Estimation of Ordinal Mixed Models by the Template Model Builder (TMB) Package in R

Manja Gersholm Grønberg* Sofie Pødenphant Jensen*
Line Katrine Harder Clemmensen*

Abstract
Ordinal mixed models constitute a flexible class of models to analyze ordinal data from various fields. Examples are data that originates from the heavily used online reviewing systems of movies, music, pain reports and customer surveys. Ordinal mixed models take the ordinal nature of data into account and allow modelling of more complex dependency structures. Furthermore, they provide a flexible and simple skeleton to model the information at hand, i.e. selecting terms in a modelling process as known from classical statistics. Ordinal mixed models can among others be estimated through the Laplace Approximation, for example using the ordinal package in R. However, this package is not optimized for large datasets. We propose to implement ordinal mixed models with the Template Model Builder (TMB) package in R. TMB provides a simple and flexible framework that enables fast optimization of the Laplace Approximation to the marginal log-likelihood. It is optimized to make fast computations for models with both many random effects and parameters (Kristensen et al. 2016). We compare an implementation using the TMB package to the ordinal package through timings on both simulated and real data. We find that an implementation with TMB gives substantial speed-ups for models with many observations, parameters and random effects and allow estimation of ordinal mixed models with even larger datasets.

Key Words: Ordinal Mixed Models, Large Data, TMB

1. Introduction

The size of data is ever increasing. This includes ordered categorical data, or ordinal data, which for example occur in recommender systems, customer surveys, and pain assessments. A commonly used approach is to treat this type of data as continuous (Agresti 2012). However, this is too restrictive as this implies a fixed distance between the classes dependent on the arbitrary class-coding and ignores the categorical nature (Tutz 2011). Treating ordinal data as categorical is not appropriate either, as this ignores the information on order (Tutz 2011).

Ordinal mixed models are regression models that are designed to model the ordinal nature of data appropriately. The models allows for complex dependency structures that makes it possible to relax the assumption of independent observations whenever this is relevant.
These models can be estimated by the maximum likelihood method optimizing the Laplace Approximation of the marginal log-likelihood. The `clmm` function from the `R` package `ordinal` can estimate ordinal mixed models by this method. At the time of writing, this package has 683 citations on google scholar which testify to the popularity of this package. However, this package is not optimized for large data sets.
The `TMB` package in `R` offers fast and efficient optimization of the Laplace Approximation. `TMB` is optimized to estimate models with both many random effects and parameters (Kristensen et al. 2016). This makes `TMB` ideal to extend the size of data sets for which ordinal

*Technical University of Denmark, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800, Kgs. Lyngby, Denmark mixed models can be estimated within reasonable time.

The proceeding is arranged as follows: section 2 introduces ordinal mixed models, estimation of them by the Laplace Approximation and the two `R` packages `ordinal` and `TMB`. Section 3 describes the experiments that are used to compare computation times between the two packages and section 4 presents the results of the experiments. The last section, section 5, finalizes this proceeding with a discussion and a conclusion.

2. Methods

2.1 Ordinal Mixed Models

Ordinal Mixed Models also named cumulative link mixed models are motivated by the assumption that the ordinal response is build from an underlying continuous latent variable (Tutz 2011). Denote the continuous latent variable y_i^* for observation i. The continuous latent variable transforms into the ordinal response by introducing $C+1$ ordered thresholds

$$-\infty \equiv \theta_0 \leq \theta_1 \leq \ldots \leq \theta_{C-1} \leq \theta_C \equiv \infty$$

that splits the real axis into C separate regions, where C is the number of ordered categories for the ordinal response variable. The ordinal response for observation i, y_i, is thus $C-1$ if y_i^* falls between θ_{C-1} and θ_C (Tutz 2011). Note that θ_0 and θ_C are special, as they are fixed to be equivalent to $-\infty$ and ∞, respectively. Thus only $C-1$ thresholds are parameters.
For ordinal mixed models it is assumed that the continuous latent variable is modeled by a mixed model of the form

$$y_i^* = \mathbf{x}_i^\top \boldsymbol{\beta} + \mathbf{z}_i^\top \mathbf{u} + \epsilon_i, \tag{1}$$

where $\mathbf{u} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\tau}})$ and ϵ_i is assumed to follow a distribution with cumulative distribution function F. $\boldsymbol{\beta}$ contains the fixed effects parameters and is a vector of size p. $\mathbf{u}$ contains the random effects and is a vector of size q. $\mathbf{x}_i$ and $\mathbf{z}_i$ are rows of the design matrices for the fixed effects and the random effects, respectively. $\boldsymbol{\tau}$ is a vector of parameters that parameterize the variance-covariance matrix $\boldsymbol{\Sigma}$ (Christensen 2012). Ordinal mixed models are thus an extension of ordinal regression models (Mccullagh 1980) which include random effects.
The cumulative probability of observation i falling in the c'th category, γ_{ic}, is obtained by combining the link between the continuous latent variable and the ordinal response with (1), such that

$$\gamma_{ic} = P(y_i \leq c) = P(y_i^* \leq \theta_c) = P(\epsilon_i \leq \theta_c - \mathbf{x}_i^\top \boldsymbol{\beta} - \mathbf{z}_i^\top \mathbf{u}). \tag{2}$$

The formal formulation of ordinal mixed models follows directly from (2), see (Tutz 2011), and is presented as

$$\gamma_{ic} = F(\eta_{ic}), \quad \eta_{ic} = \theta_c - \mathbf{x}_i^\top \boldsymbol{\beta} - \mathbf{z}_i^\top \mathbf{u}, \quad i = 1, ..., N, \quad c = 0, ..., C. \tag{3}$$

For notational convenience and implementational ease, the notation for ordinal mixed models in (3) is changed into matrix notation. Christensen (2012) motivates the matrix formulation by introducing a k-notation. Let π_{ic} be the probability of observation i falling in category c and π_i be the probability of observation i falling in its actual category. The k-notation is then presented as

$$\pi_i = \gamma_{i1} - \gamma_{i2} \text{ where } \gamma_{ik} = \gamma_{ic} \text{ for } j = y_i - k + 1.$$

Using the k-notation rewrites (3) to

$$\gamma_{ik} = F(\eta_{ik}), \quad \eta_{ik} = \theta_{ik} - \mathbf{x}_i^\top \boldsymbol{\beta} - \mathbf{z}_i^\top \mathbf{u}, \quad k = 1, 2.$$

The matrix form then reads

$$\boldsymbol{\gamma}_k = F(\boldsymbol{\eta}_k), \quad \boldsymbol{\eta}_k = \mathbf{A}_k \boldsymbol{\theta} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u} + \mathbf{o}_k, \tag{4}$$

where $\mathbf{A}$ is a design matrix of dimension $N \times C$ that takes the value of 1 at the c'th position in row i if observation i is in category c and 0 otherwise. $\mathbf{X}$ and $\mathbf{Z}$ are the design matrices for the fixed effects and the random effects respectively and have dimensions $N \times p$ and $N \times q$. $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \dots & \theta_{C-1} \end{bmatrix}$ and has length $C - 1$. $\mathbf{A}_1 = \mathbf{A}[, -C]$ is the $\mathbf{A}$ matrix without the C'th column while $\mathbf{A}_2 = \mathbf{A}[, -1]$ is the $\mathbf{A}$ matrix without the first column. $\mathbf{o}_1 = \mathbf{A}[, C]\theta_0$ and $\mathbf{o}_2 = \mathbf{A}[, 1]\theta_C$ are offset vectors of size N. For implementational purposes, θ_0 and θ_C are simply large positive and negative numbers (Christensen 2012).

2.2 Estimation

This proceeding focuses on estimation of the ordinal mixed models by the maximum likelihood method. The maximum likelihood method requires specification of the marginal likelihood function. We obtain the marginal log-likelihood function for ordinal mixed models (4) by integrating or marginalizing out the random effects of the joint log-likelihood such that

$$\ell(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\tau}; \mathbf{y}) = \log \int_{\mathbb{R}^q} p_{\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\tau}}(\mathbf{y}, \mathbf{u}) d\mathbf{u} \tag{5}$$

$$= \log \int_{\mathbb{R}^q} p_{\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\tau}}(\mathbf{y}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u} \tag{6}$$

where

$$p_{\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\tau}}(\mathbf{y}|\mathbf{u}) = \boldsymbol{\pi}, \quad \boldsymbol{\pi} = \boldsymbol{\gamma}_1 - \boldsymbol{\gamma}_2,$$
$$p(\mathbf{u}) = (2\pi)^{-\frac{q}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp(-\frac{1}{2}\mathbf{u}^\top \boldsymbol{\Sigma}^{-1} \mathbf{u}).$$

and $\boldsymbol{\pi} = \begin{bmatrix} \pi_i & \dots & \pi_N \end{bmatrix}$ contains the probabilities of observation i falling in category c. The conditional distribution of the data given the random effects, $p_{\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\tau}}(\mathbf{y}|\mathbf{u})$, is derived by realising that the ordinal response can be viewed as multivariate. From this point of view the multivariate response for an observation i falling in category c, $\tilde{\mathbf{y}}_i$ is a vector of length C, that takes the value of 1 in the c'th entry and 0 otherwise. The response then follows a multinomial distribution $\tilde{\mathbf{y}}_i = \text{multinom}(1, \boldsymbol{\pi}_i)$ where $\boldsymbol{\pi}_i = \begin{bmatrix} \pi_{i1} & \dots & \pi_{iC} \end{bmatrix}$ (Christensen 2018).

The integral in (5) rarely has a closed-form solution. Direct optimization of the log-likelihood function to estimate the parameters is therefore rarely computationally feasible (Christensen 2012). Instead, a numerical approximation to the integral (5) is optimized. Several techniques exist with their respective pros and cons like the Laplace Approximation, Gauss-Hermite Quadrature, Adaptive Gauss-Hermite Quadrature and Monte Carlo Intergration (Christensen 2012). Here the focus is the Laplace Approximation to the marginal log-likelihood. The Laplace Approximation turns the original integration problem in to an optimization problem. It is derived by replacing the joint log-likelihood with a second order Taylor-expansion of the joint log-likelihood around the optimum of the joint log-likelihood with respect to the random effects $\hat{\mathbf{u}}$. The Laplace approximation to the marginal log-likelihood is then given by

$$\ell_{\text{LA}}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\tau}; \mathbf{y}) = \log p_{\boldsymbol{\theta},\boldsymbol{\beta},\boldsymbol{\tau}}(\mathbf{y}, \hat{\mathbf{u}}) + \frac{q}{2}\log(2\pi) - \frac{1}{2}\log|-\mathbf{H}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\tau}, \hat{\mathbf{u}})|,$$

where $\mathbf{H}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\tau}, \hat{\mathbf{u}})$ is the Hessian of the joint log-likelihood with respect to the random effects $\mathbf{u}$ evaluated at the optimum $\hat{\mathbf{u}}$ (Madsen and Thyregod 2011; Christensen 2012). Optimization of the Laplace Approximation may then be regarded as a nested optimization procedure. The inner optimization refers to the optimization of the joint log-likelihood with respect to the random effects to obtain $\hat{\mathbf{u}}$ and thereby the Laplace Approximation to the marginal log-likelihood. The outer optimization refers to optimization of the Laplace approximation itself to obtain the parameter estimates (Fournier et al. 2011; Madsen and Thyregod 2011).

2.3 R packages: `ordinal` and `TMB`

The R-package `ordinal` enables estimation of ordinal mixed models of various forms including multiple random effects, scale effects and threshold structures, through the function `clmm`. The models are estimated by the maximum likelihood method using either the Laplace Approximation or Adaptive Gauss-Hermite Quadrature to approximate the integral in (5) (Christensen 2019). However, for this proceeding only the Laplace Approximation is considered since this in general is the computationally fastest method of the two (Christensen 2019).

The R-package `TMB` is not exclusively oriented towards estimation of ordinal mixed models but is a more general package that enables fast optimization of the Laplace approximation to the marginal log-likelihood. `TMB` can therefore estimate a range of different models and is thus very flexible. One of the main advantageous of TMB is that it uses automatic differentiation (AD) to provide up to third order derivatives of the joint log-likelihood (Kristensen et al. 2016). Automatic differentiation (AD) exploits that all computations may be seen as a sequence of simple operations, such as addition, subtraction etc., and combines this with the chain rule to provide an efficient differentiation method. The bookkeeping of the simple operations to represent the function is often referred to as a tape (Madsen and Thyregod 2011). The tape can be run in two main modes: forward mode and reverse mode (Kristensen et al. 2016). Pros and cons of the two modes wrt. computation time are discussed later. The Laplace Approximation cannot trivially be split into a sequence of simple operations because the Laplace approximation is a product of an optimization problem itself. It may therefore be difficult to see how AD can be used to obtain the gradient of the Laplace Approximation used for optimization. However, since the Laplace approximation is obtained using second order derivatives of the joint log-likelihood function, the gradient of the Laplace Approximation can be obtained by use of up to third order derivatives (Jensen 2019).

The primary computational difference between the `ordinal` package and the `TMB` package is the calculations of derivatives used for the inner and outer optimizations.

The inner optimization (optimization of the joint log-likelihood wrt. the random effects to obtain the Laplace approximation to the marginal log-likelihood) employs a Newton's method for both packages. This requires the gradient and the Hessian of the joint log-likelihood. The `ordinal` package provides the analytic gradient and Hessian (Christensen

2012) whereas TMB obtains them by AD (Kristensen et al. 2016).

A quasi-Newton method performs the outer optimization (optimization of the Laplace Approximation itself) using the `nlminb` package in R in both the `ordinal` and TMB package. This requires the gradient of the Laplace approximation to the marginal log-likelihood. The `ordinal` package approximates the gradient by the finite difference methods (Christensen 2012) whereas the TMB package calculates the gradient by AD (Kristensen et al. 2016).

If the number of parameters is s then the cost, in terms of computations, required to approximate the gradient using the finite difference method corresponds to the cost of $s+1$ evaluations of the log-likelihood (Madsen and Thyregod 2011). The cost of forward mode AD is less than the cost of $4s$ evaluations of the log-likelihood (Skaug and Fournier 2006) whereas the cost of reverse mode AD is less than the cost of 4 evaluations of the log-likelihood (Griewank 2000; Madsen and Thyregod 2011). Note that the cost of reverse mode AD does not depend on the number of parameters and therefore obtains an advantage computationally (Madsen and Thyregod 2011). However, the drawback of reverse mode AD is the memory usage. The memory required for reverse mode can be huge, if the sequence of simple operations needed to represent the function is long, as all intermediate steps need to be stored (Madsen and Thyregod 2011; Jensen 2019).

Both the `ordinal` package and the TMB package explores the sparsity of the Hessian of the random effects to make the computations more efficient (Kristensen et al. 2016; Christensen 2012). Additionally, TMB also gives easy access to parallelization of the code, using the type `parallel_accumulator` that gives potential for speed-ups (Kristensen et al. 2016).

Finally, it should be mentioned that the `ordinal` package is very user-friendly compared to the TMB package. The syntax to specify models in the `ordinal` package is similar to the well-known notation from the `lme4` package and is therefore very easy to use (Christensen 2019). The TMB package requires specification of the negative joint log-likelihood function in a C++ file and an R-file to do data pre- and post-processing along with optimization of the Laplace Approximation (outer optimization) (Kristensen et al. 2016). Using TMB therefore requires both C++ experience and a much deeper understanding of the theory behind ordinal mixed models. However, with a wrapper for the TMB interface, the TMB implementation could be just as user-friendly as the `ordinal` package.

3. Experiments

The experiments compare computation times of ordinal mixed models estimated by the `ordinal` package to an implementation of these models with the TMB package. To this end, two simple models are considered. The models consist of two additive effects. The first model consider one factor random and the other factor fixed such that:

$$\gamma_{ic} = F(\eta_{ic}), \quad \eta_{ic} = \theta_c - \alpha(\texttt{fac1}_i) - b(\texttt{fac2}_i), \quad i = 1, ..., N, \quad c = 0, ..., C \tag{7}$$

where $b \sim N(0, \sigma_b^2)$.
The second model considers both effects random, giving the model

$$\gamma_{ic} = F(\eta_{ic}), \quad \eta_{ic} = \theta_c - a(\texttt{fac1}_i) - b(\texttt{fac2}_i), \quad i = 1, ..., N, \quad c = 0, ..., C \tag{8}$$

where $a \sim N(0, \sigma_a^2)$ and $b \sim N(0, \sigma_b^2)$. For the first block of experiments the data is simulated according to the models. Specifications of the number of observations and the number of levels for the factors in the simulated datasets are seen in table 1 in the appendix.

The second block of experiments is performed on subsets of the training set from the Netflix Prize Challenge (Netflix 2009). By working with non-simulated data that does not fit the model under consideration, the second block of experiments better resemble computation times in reality. The Netflix Prize data set originates from a real-life recommender system and consist of millions of observations where users have rated movies on a 5-star scale. It additionally includes timestamps for each rating (Netflix 2009). However, since the objective of these experiments solely concentrates on computation time we will disregard the timestamp information and build two simple models from the effects of user and movie. The first model considers the effect of user random and the effect of movie fixed while the second model considers both effects random. Thereby the first model is equal to model (7) and the second model is equal to model (8), where factor 1 is related to movies and factor 2 is related to users. The subsets of the Netflix Prize data set are constructed to be relatively dense, to yield as large data sets as possible. Further specifications of number of observations, number of users and number of movies for each subset is seen in table 2 in the appendix.

All computation times are obtained by taking the average of 10 repeated computations. The computations are run on a MacBook Pro with a 2.8 GHz Quad-Core Intel Core i7 processor, 16GB 2133 MHz LPDDR3 memory, and Intel Iris Plus Graphics 655 1536 MB graphics.

4. Results

The results compare computation times for the `ordinal` package to computation times for the `TMB` package. The comparisons consider a version where the Hessian is calculated at optimum and a version where it is not. Additionally, the `TMB` package allows for easy parallelization as described in section 2.3. Thus computation times using the parallelization option with 4 threds are also included for `TMB`.

Figure 1 and 3 show the average computation times of 10 repetitions when estimating model (7) and model (8), respectively, for the simulated data. The greatest differences in computation time are seen for model (7) which considers one factor as fixed and one factor as random (figure 1). `TMB` is 54.17 times faster than `ordinal` (without Hessian), with 150 levels of both the fixed factor and the random factor and thereby $22,500$ observations. The faster computation times means that this model can now be estimated with $160,000$ observations and 400 levels of each factor within 10 minutes.
The computation times are less different for model (8), figure 3. However here, `TMB` is 3.42 times faster than `ordinal` (without Hessian), with 450 levels of both of the random factors and thereby $202,500$ observations. `TMB` thus enables estimation of model (7) with up to 800 levels of each of the random factors and $640,000$ observations within 10 minutes. Parallelization, using `TMB`'s `parallel_accumulator()` does not seem to yield a significant speed-up for model (7) but yields a notable speedup for model (8). However, notice from figures 2 and 4 that the differences in computation time for the smallest datasets are small.

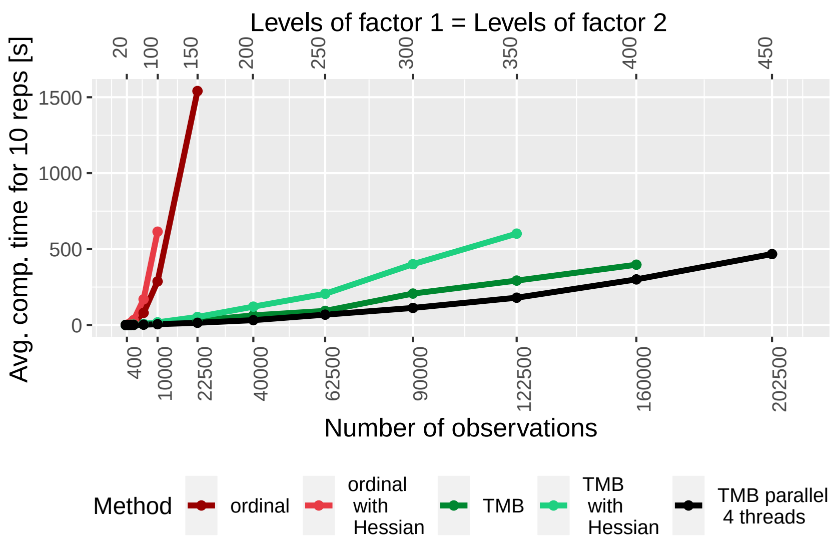

Figure 1: Average computation times of 10 repetitions when estimating model (7) for the simulated data. Computation times for the `ordinal` package are indicated with red. Computation times for `TMB` are indicated with green and black for parallel computations. ”with Hessian” indicates that the Hessian is calculated at optimum.

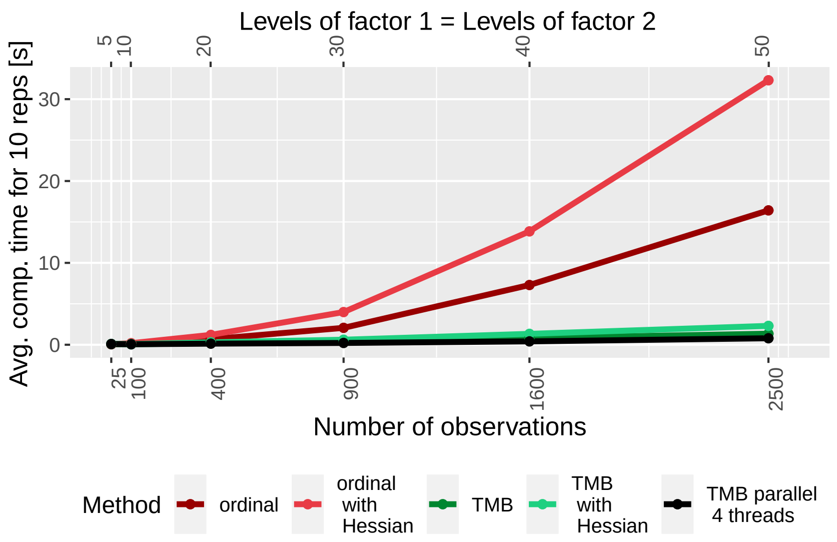

Figure 2: Zoom of figure 1. Average computation times of 10 repetitions when estimating model (7) for the simulated data. Computation times for the `ordinal` package are indicated with red. Computation times for `TMB` are indicated with green and black for parallel computations. ”with Hessian” indicates that the Hessian is calculated at optimum.

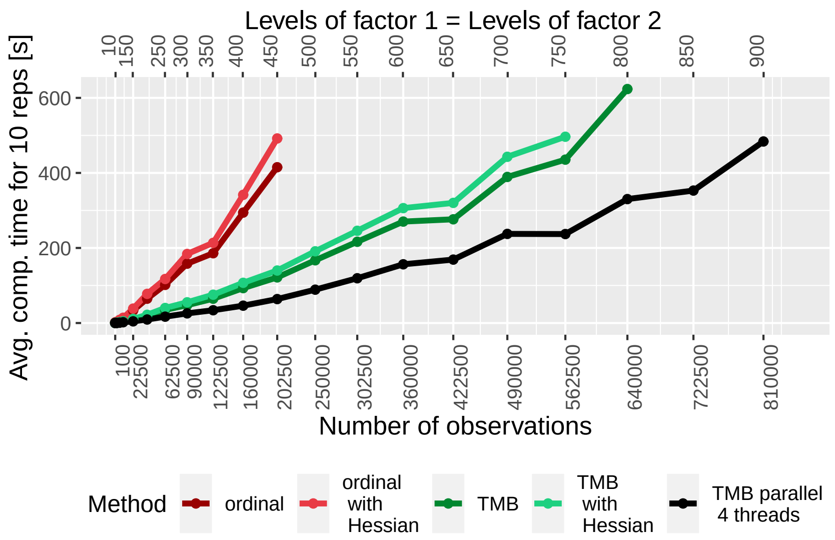

Figure 3: Average computation times of 10 repetitions when estimating model (8) for the simulated data. Computation times for the `ordinal` package are indicated with red. Computation times for `TMB` are indicated with green and black for parallel computations. ”with Hessian” indicates that the Hessian is calculated at optimum.

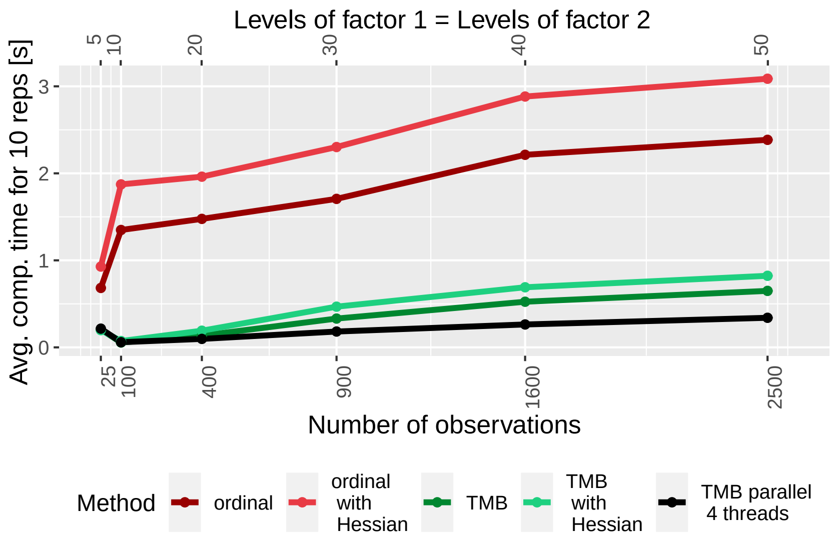

Figure 4: Zoom of figure 3. Average computation times of 10 repetitions when estimating model (8) for the simulated data. Computation times for the `ordinal` package are indicated with red. Computation times for `TMB` are indicated with green and black for parallel computations. ”with Hessian” indicates that the Hessian is calculated at optimum.

Figure 5 and 7 show the average computation times of 10 repetitions when estimating model (7) and model (8), respectively, for the subsets of the data set from the Netflix Prize Challenge (Netflix 2009). The figures show exactly the same trends as figures 1 and 3. Thus, the results for simulated data seem to carry over to real-life data. The greatest differences in computation times are still seen for the model where one factor (movie) is assumed fixed and one factor (user) is assumed random. Here, `TMB` is 51.62 times faster than `ordinal` (without Hessian), with 150 movies and users and a total of $22,350$ observations. For the model where both movie and user is assumed random, `TMB` is 3.59 times faster than `ordinal` (without Hessian), with 450 users and movies and a total of 191, 040 observations. Figures 6 and 8 show, similar to the above, that the differences in computation time for the smallest datasets are small.

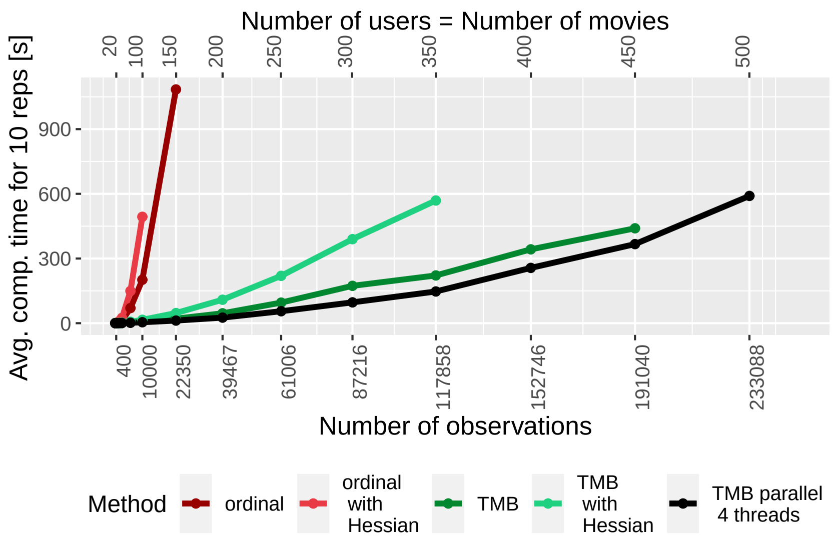

Figure 5: Average computation times of 10 repetitions when estimating model (7) for the Netflix Prize data set (movie fixed and user random). Computation times for the `ordinal` package are indicated with red. Computation times for `TMB` are indicated with green and black for parallel computations. "with Hessian" indicates that the Hessian is calculated at optimum.

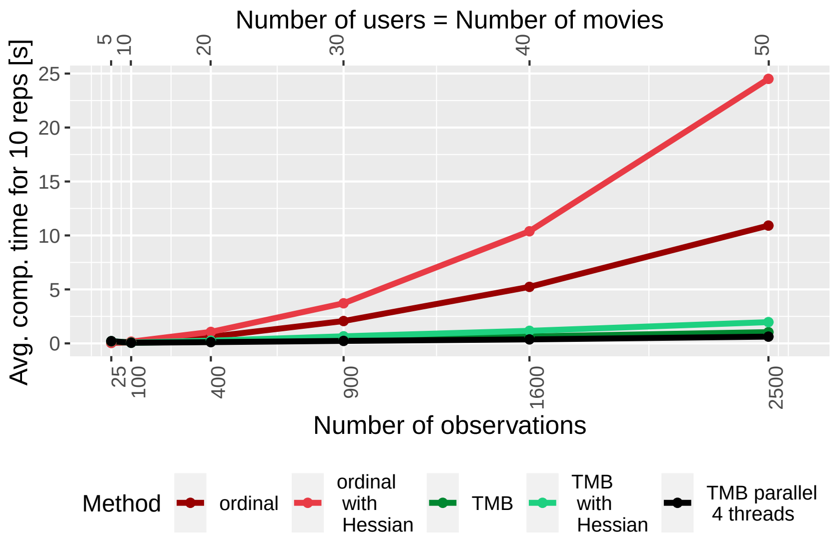

Figure 6: Zoom of figure 5. Average computation times of 10 repetitions when estimating model (7) for the Netflix Prize data set (movie fixed and user random). Computation times for the `ordinal` package are indicated with red. Computation times for `TMB` are indicated with green and black for parallel computations. "with Hessian" indicates that the Hessian is calculated at optimum.

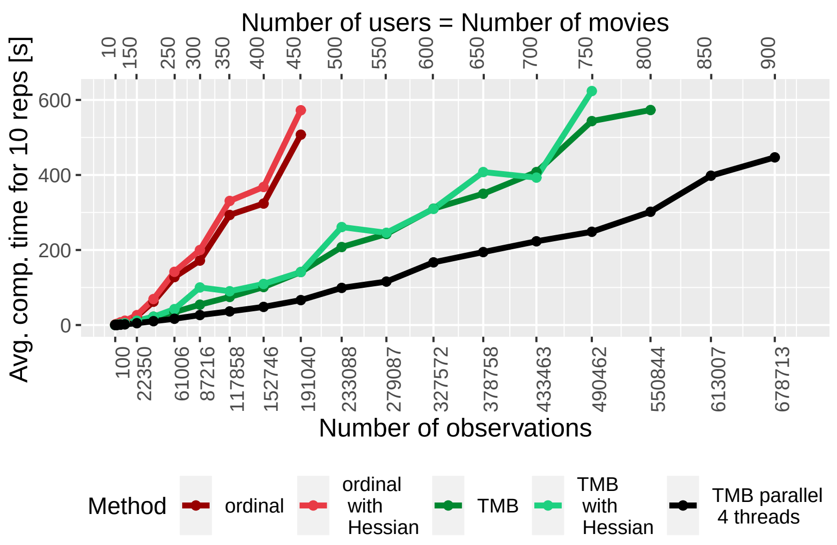

Figure 7: Average computation times of 10 repetitions when estimating model (8) for the Netflix Prize data set (movie and user random). Computation times for the `ordinal` package are indicated with red. Computation times for `TMB` are indicated with green and black for parallel computations. "with Hessian" indicates that the Hessian is calculated at optimum.

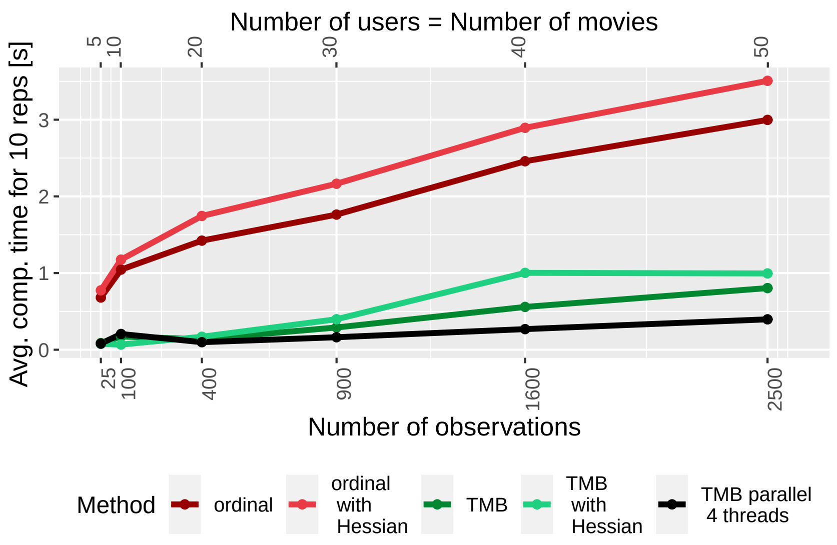

Figure 8: Zoom of figure 7. Average computation times of 10 repetitions when estimating model (8) for the Netflix Prize data set (movie and user random). Computation times for the `ordinal` package are indicated with red. Computation times for `TMB` are indicated with green and black for parallel computations. "with Hessian" indicates that the Hessian is calculated at optimum.

5. Discussion and Conclusion

The results illustrate that for the simple models considered here, estimation of ordinal mixed models are faster with the TMB package than with the ordinal package for large data sets with many parameters and random effects. Employing the Netflix Prize data set with 150 users, 150 movies and a total of $22,350$ observations, `TMB` is 51.62 times faster than `ordinal` estimating a model that assumes the user effect random and the movie effect fixed. This means that using the TMB implementation we are able to estimate ordinal mixed models with larger data sets and more parameters and random effects. There is no significant difference in computation time between the two implementations for smaller datasets with a low number of parameters and random effects, for all practical purposes.

However, even though we are able to estimate ordinal mixed models with larger data sets, there is still a need for methods that can estimate ordinal mixed models for data sets as large as the entire Netflix Prize data set (Netflix 2009). This is not possible with `TMB` (within reasonable time at least).

The `ordinal` package is more user-friendly than the `TMB` package. In contrast to the `ordinal` package, the `TMB` package is not exclusively oriented towards ordinal models, but is a general tool that enables efficient optimization of the Laplace approximation. It therefore requires the user to have a good theoretical understanding of ordinal mixed models and `C++` programming skills in order to use `TMB`. However, we believe that with a wrapper around the `TMB` framework, an implementation with `TMB` can be user-friendly too. This is left for future work.

References

Agresti, Alan (2012). *Analysis of Ordinal Categorical Data: Second Edition*. eng. John Wiley and Sons Inc, pp. 1–396. ISBN: 0470082895, 9780470082898, 0470594004, 1283542757, 0470593997, 9780470594001, 9781283542753, 9780470593998, 1118209990, 9786613855206. DOI: 10.1002/9780470594001.

Christensen, Rune Haubo Bojesen (2012). "Sensometrics: Thurstonian and Statistical Models". eng.

— (2018). "Cumulative Link Models for Ordinal Regression with the R Package ordinal". In:

— (2019). *ordinal—Regression Models for Ordinal Data*. R package version 2019.12-10. https://CRAN.R-project.org/package=ordinal.

Fournier, David A. et al. (2011). "AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models". eng. In: *Optimization Methods and Software* 27.2, pp. 233–249. ISSN: 10294937, 10556788, 10267670. DOI: 10.1080/10556788.2011.597854.

Griewank, Andreas (2000). *Evaluating derivatives : principles and techniques of algorithmic differentiation*. Philadelphia, 369 s. ISBN: 0898714516.

Jensen, Sofie Pødenphant (2019). "Biadditive Mixed Models - Advancing Computational Methods and Applications". eng.

Kristensen, Kasper et al. (2016). "TMB: Automatic differentiation and laplace approximation". eng. In: *Journal of Statistical Software* 70. ISSN: 15487660. DOI: 10.18637/jss.v070.i05.

Madsen, Henrik and Poul Thyregod (2011). *Introduction to general and generalized linear models*. eng. CRC Press, XIII, 302 S. (unknown). ISBN: 0429111703, 1439891141, 1420091557, 9781420091557.

Mccullagh, P. (1980). “Regression Models for Ordinal Data”. eng. In: *Journal of the Royal Statistical Society Series B-methodological* 42.2, pp. 109–142. ISSN: 00359246.

Netflix (2009). *Netflix Prize Data*. `http://www.netflixprize.com`. Accessed: 2020-07-02, downloaded from `https://www.kaggle.com/netflix-inc/netflix-prize-data`.

Skaug, Hans J. and David A. Fournier (2006). “Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models”. eng. In: *Computational Statistics and Data Analysis* 51.2, pp. 699–709. ISSN: 18727352, 01679473. DOI: `10.1016/j.csda.2006.03.005`.

Tutz, Gerhard (2011). *Regression for categorical data*. eng. Cambridge University Press, pp. 1–561. ISBN: 9781107009653, 9780511842061. DOI: `10.1017/CBO9780511842061`.

6. Appendix

Observations	**Levels factor 1**	**Levels factor 2**	**Threshold parameters**
25	5	5	4
100	10	10	4
400	20	20	4
900	30	30	4
1600	40	40	4
2500	50	50	4
5625	75	75	4
10^4	100	100	4
$2.25 \cdot 10^4$	150	150	4
$4 \cdot 10^4$	200	200	4
$6.25 \cdot 10^4$	250	250	4
$9 \cdot 10^4$	300	300	4
$1.225 \cdot 10^5$	350	350	4
$1.6 \cdot 10^5$	400	400	4
$2.025 \cdot 10^5$	450	450	4
$2.5 \cdot 10^5$	500	500	4
$3.025 \cdot 10^5$	550	550	4
$3.6 \cdot 10^5$	600	600	4
$4.225 \cdot 10^5$	650	650	4
$4.9 \cdot 10^5$	700	700	4
$5.625 \cdot 10^5$	750	750	4
$6.4 \cdot 10^5$	800	800	4
$7.225 \cdot 10^5$	850	850	4
$8.1 \cdot 10^5$	900	900	4

Table 1: Number of observations, number of levels for the two factors and the number of threshold parameters for the datasets simulated according to models (7) and (8).

Observations	**Users**	**Movies**	**Threshold parameters**
25	5	5	4
100	10	10	4
400	20	20	4
900	30	30	4
1600	40	40	4
2500	50	50	4
5625	75	75	4
$10,000$	100	100	4
$22,350$	150	150	4
$39,467$	200	200	4
$61,006$	250	250	4
$87,216$	300	300	4
$117,858$	350	350	4
$152,746$	400	400	4
$191,040$	450	450	4
$233,088$	500	500	4
$279,087$	550	550	4
$327,572$	600	600	4
$378,758$	650	650	4
$433,463$	700	700	4
$490,462$	750	750	4
$550,844$	800	800	4
$613,007$	850	850	4
$678,713$	900	900	4

Table 2: Number of observations, number of movies and users and the number of threshold parameters for the subsets of the Netflix prize dataset.