the levels of the corresponding grouping factor will be sorted by descreasing incidence.

We can fit the `insteval` model with triangular storage for $\mathbf{L}[2, 2]$ as

```
m1f = fit(MixedModel, form, dat; RFPthreshold=1500, progress=false);
BlockDescription(m1f)
```

for which evaluation of the objective takes

```
@be objective!($m1f, $(m1f.optsum.final))
```

```
Benchmark: 18 samples with 1 evaluation
 min    5.801 ms (32 allocs: 752 bytes)
```

Table 4: Block structure of the `A` and `L` matrices in the full storage `insteval` example.

rows	s	d	dept	fixed
2972	Diagonal			
1128	Sparse	Diag/Dense		
28	Dense	Sparse/Dense	BlkDiag/Dense	
3	Dense	Dense	Dense	Dense

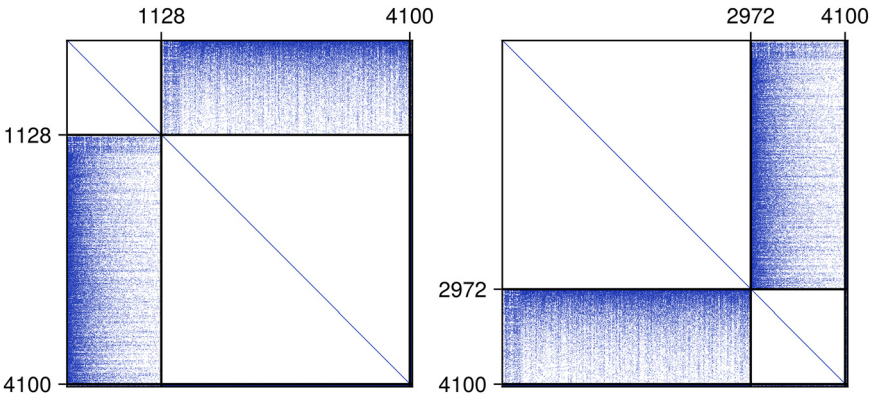


Figure 1: Sparsity pattern of the augmented matrix **A** when the order of the random effects is not carefully chosen (left) and when the order is carefully chosen (right).

```
median 5.829 ms (32 allocs: 752 bytes)
mean   5.829 ms (32 allocs: 752 bytes)
max    5.865 ms (32 allocs: 752 bytes)
```

That is, evaluation of the objective is slightly faster with the full square storage of $\mathbf{L}[2,2]$, compared to the RFP storage, but at the expense of more memory usage

```
(Base.summarysize(m1), Base.summarysize(m1f))
```

```
(18830527, 23914895)
```

3.3 Special Role of the [1,1] Block

The reason for ordering the random-effects terms so that the [1,1] block of **A** is as large as possible is because the diagonal or block diagonal structure of this block is preserved in the [1,1] block of **L**. To illustrate the importance of choosing the first block of the augmented matrix **A**, we fit the same model with (a) the [1,1] block corresponding to the random effect term for the instructor (`1 | d`) with 1128 levels, and (b) the [1,1] block corresponding to the random effect term for students (`(1 | s)` with 2972 levels).

Figure 1 shows the non-zero elements of the augmented matrix, **A**, when (a) the first block corresponds to the random effect terms for the instructor (`(1 | d)` with 1128 levels), and (b) the first block corresponds to the student (`(1 | s)` with 2972 levels), and Figure 2 shows the non-zero elements for the corresponding lower-triangular Cholesky factor. When performing a block-wise Cholesky factorization, the diagonal structure of the [1, 1] block from the input matrix

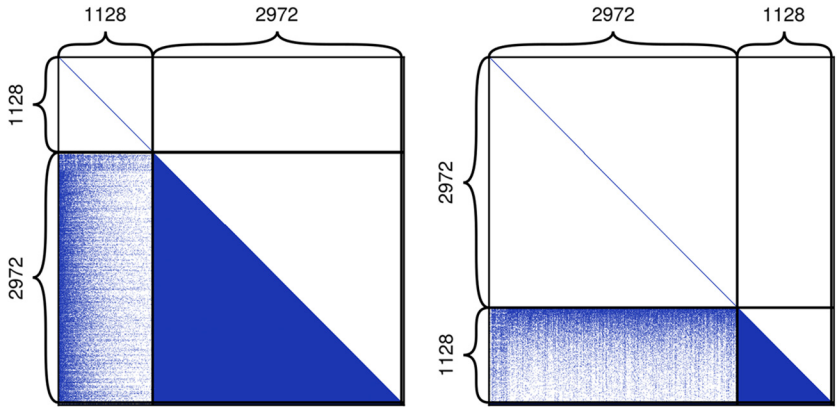


Figure 2: Sparsity pattern of the lower triangular Cholesky factor when the order of the random effects is not carefully chosen (left) and when the order is carefully chosen (right).

is preserved in the Cholesky factor, but the diagonal structure for the subsequent diagonal blocks (e.g. [2,2]) is not preserved (except in the case that the levels of the first grouping factor are strictly nested within the levels of the second grouping factor). While the models are theoretically identical, the latter is computationally more efficient as the largest diagonal block preserves its structure. In fact, for the `insteval` dataset, the former order leads to approximately 4.5 million non-zero entries in the Cholesky factor, whereas choosing as first the random effect with the largest number of levels leads to a Cholesky factor with approximately 775,000 non-zero elements, which is an order of magnitude lower. In practice this change gives an order of magnitude difference between the runtimes for fitting the two versions of the model for the `insteval` data.

Because the [1,1] block of $\mathbf{L}$ is diagonal and the [1,1] block of $\mathbf{\Lambda}_\theta$ is a multiple of the identity, the remaining blocks in the first column of blocks of $\mathbf{L}$ will have exactly the same sparsity structure as the corresponding blocks of $\mathbf{A}$. For example, the [2,1] block of $\mathbf{A}$ is stored as a sparse matrix because only about 2% (73421) of the $2792 \times 1128 = 3352416$ elements of that matrix are non-zero. The [2,1] block of $\mathbf{L}$ has exactly the same sparsity structure because each element of that block of $\mathbf{L}$ is a scalar multiple of the corresponding element of that block of $\mathbf{A}$.

4 Application to Data from an Observational Study

We consider a large observational study – ratings of movies by users of movielens.org made available at the grouplens.org download site (Harper and Konstan, 2016) as the *MovieLens 32M Dataset* (`ml-32m`), consisting of roughly 32 million ratings of over 80,000 movies by over 200,000 users. The purpose of this section is to study which dimensions of the data have the greatest effect on the amount of memory used to represent the model and the time required to fit a model. To do so we use a simple model:

```
frm = @formula(rating ~ 1 + (1 | userId) + (1 | movieId))
fit(LinearMixedModel, frm, dataset)
```

where `dataset` will be different subsets of the full data explained below.

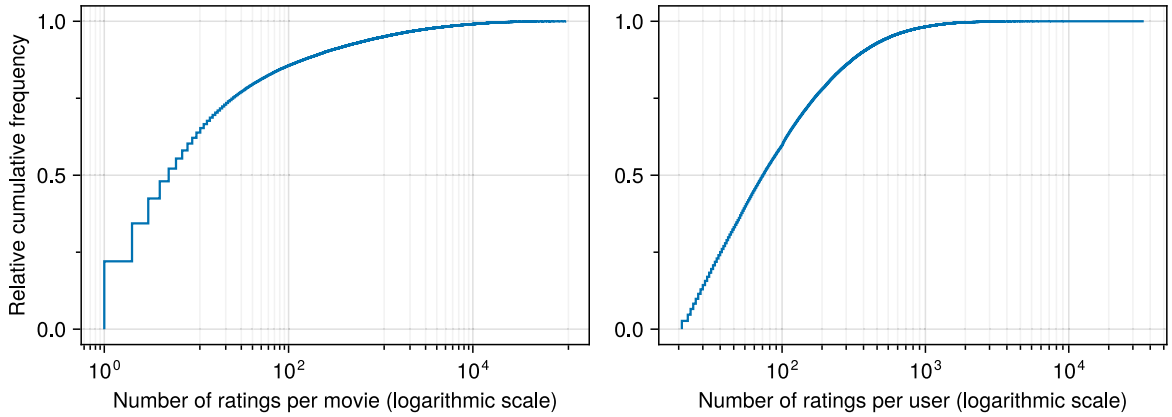


Figure 3: Empirical cumulative distribution plots of the number of ratings per movie and per user. The horizontal axes are on a logarithmic scale.

4.1 Structure of the Data

The `ratings.csv` file in the `ml-32m.zip` file downloaded from the Grouplens site provides a table of about 32 million ratings with the corresponding movie number and user number. Creating a `ratings` data frame from this file and counting the number of ratings for each movie id provides the `movies` data frame with columns `movieId` and `mnrtngs`. Similarly counting the number of ratings for each user provides the `users` data frame with columns `userId` and `unrtngs`. Joining these two derived data frames with the original `ratings` data frame (on `movieId` and `userId`, respectively) allows for the `ratings` data frame to be subsetted according to a minimum number of ratings per user and/or per movie. The supplemental materials contain a Julia script, `ml32-download.jl` to perform the downloading and extraction of the csv file and creation of the subsequent tables.

The data from this observational study are extremely unbalanced with respect to the grouping factors, `userId` and `movieId`, as shown in the empirical cumulative distribution function (ecdf) plots in Figure 3. (Note that the horizontal axis in each panel is on a logarithmic scale.)

This selection of ratings was limited to users who had provided at least 20 ratings.

In this collection of about 32 million ratings, over 22% of the movies are rated only once. The median number of ratings is 5 but the maximum is about 103,000.

The ecdf plot of the number of ratings per user shows a similar pattern to that of the movies — a few users with a very large number of ratings and many users with just a few ratings.

For example, the minimum number of movies rated is 20 (due to the inclusion constraint); the median number of movies rated is around 70; but the maximum is over 33,000 (which is a lot of movies–over 30 years of 3 movies per day every day–if this user actually watched all of the movies they rated).

Movies with very few ratings provide little information about overall trends or even about the movie being rated. We can imagine that the "shrinkage" of random effects for movies with just a few ratings pulls their adjusted rating strongly towards the overall average.

Similarly, users who rate very few movies add little information, even about the movies that they rated, because there isn't sufficient information to contrast a specific rating with a typical rating for the user.

One way of dealing with the extreme imbalance in the number of observations per user or

Table 5: Summary of results for the `ml-32m` data modeling run speeds. The `mc` column is the movie cutoff, `uc` is the user cutoff, `nratings`, `nusers` and `nmvie` are, respectively, the number of observations, users and movies in the reduced data set, `modelsz` and `L22sz` are the model size and `l[2,2]` size (in GiB), `nv` is the number of objective function evaluations until convergence, `fittime` and `mnevtm` are the number of seconds required to fit the model and the mean time to perform a single objective function evaluation.

mc	uc	nratings	nusers	nmvie	modelsz	L22sz	nv	fittime	mnevtm
1	20	32000204	200948	84432	29.4433	26.557	30	10737.6	357.92
1	40	30433400	144848	84205	29.1536	26.4144	35	12415.6	354.731
1	80	27569316	94380	83897	28.7014	26.2215	30	10559.3	351.978
2	20	31981597	200948	65825	19.0254	16.1417	39	8001.65	205.171
2	40	30415014	144848	65819	18.8754	16.1387	33	6802.02	206.122
2	80	27551218	94380	65799	18.6064	16.1289	31	6455.5	208.242
5	20	31921467	200948	43884	10.0515	7.17435	32	3378.6	105.581
5	40	30355623	144848	43884	9.90457	7.17435	40	4119.12	102.978
5	80	27492886	94380	43884	9.64538	7.17435	32	3287.72	102.741
10	20	31842705	200948	31961	6.67434	3.80552	33	2448.78	74.2054
10	40	30277758	144848	31961	6.52746	3.80552	37	2779.28	75.1157
10	80	27416310	94380	31961	6.26838	3.80552	39	2850.05	73.0783
15	20	31777786	200948	26428	5.46465	2.60199	35	2293.55	65.5301
15	40	30213583	144848	26428	5.31783	2.60199	30	1971.5	65.7168
15	80	27353366	94380	26428	5.05886	2.60199	30	1994.77	66.4925
20	20	31725920	200948	23350	4.88908	2.0312	34	2139.58	62.9288
20	40	30162330	144848	23350	4.74232	2.0312	33	2071.76	62.7807
20	80	27303019	94380	23350	4.48344	2.0312	33	2072.8	62.812
50	20	31498689	200947	16034	3.79464	0.957792	51	2703.92	53.018
50	40	29938038	144848	16034	3.64814	0.957792	37	1944.0	52.5406
50	80	27083536	94380	16034	3.38968	0.957792	36	1868.46	51.9018

per movie is to set a minimum number of ratings for a user or for a movie to be included in the data used to fit the model.

4.1.1 Models Fit with Lower Bounds on Ratings per User and per Movie

We fit a simple model to this dataset using different thresholds on the number of ratings per movie and the number of ratings per user. These fits were performed on a cloud instance with a generous amount of memory (128 GiB) and using 12 threads and the Intel Math Kernel Library (MKL) for numerical linear algebra. The results are summarized in Table 5.

4.1.2 Dimensions of the Model Versus Cut-Off Values

As shown in the first panel of Figure 4, the number of ratings varies from a little over 27 million to 32 million. For this set of cutoff values, the user cutoff has more impact on the number of

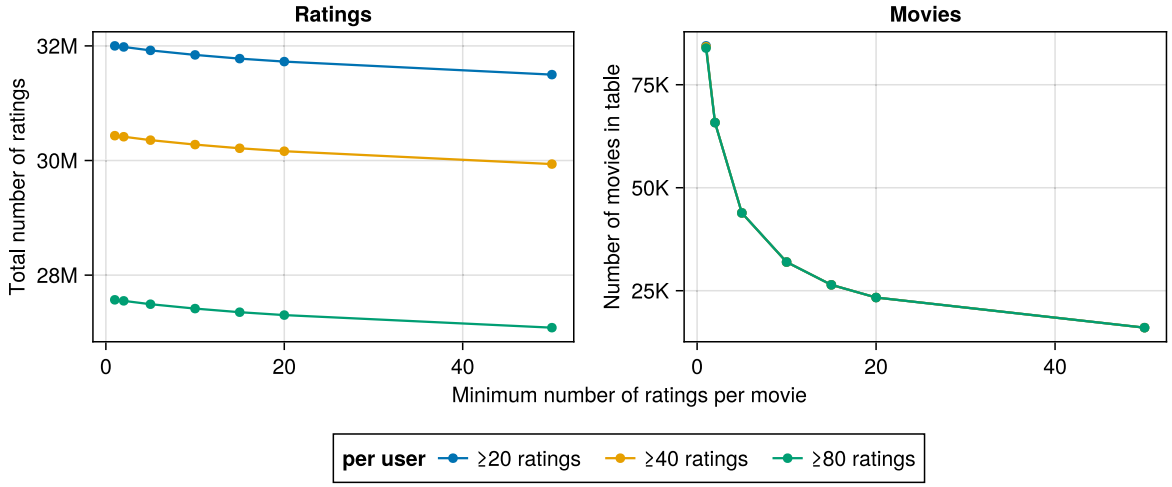


Figure 4: Number of ratings (left) and movies (right) in reduced table by movie cutoff and by user cutoff. There are three lines in the right panel but they are essentially superimposed.

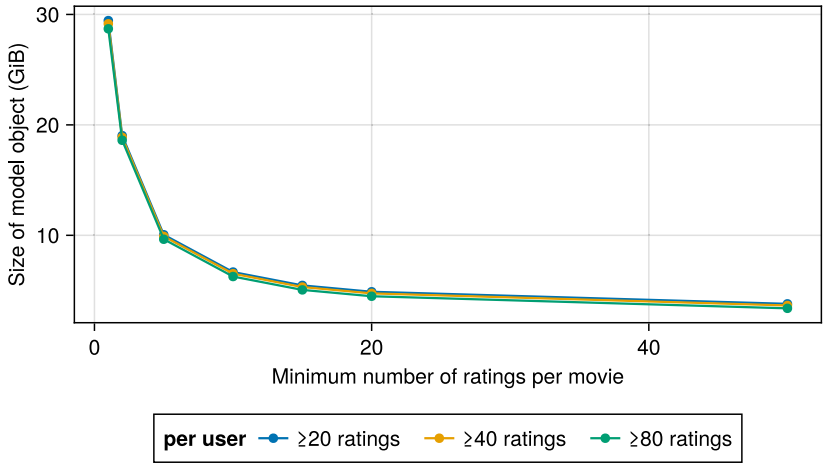


Figure 5: Memory footprint of the model representation by minimum number of ratings per user and per movie.

ratings in the reduced dataset than does the movie cutoff. A glance at Table 5 shows that the number of users is essentially a function of only the user cutoff.

The second panel in Figure 4 shows the similarly unsurprising result that the number of movies in the reduced table is essentially determined by the movie cutoff. (There are three different, but nearly coincident, lines in this panel.)

4.1.3 Memory Footprint of the Model Representation

Figure 5 shows that the overall memory size (the "memory footprint") of the model representation depends primarily on the movie cutoff. (There are also three distinct, but nearly coincident, lines in Figure 5, corresponding to the three user cutoff values.)

Figure 6 shows the dominance of the `L[2,2]` block, which is a dense lower triangular matrix (in RFP format), in the overall memory footprint of the model. When all the movies are included

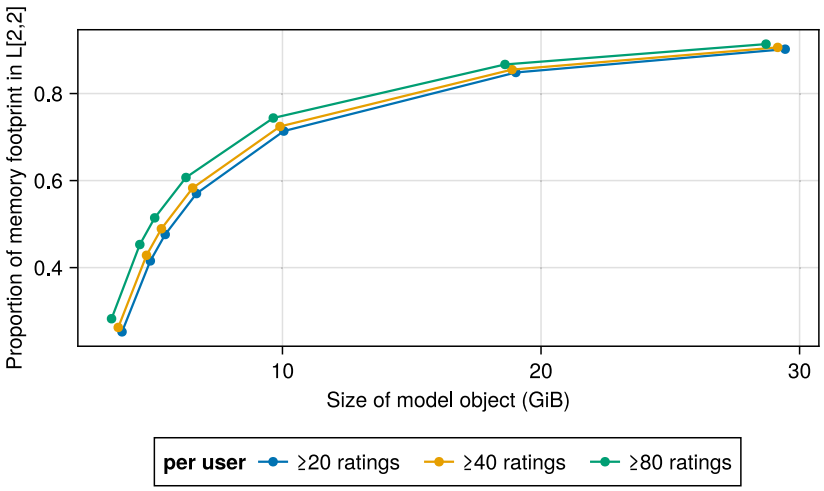


Figure 6: Proportion of memory footprint of the model in L[2,2] versus the overall model size (GiB). The dots are at different movie cutoff (mc) values, with the largest memory footprints at mc=2 and the smallest at mc=50.

in the data to which the model is fit the total memory footprint is over 30 GiB, and over 80% of that memory is for `L[2,2]`. Even when requiring a minimum of 80 ratings per movie, `L[2,2]` accounts for about 35% of the memory footprint.

The fact that the `[2,2]` block of `A` is diagonal but the `[2,2]` block of `L` is dense lower-triangular is described as "fill-in" and leads to a somewhat unintuitive conclusion. The memory footprint of the model representation depends strongly on the number of movies, less strongly on the number of users and almost not at all on the number of ratings (Figure 5).

Note that although the dimensions of the `L[2,1]` block are larger than those of the `L[2,2]` block its memory footprint is smaller than that of the `L[2,2]` block, because it is stored as a sparse matrix. The matrix is about 98% zeros or, equivalently, a little over 2% nonzeros, which makes the sparse representation much smaller than the dense representation. Also, the number and positions of the non-zeros in `L[2,1]` is exactly the same as those in `A[2,1]`. That is, there is no fill-in in the `[2,1]` block

The fact that the memory footprint of the model is dominated by that of the `[2,2]` block makes the use of the RFP format for that block effective in reducing memory usage.

In general, for models with scalar random effects for two incompletely crossed grouping factors, the memory footprint depends strongly on the smaller of the number of levels of the grouping factors, less strongly on the larger number of levels, and almost not at all on the number of observations.

4.1.4 Speed of Log-Likelihood Evaluation

The time required to fit a model to large data sets is dominated by the time required to evaluate the log-likelihood during the optimization of the parameter estimates. The mean time per evaluation is given in the `mnevtm` column of Table 5. Also given is the number of evaluations to convergence, `nv`, and the total time to fit the model, `fittime`. The reason for considering `mnevtm` in addition to `fittime` and `nv` is because the `mnevtm` for one model, relative to other models, is reasonably stable across computers whereas `nv`, and hence, `fittime`, can be affected by seem-

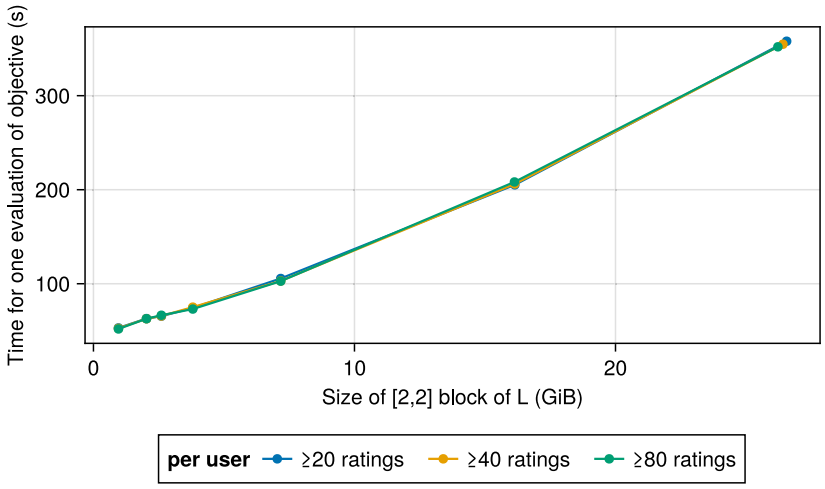


Figure 7: Evaluation time for the objective (s) versus size of the [2,2] block of L (GiB).

ingly trivial variations in function values resulting from different implementations of low-level calculations, such as the BLAS (Basic Linear Algebra Subroutines). (Floating point arithmetic is not guaranteed to be associative and custom BLAS implementations may rearrange the order of operations when evaluating products and sums.)

That is, we can't expect to reproduce `nv` exactly when fitting the same model to the same data but on different computers or with slightly different versions of software. However, we can expect the pattern in `mnevtm` with respect to the user and movie cutoffs to be reproducible.

As shown in Figure 7 the evaluation time for the objective is predominantly a function of the size of the `[2,2]` block of `L`.

5 Further Comments and Directions for Research

5.1 REML Estimation

Throughout the discussion we have considered maximum likelihood estimation, which is the default estimation criterion. However, it is also possible to fit mixed models using the restricted maximum likelihood (REML) criterion by setting the optional argument `REML=true` in the call to the `fit` function. Briefly, the maximum likelihood estimates of the random effects variance components are biased. The REML estimates provide less biased estimates of variance components. For more details see (see Bates et al., 2015, Section 3.4). For instance:

```
mreml = fit(MixedModel, form, dat; REML=true, progress=false);
print(mreml)
```

```
Linear mixed model fit by REML
 y ~ 1 + service + (1 | d) + (1 | s) + (1 | dept) + (0 + service | dept)
 REML criterion at convergence: 237658.6094518444
```

```
Variance components:
            Column    Variance  Std.Dev.   Corr.
s        (Intercept)  0.1053183 0.3245278
d        (Intercept)  0.2624350 0.5122841
dept     (Intercept)  0.0030480 0.0552084
         service      0.0256223 0.1600696   .
Residual              1.3850031 1.1768616
 Number of obs: 73421; levels of grouping factors: 2972, 1128, 14

  Fixed-effects parameters:
──────────────────────────────────────────────────
                 Coef.  Std. Error       z  Pr(>|z|)
──────────────────────────────────────────────────
(Intercept)   3.27771     0.0242422  135.21    <1e-99
service      -0.0502815   0.0457775   -1.10    0.2720
──────────────────────────────────────────────────
```

The estimates obtained above are the same as those obtained from `lme4::lmer` function in R.

We used the formula `y ~ 1 + service + (1 | d) + (1 | s) + (1 | dept) + (0 + service | dept)` to demonstrate that *MixedModels.jl* amalgamates random effects terms with the same grouping factor. In practice we can use the more concise formula: `y ~ 1 + service + (1 | d) + (1 | s) + zerocorr(service | dept)`. That is, the term `zerocorr(service | dept)` is equivalent to adding the two terms `(1 | dept) + (0 + service | dept)`, and the same as the *lme4* formula term `(1 + service || dept)`.

Although we only demonstrated the fitting of linear mixed models, *MixedModels.jl* can also fit generalized linear mixed models including logistic, probit, poisson, and negative binomial models. To fit a probit model for instance we use the following syntax:

```
probmod = fit(MixedModel, frm, dt, Bernoulli(), ProbitLink())
```

where `frm` is a formula and `dt` is a dataset. Refer to the *MixedModels.jl* documentation for more details.

5.2 Evaluation of the Gradient of the Profiled Objective

As mentioned in Section 2, the profiled log-likelihood is an affine function of the logarithms of the diagonal elements of the blocked Cholesky factor, $\mathbf{L}_{\boldsymbol{\theta}}$. The detailed derivation of the profiled REML criterion in Bates et al. (2015) shows that it too is an affine function of the logarithms of the diagonal elements of $\mathbf{L}_{\boldsymbol{\theta}}$. For the purposes of evaluating the gradient of the profiled log-likelihood or the profiled REML criterion we only must consider the part of the objective that is a linear combination of the logarithms of the diagonal elements of $\mathbf{L}_{\boldsymbol{\theta}}$, which can be written as

$$f(\boldsymbol{\theta}) = \sum_{j=1}^{q+p+1} w_j (2 \log \ell_{jj}), \tag{16}$$

where the weights, w_j, $j = 1, \ldots, q+p+1$ for the profiled log-likelihood are 1 for $j = 1, \ldots, q$ (the random-effects rows), 0 for $j = q+1, \ldots, q+p$ (the fixed-effects rows), and n for $j = q+p+1$

(the response row). As shown in Bates et al. (2015) the weights are 1 for $j = 1, \ldots, (q+p)$ (both random- and fixed-effects rows) and $n - p$ for $j = q + p + 1$ for the profiled REML criterion.

5.2.1 Symbolic Differentiation of the Cholesky Factor

Following Murray (2016), we use symbolic differentiation of the defining relationship

$$\boldsymbol{\Omega}_{\boldsymbol{\theta}} = \mathbf{L}_{\boldsymbol{\theta}}\mathbf{L}_{\boldsymbol{\theta}}^{\top} \tag{17}$$

to produce

$$\dot{\boldsymbol{\Omega}} = \dot{\mathbf{L}}\mathbf{L}^{\top} + \mathbf{L}\dot{\mathbf{L}}^{\top}, \tag{18}$$

where the dot indicates partial differentiation with respect to an element of $\boldsymbol{\theta}$ (in this and subsequent equations we drop the subscript $\boldsymbol{\theta}$ on $\boldsymbol{\Omega}$ and $\mathbf{L}$).

Premultiplying Equation (18) by $\mathbf{L}^{-1}$ and postmultiplying by $\mathbf{L}^{-\top}$ produces

$$\mathbf{L}^{-1}\dot{\boldsymbol{\Omega}}\mathbf{L}^{-\top} = \mathbf{L}^{-1}\dot{\mathbf{L}} + \dot{\mathbf{L}}^{\top}\mathbf{L}^{-\top} = \mathbf{L}^{-1}\dot{\mathbf{L}} + \left(\mathbf{L}^{-1}\dot{\mathbf{L}}\right)^{\top}. \tag{19}$$

Because $\mathbf{L}$, $\dot{\mathbf{L}}$, and $\mathbf{L}^{-1}$ are all lower triangular, $\mathbf{L}^{-1}\dot{\mathbf{L}}$ is also lower triangular and the jth diagonal element of $\mathbf{L}^{-1}\dot{\boldsymbol{\Omega}}\mathbf{L}^{-\top}$ is exactly the partial derivative of $2\log\ell_{jj}$. Thus the terms in the gradient of the profiled objective are

$$\frac{\partial f}{\partial \theta_i} = \operatorname{tr}\left(\mathbf{W}\mathbf{L}^{-1}\frac{\partial \boldsymbol{\Omega}}{\partial \theta_i}\mathbf{L}^{-\top}\right) = \operatorname{tr}\left(\mathbf{L}^{-\top}\mathbf{W}\mathbf{L}^{-1}\frac{\partial \boldsymbol{\Omega}}{\partial \theta_i}\right) \quad i = 1, \ldots, r, \tag{20}$$

where r is the dimension of $\boldsymbol{\theta}$ and $\mathbf{W}$ is a diagonal matrix of the weights in f.

In the final expression of Equation (20), which was suggested by Antropic's Claude-fable AI, the symmetric matrix $\mathbf{L}^{-\top}\mathbf{W}\mathbf{L}^{-1}$ only needs to be evaluated once for a given value of $\boldsymbol{\theta}$. That is, it does not depend upon $i = 1, \ldots, r$. Thus the cost of evaluating this expression can be amortized over all the elements of the gradient.

Preliminary investigations show that gradient evaluations are most effective in reducing the effort to optimize the objective when the dimension of $\boldsymbol{\theta}$ is moderate to large.

5.3 A[2,1] as a Biadjacency Matrix

In both of the examples we have considered, the first and second random-effects terms are simple scalar terms and each observation corresponds to a unique combination of levels of these two factors. For example, in the `movielens` data each rating corresponds to a unique combination of a user and a movie.

In cases like these, the $\mathbf{A}[2, 1]$ matrix is the biadjacency matrix for a bipartite graph (U, V, E) where the nodes U are the individual users, the nodes V are the individual movies and the edges, E, link a movie to a user who rated it. The total number of edges in the bipartite graph is n, the number of observations in the data set. Furthermore, the diagonal elements in the diagonal matrices $\mathbf{A}[1, 1]$ and $\mathbf{A}[2, 2]$ are the degrees of the vertices U and V, respectively. They are also the column- and row-sums of $\mathbf{A}[2, 1]$. It may be possible to derive new computational approaches from these relationships using graph theory, an avenue we hope to explore in future research.

6 Summary and Discussion

We demonstrated how the *MixedModels.jl* package can be used to fit models with two distinct grouping factors (the `ml-32m` example) and three distinct grouping factors (the `insteval` example). Fitting models with a single grouping factor or with more than three grouping factors is straightforward.

The key contribution of *MixedModels.jl* is the new fitting procedure based on the augmented matrix $\mathbf{A}$ (Equation (15)) and the simplified formula for the profiled loglikelihood (Equation (14)). As a result of this simplification, we do not need to compute the conditional mode of the random effects ($\tilde{\mathbf{u}}$ or $\tilde{\mathbf{b}}$) for each evaluation of the objective function. This means we avoid roughly $O(n)$ computations at each evaluation as compared to previous implementations of mixed effects models, making the algorithm more scalable. In fact the conditional modes are computed only once at the converged values of the underlying parameter $\boldsymbol{\theta}$. Moreover, at each iteration we only need to use the augmented Gram matrix $\mathbf{A}$ (Equation (15)), which is a blocked, symmetric, sparse matrix of size $q + p + 1$ and the covariance parameter vector, $\boldsymbol{\theta}$, to form the blocked Cholesky factor, $\mathbf{L}$. This update is performed in-place in the storage for $\mathbf{L}$.

Secondly, by rearranging the random effects terms so that the grouping factor with the greatest number of random effects corresponds to the $[1, 1]$ block of the Cholesky factor, we reduce the "fill-in" of the Cholesky factor. These two changes together contribute to a substantial improvement in computational efficiency.

Finally, the multiple dispatch feature of the Julia language allows us to use specialized algorithms for different kinds of matrices instead of relying on generic algorithms. This is relevant because we have both sparse and dense operations in different blocks of the Cholesky factor. By dispatching on the type of the block, we can create performant algorithms for each combination of matrix types that can occur.

Supplementary Material

The supplementary material for this article is available on the official website of the Journal of Data Science at https://jds-online.org/. It may include additional results, appendices, tables, figures, code, and data.

Acknowledgement

All figures were produced using the *AlgebraOfGraphics.jl* package with the *CairoMakie.jl* backend in the *Makie.jl* graphics system (Danisch and Krumbiegel, 2021). Julius Krumbiegel and Anshul Singhvi provided invaluable help and feedback with *Makie.jl*. The plots in this work would not have been possible without them. Finally, we would like to thank the Julia community for bugfixes and answering our questions in the community support channels.

Funding

This research was supported by the Center for Interdisciplinary Research, Bielefeld (ZiF) Cooperation Group "Statistical models for psychological and linguistic data". We would like to thank Ben Bolker for helpful comments on the manuscript.

References

Baayen RH, Davidson DJ, Bates DM (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59: 390–412. https://doi.org/10.1016/j.jml.2007.12.005

Bates D, Maechler M, Bolker BM, Walker S (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1): 1–48. https://doi.org/10.18637/jss.v067.i01

Bezanson J, Edelman A, Karpinski S, Shah VB (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1): 65–98. https://doi.org/10.1137/141000671

Chen Y, Davis TA, Hager WW, Rajamanickam S (2008). Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software*, 35(3): 1–14. https://doi.org/10.1145/1391989.1391995

Danisch S, Krumbiegel J (2021). Makie.jl: Flexible high-performance data visualization for Julia. *Journal of Open Source Software*, 6(65): 3349. https://doi.org/10.21105/joss.03349

Fühner T, Granacher U, Golle K, Kliegl R (2021). Age and sex effects in physical fitness components of 108,295 third graders including 515 primary schools and 9 cohorts. *Scientific Reports*, 11(1): 17566. https://doi.org/10.1038/s41598-021-97000-4

Gustavson FG, Wasniewski J, Dongarra JJ, Langou J (2010). Rectangular full packed format for Cholesky's algorithm: Factorization, solution and inversion, *ACM Transactions on Mathematical Software*, 37(2): 1–21. https://doi.org/10.1145/1731022.1731028

Harper FM, Konstan JA (2016). The MovieLens datasets. *ACM Transactions on Interactive Intelligent Systems*, 5(4): 1–19. https://doi.org/10.1145/2827872

Murray I (2016). Differentiation of the Cholesky decomposition. https://arxiv.org/abs/1602.07527.

Pearson H, Ledford H, Hutson M, Van Noorden R (2025). Exclusive: The most-cited papers of the twenty-first century. *Nature*, 640(8059): 588–592. https://doi.org/10.1038/d41586-025-01125-9