Master's Degree in Data Science

Scalable Inference for Crossed Random Effects Models

Author: Maximilian Müller

Directors: Omiros Papaspiliopoulos and Giacomo Zanella

17.08.2020

ABSTRACT IN ENGLISH:

In order to apply statistical learning in the framework of crossed random effects models it is necessary to efficiently compute the Cholesky factor $\boldsymbol{L}$ of the models precision matrix. In this paper we show that for the case of 2 factors the crucial point to this end is not only the sparsity of $\boldsymbol{L}$, but also the arrangement of non-zero entries. In particular, we express the number of flops required for the calculation of $\boldsymbol{L}$ by the number of 3-cycles in the corresponding graph. We then introduce specific designs of 2-factor crossed random effects models for which we can prove sparsity and density, respectively. We confirm our results by numerical studies with the R-packages *Spam* and *Matrix* and find hints that approximations of the Cholesky factor could be an interesting approach for further decrease of the cost of computing $\boldsymbol{L}$.

ABSTRACT IN CATALAN/ SPANISH:

Para aplicar el aprendizaje estadístico en el marco de los modelos de efectos aleatorios cruzados es necesario calcular eficientemente el factor $\boldsymbol{L}$ de Cholesky de la matriz de precisión de los modelos. En este trabajo mostramos que para el caso de dos factores el punto crucial para este fin no es sólo la dispersión de $\boldsymbol{L}$, sino también la disposición de las entradas no nulas. En particular, expresamos el número de FLOPS necesarios para el cálculo de $\boldsymbol{L}$ por el número de 3-ciclos en el gráfico correspondiente. A continuación, introducimos diseños específicos de modelos de efectos aleatorios cruzados de 2 factores para los que podemos probar la dispersión y la densidad, respectivamente. Confirmamos nuestros resultados mediante estudios numéricos con los paquetes Spam y Matrix de R y encontramos indicios de que las aproximaciones del factor Cholesky podrían ser un enfoque interesante para una mayor disminución del costo del cálculo de $\boldsymbol{L}$.

KEYWORDS IN ENGLISH:

Crossed random effects models, cholesky factorization, sparse linear algebra, bayesian statistics

KEYWORDS IN CATALAN/ SPANISH:

Modelos de efectos aleatorios cruzados, factorización cholesky, álgebra lineal dispersa, estadísticas bayesianas

MASTER THESIS

Scalable Inference for Crossed Random Effects Models

Submitted in partial fulfillment of the requirements for the
Master's Degree in Data Science
at the BARCELONA GSE

under the supervision of
OMIROS PAPASPILIOPOULOS (UPF)
GIACOMO ZANELLA (BOCCONI)

Maximilian Müller

Barcelona, 22^{th} of June 2020

Contents

1 Introduction

In many modern statistical setups crossed random effects models play an increasingly important role. They are additive models that relate a response variable y to categorical predictors and are commonly used to describe the different sources of variation in a dataset (Papaspiliopoulos et al. [2019]). In general, many categorical predictors can be modelled. However, the example we will use throughout this thesis is the arrangement with 2 factors, where e.g. customers are linked to products through ratings, orderings or clicks on webpages. This framework is especially interesting since it can describe typical recommender system problems. The 2-factor crossed random effects model and the response variable y_{ij} can be written as

$$y_{ij} = \mu + a_i + b_j + \epsilon_{ij} \qquad (i,j) \in S \subseteq [I] \times [J] \tag{1}$$

where a_i is the i^{th} level of the categorical factor a, b_j is the j^{th} level of the categorical factor b and μ has the interpretation of a global mean. For this paper we will refer to the categorical factors a and b as customers and products, respectively. The model parameters are assumed to follow normal distributions:

$$\begin{aligned} a_i &\overset{iid}{\sim} N(0, \tau_a^{-1}) \\ b_j &\overset{iid}{\sim} N(0, \tau_b^{-1}) \\ \epsilon_{ij} &\overset{iid}{\sim} N(0, \tau_e^{-1}) \\ p(\mu) &\propto 1 \end{aligned}$$

In the more general case the precision parameters τ might underlie a distribution themselves which can vary with i, j. For the scope of this paper we will take them to be independent of i, j and to be constant and known. For convenience, we can store the global parameters in a parameter vector

$$\boldsymbol{\beta} = (\mu, a_1, ..., a_I, b_1, ..., b_J)$$

The most prominent example of the recommender system setup is the so-called Netflix problem which has challenged thousands of participants to predict movie grades of Netflix customers. In the dataset there were about 100.000.000 ratings of 17.700 movies by 480.000 customers. These numbers are typical for many of todays problems, often the data contains even millions of customers and equally millions of products, making it very high-dimensional. Applying statistical learning in this context and trying to learn a predictive function that maps the data to the response variable can therefore be very challenging. As we will see in this thesis, the computational part of Bayesian inference can be boiled down to sampling from posterior distibutions, which in return can be turned into convenient linear algebra problems if the posterior is of Gaussian form. For crossed random effects models this is the case, as it will be shown in Section 5.1. The required algorithms, however, scale cubically with the dimension of the data. In the case of the high-dimensional setting in recommender systems this is therefore unpractical. In particular, the computational bottleneck is the computation of the Cholesky factor of the precision matrix. Since the precision matrix of the posterior is also sparse for crossed random effects models, one can, however, take advantage of sparse linear algebra methods. As we will show, this can reduce the computational complexity greatly and make inference based on linear algebra methods scalable to high dimensions.

1.1 Main Contributions

One of the main contributions of this paper is theorem 3.2, where it is derived that the crucial point for efficient computation of the Cholesky factor is not only the sparsity of the Cholesky factor, but also the arrangements of the non-zero entries. In particular, if the exact structure of $\boldsymbol{L}$ is known, the number of flops required for the calculation of $\boldsymbol{L}$ can be expressed in terms of the number of 3-cycles in the graph of $\boldsymbol{L}$. This leads to a lower bound of $\mathcal{O}(\|\boldsymbol{L}\|_0)$ and an upper bound of $\mathcal{O}(\|\boldsymbol{L}\|_0^{1.5})$, where $\|\boldsymbol{L}\|_0$ are the number of non-zero entries in $\boldsymbol{L}$. Even though there is a wide range of literature dealing with the computational cost of Cholesky decomposition, up to our knowledge it has not been expressed in terms of the number of 3-cycles. We furthermore extend this result to the more general case where the exact sparsity structure is not exactly known but has to be predicted taking advantage of the concepts of future set (theorem 2.8 in Rue and Held [2004]) and the fill (Gilbert [1994]).

A big part of this paper is then devoted to a thorough exploration of 2-factor crossed random effects models and their interaction with sparsity. Since this is hard for the general case, we introduce specific designs which allow simpler derivations. We show that the position of μ in the so-called $ab\mu$ ordering is optimal for reduced fill-in ratio. We then introduce the Markovian Design and the Problematic Design, for which we can prove that the resulting Cholesky factor will be sparse in one case and dense in the other.

Furthermore, we investigate the performance of the R packages $\boldsymbol{Spam}$ and $\boldsymbol{Matrix}$ on these cases and confirm our analytical results numerically. We also investigate random Erdös-Renyi graphs and find no evidence that these can be ordered such that the Cholesky factor is sparse. In all investigated cases we find hints that approximations of the Cholesky factor could be an interesting approach for further decrease of the cost of computing $\boldsymbol{L}$.

1.2 Outline of This Paper

The outline of this paper is as follows: In Section 2 it is shown how Bayesian learning can be turned into a linear algebra problem and in particular into the computation of the Cholesky factor $\boldsymbol{L}$ of the precision matrix. Then, in Section 3 the calculation of the Cholesky factor is examined more thoroughly and bounds for the computational cost are derived. Graphical models are introduced as a handy tool to investigate the conditions for sparsity of $\boldsymbol{L}$ in Section 4. In particular, the concepts of future set and fill graph are presented and related to sparsity and specifically the cost of computing $\boldsymbol{L}$. Section 5 introduces the special designs and orderings of the 2-factor crossed random effects model, whose sparsity is investigated analytically in Section 6. Finally, the results of the numerical studies are presented in Section 7. In Section 8 the findings of this paper are summarized and and outlook over possible future work is given in Section 9.

1.3 Conventions and Definitions

For this paper, we will adapt the following conventions and definitions: Vectors $\boldsymbol{x}$ will be denoted in bold lowercase letters and matrices $\boldsymbol{A}$ in bold uppercase letters. Scalars can be upper and lowercase, but not bold. A range is indicated with a semicolon, e.g. 3:6 indicates the set $\{3, 4, 5, 6\}$. This is also used for vector and matrix subsetting. When subsetting a vector or a matrix, a single colon : represents all possible elements of the corresponding dimension. $\boldsymbol{Q}_{3:6,:}$ denotes the submatrix of $\boldsymbol{Q}$ that is obtained if one selects all columns and rows 3 to 6. For single elements of a matrix the comma is usually neglected for convenience. Q_{ij} hence denotes the element in row i and column j of matrix $\boldsymbol{Q}$. A big part of this thesis is devoted to deriving sparsity conditions for Choleksy factors. We therefore want to give formal definitions to the most relevant concepts. For a positive definite square matrix $\boldsymbol{Q}$ the decomposition into the product of a lower triangular matrix $\boldsymbol{L}$ and its conjugate transpose is called Cholesky factorization:

$$\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$$

We refer to $\boldsymbol{L}$ as the Cholesky factor of $\boldsymbol{Q}$. A square matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ is sparse if the number of non-zero entries in $\boldsymbol{B}$ is of $\mathcal{O}(n)$. Sparsity is assessed in terms of the non-zero elements of a matrix. These can be expressed conveniently as the L0-norm:

$$\|\boldsymbol{Q}\|_0 = |\{Q_{ij}|Q_{ij} \neq 0\}|$$

is the number of non-zeros in $\boldsymbol{Q}$. And finally, for measuring the size of matrices or differences between two matrices it is helpful to introduce the Frobenius norm. For a Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ with entries a_{ij} the Frobenius norm is defined as

$$||\boldsymbol{A}||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}|^2}$$

2 Bayesian Learning as a Linear Algebra Problem

In Section 5.1 and in more detail in appendix A it will be shown that the posterior of a 2-factor crossed random effects model is a sparse Gaussian. In this section we will derive how for this case Bayesian learning becomes first a sampling problem which can then be formulated as a linear algebra problem.

2.1 Bayesian Learning as a Sampling Problem

In Bayesian learning, one deals with data $\boldsymbol{y}$ and parameters $\boldsymbol{\theta}$. For the parameters, a prior $p(\boldsymbol{\theta})$ is assumed and for the data, a model $p(\boldsymbol{y}|\boldsymbol{\theta})$ is defined. One is then usually interested in the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$ which can be obtained by Bayes rule:

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

In order to obtain the exact posterior distribution, a normalization is necessary, which involves an integral over $p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$, i.e. the non-normalized posterior. Also, marginalization or computing expectations with respect to the posterior require the computation of integrals with respect to the posterior. This need for integration is usually the main computational bottleneck. For crossed random effects the integration has to be performed over the space of factors and can easily become infeasible as the space is typically very high-dimensional. Therefore, alternatives like MCMC methods are commonly used. The principle of MCMC methods is to draw samples from a distribution that is not necessarily normalized such that the relative importance of the parameter space is maintained. These random samples are then used to approximate the integral in a way that it converges to its true expression.

2.2 Sampling from a Gaussian as a Linear Algebra Problem

It is now clear that Bayesian inference can be addressed by sampling from the posterior distribution. In the case of 2-factor crossed random effects models the posterior is a Gaussian. In order to obtain a sample $\boldsymbol{V}$ from a m-dimensional normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ one usually first samples from a m-dimensional standard normal, $\boldsymbol{z} \sim N(\boldsymbol{0}, \boldsymbol{I}_m)$. $\boldsymbol{V}$ can then be calculated via

$$\boldsymbol{V} = \boldsymbol{\mu} + \tilde{\boldsymbol{L}}\boldsymbol{z}$$

where $\tilde{\boldsymbol{L}}$ is the Cholesky factor of $\boldsymbol{\Sigma} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T$. In many cases, also for 2-factor crossed random effects models, the precision matrix $\boldsymbol{Q}$ is easier to obtain. $\boldsymbol{V}$ is then calculated by first solving

$$\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{z}$$

for $\boldsymbol{x}$, where $\boldsymbol{L}$ is the Cholesky factor of $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$ and again $\boldsymbol{z} \sim N(\boldsymbol{0}, \boldsymbol{I}_m)$. $\boldsymbol{V}$ is then obtained according to

$$\boldsymbol{V} = \boldsymbol{\mu} + \boldsymbol{x}$$

In both cases the computational bottleneck is the calculation of the Cholesky factor as it requires $\mathcal{O}(n^3)$ flops for dense matrices. However, if $\boldsymbol{Q}$ is sparse, sparse linear algebra methods can be applied. As we will see in the following section, a sparse $\boldsymbol{Q}$ is not enough for an efficient calculation of $\boldsymbol{L}$. We will therefore investigate how $\boldsymbol{L}$ is computed and under what circumstances we can improve its cost.

3 Computation of the Cholesky Factor of a Sparse Matrix

In this section we will investigate how the Cholesky factor is calculated and derive bounds for the computational cost if $\boldsymbol{L}$ is sparse. A well-known algorithm for the calculation of $\boldsymbol{L}$ is the Gaxpy-Rich algorithm (4.2.5 in Golub and van Loan [2013]) which is based on the following observations: For a matrix $\boldsymbol{Q}$ with Cholesky factor $\boldsymbol{L}$, i.e. $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$, one can write

$$Q_{ij} = \sum_{k=1}^{j} L_{jk}L_{ik} \quad \forall i \geq j$$

which follows directly from the definition of matrix multiplication. One can rearrange terms and define v_{ij} as

$$L_{jj}L_{ij} = Q_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk} \equiv v_{ij} \quad \forall i \geq j$$

The elements of the Cholesky factor can then be calculated from v_{ij} via

$$\begin{aligned} L_{jj} &= \sqrt{v_{jj}} \\ L_{ij} &= \frac{v_{ij}}{L_{jj}} = \frac{v_{ij}}{\sqrt{v_{jj}}} \end{aligned} \tag{2}$$

One can hence compute v_{ij} and L_{ij} by iterating through the columns of $\boldsymbol{L}$ and using equations (2). This leads to the Gaxpy-Rich algorithm:

Algorithm 1: Gaxpy-Rich

Result: Calculates the Cholesky factor $\boldsymbol{L}$ of a matrix $\boldsymbol{Q}$
for *i=1:n* **do**
  set $L_{i1} = \frac{Q_{i1}}{\sqrt{Q_{11}}}$
end
for *j=2:n* **do**
  for *i=j:n* **do**
    compute $v_{ij} = Q_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk}$;
    set $L_{ij} = \frac{v_{ij}}{\sqrt{v_{jj}}}$;
  end
end

For dense matrices the algorithm needs $\mathcal{O}(n^3)$ flops to calculate $\boldsymbol{L}$ because the two for loops are of $O(n)$ and the calculation of $\sum_{k=1}^{j-1} L_{ik}L_{jk}$ is also of $\mathcal{O}(n)$, as it is an element-wise multiplication of vectors, i.e. an inner product. This last observation is important: In order to calculate an entry L_{ij} the inner product between rows i and j in $\boldsymbol{L}$ has to be calculated. This is illustrated in Figure 1, where for the calculation of L_{75} (green) the inner product between the blue rows has to be taken.

In order to see how sparsity can help to improve the cost of computing $\boldsymbol{L}$, it is instructive to note from equation (2) that

$$L_{ij} \propto Q_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk} \tag{3}$$

This can be used to make strong statements about the sparsity structure of $\boldsymbol{L}$ that are at the heart of efficient computations of the Cholesky factor.

Lemma 3.1. *If $L_{ij} = 0$ one of the following three conditions has to be met*

- $Q_{ij} = 0$ *and* $L_{ik}L_{jk} = 0 \quad \forall \quad k \in \{1, ..., j-1\}$
- $Q_{ij} \neq 0$ *and* $L_{ik}L_{jk} \neq 0$ *for some* $k \in \{1, ..., j-1\}$*, but* $Q_{ij} = \sum_{k=1}^{j-1} L_{ik}L_{jk}$
- $Q_{ij} = 0$ *and* $L_{ik}L_{jk} \neq 0$ *for some* $k \in \{1, ..., j-1\}$*, but* $\sum_{k=1}^{j-1} L_{ik}L_{jk} = 0$

Proof. The Lemma follows directly from equation (3). □

The later two conditions are referred to as numerical cancellation.

Definition 3.1. We define numerical cancellation as the case when $L_{ik}L_{jk} \neq 0$ for some $k \in \{1, ..., j-1\}$, but $L_{ij} = 0$.

Furthermore, one notes that if we knew the exact sparsity structure of $\boldsymbol{L}$, i.e. the positions of the non-zero elements beforehand, we would only have to calculate these non-zero elements. This can lead to a great reduction in computational cost if $\boldsymbol{L}$ is sparse.

Theorem 3.2. *Let $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$ be a matrix with Cholesky factor $\boldsymbol{L}$. Let G_L be an undirected graph that has edges between vertices $i > j$ if and only if $L_{ij} \neq 0$. If the positions of the non-zero entries of $\boldsymbol{L}$ are known, the Cholesky factor can be computed with*

$$f = \mathcal{O}\left(\|\boldsymbol{L}\|_0 + \eta_3(\boldsymbol{L})\right)$$

flops, where $\eta_3(\boldsymbol{L})$ is the number of three-cycles in G_L.

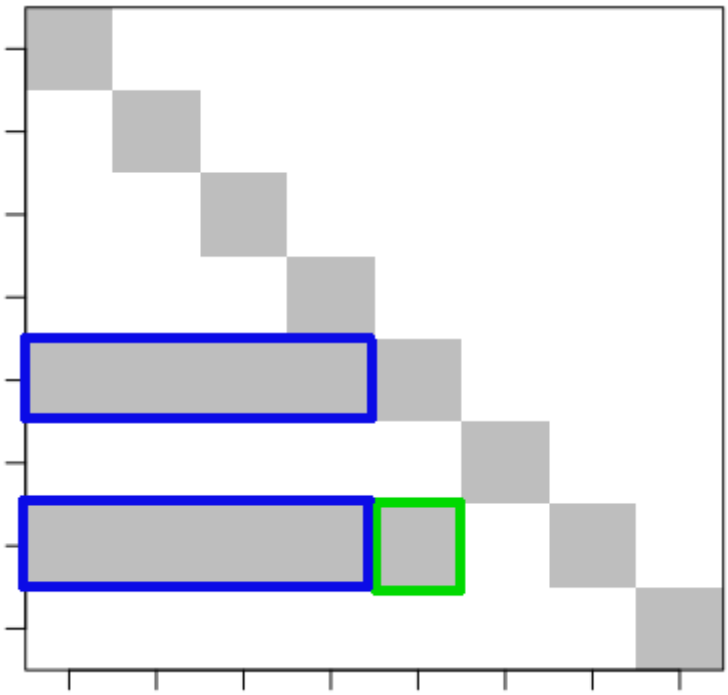


Figure 1: Illustration of $\sum_{k=1}^{j-1} L_{ik}L_{jk}$ for $i = 7, j = 5$. The sum that is required for the calculation of L_{75} (green cell) corresponds to an inner product between row 5 and row 7, which are marked blue.

Proof. For every non-zero entry of $\boldsymbol{L}$, v_{ij} has to be initialized. This thus requires $\|\boldsymbol{L}\|_0$ flops. Furthermore, after v_{ij} was computed, every L_{ij} is obtained by $v_{ij}/\sqrt{v_{jj}}$, which requires another $\|\boldsymbol{L}\|_0$ flops. For the calculation of v_{ij} we will distinguish the two cases where $i = j$ and where $i \neq j$.

If $i = j$, we are calculating diagonal entries of $\boldsymbol{L}$ and the product $L_{ik}L_{jk}$ becomes L_{ik}^2. For every diagonal element L_{ii} we thus have to square all elements of row i and subtract them from Q_{ii}. This is done for every diagonal element which leads to square-subtracting each off-diagonal non-zero element once, requiring at most $\mathcal{O}(\|\boldsymbol{L}\|_0)$ flops.

For $i \neq j$ we can assume $i > j$ without loss of generality, because we know that the Cholesky factor $\boldsymbol{L}$ is lower triangular. Since we know the exact structure of $\boldsymbol{L}$, we do not have to compute all products of $L_{jk} \times L_{ik}$, but only those for which L_{jk}, L_{ik} are both non-zero. For every non-zero element L_{ij} the computational cost resulting from $\sum_{k=1}^{j-1} L_{ik}L_{jk}$ is therefore of the order of the number of times L_{ik}, L_{jk} are pairwise non-zero. $L_{ik} \neq 0, L_{jk} \neq 0, L_{ij} \neq 0$ for $k < j < i$ however just correspond to three edges in G_L that connect nodes i, j, k and thus create a triangle. This means that the fact that in order to calculate L_{ij} it is necessary to compute the product between L_{jk}, L_{ik} because they are both non-zero is uniquely represented by a triangle in G_L. The overall cost resulting from the sums $\sum_{k=1}^{j-1} L_{ik}L_{jk}$ is hence of the order of the number of triangles in G_L which we denote with $\eta_3(\boldsymbol{L})$:

$$\eta_3(\boldsymbol{L}) = |\{(i, j, k) | L_{ik} \neq 0, L_{jk} \neq 0, L_{ij} \neq 0, k < j < i\}| \tag{4}$$

□

Corollary 3.2.1. *More specifically, the cost from theorem 3.2 can be bounded by*

$$\mathcal{O}\left(\|\boldsymbol{L}\|_0\right) \leq f \leq \mathcal{O}\left(\|\boldsymbol{L}\|_0 + \|\boldsymbol{L}\|_0^{1.5}\right)$$

Proof. The lower bound is immediate: If there is no three-cycle in G_L, $\eta_3(\boldsymbol{L}) = 0$ and the cost is $\mathcal{O}\left(\|\boldsymbol{L}\|_0\right)$. For the upper bound it is necessary to find the maximal number of 3-cycles in a graph with $\|\boldsymbol{L}\|_0$ edges. In order to address this, one has to note that the most number of 3-cycles appear in a maximally connected

graph. As we have $\|\boldsymbol{L}\|_0$ edges and a fully connected graph with n nodes has $n(n-1)$ edges, the number of nodes for a maximally connected graph with $\|\boldsymbol{L}\|_0$ edges will be of $\mathcal{O}(\sqrt{\|\boldsymbol{L}\|_0})$. For a fully connected graph with n nodes it is trivial to find the number of 3-cycles since it is equivalent to the number of ways one can choose 3 nodes from n, i.e. $\binom{n}{3}$. Hence,

$$\eta_3(\boldsymbol{L}) \leq \mathcal{O}\left(\binom{\sqrt{\|\boldsymbol{L}\|_0}}{3}\right)$$

Using $\binom{n}{k} \leq (\frac{en}{k})^k < n^k$ for $k > e$, this can be written as

$$\eta_3(\boldsymbol{L}) \leq \mathcal{O}\left(\|\boldsymbol{L}\|_0^{1.5}\right)$$

This result is confirmed in Rivin [2002] (theorem 4), where the author shows bounds for general k-cycles. □

An immediate implication of theorem 3.2 is that neither the number of non-zero entries in $\boldsymbol{Q}$ nor in $\boldsymbol{L}$ completely determine the computational cost of calculating the Cholesky factor. Instead, it is relevant how the non-zero entries in L are ordered with respect to each other, i.e. how many 3-cycles they form. The intuition behind the implications of theorem 3.2 can be understood from Figure 2. In the left case, the graph of $\boldsymbol{L}$ contains no cycles. This is equivalent to the fact that for no $k < j < i$ there are entries L_{ik}, L_{jk} that are both non-zero, which can also be seen immediately from $\boldsymbol{L}$ in matrix form. Hence, for no off-diagonal entry L_{ij} a product $L_{ik}L_{jk}$ has to be computed.
In the right case, however, there are 3 combinations of $k < j < i$ such that L_{ik}, L_{jk} that are both non-zero. Each of these combinations requires 2 additional flops (computing the product and substracting it). In $\boldsymbol{L}$, the colored cubes represent the pairwise non-zero L_{ik}, L_{jk} and the colored dot the entry L_{ij} which is affected by that. The equivalent 3-cycles are drawn in the same color in the graph of the Cholesky factor. For L_{56} (blue), 2 additional flops are required as L_{53} and L_{63} are both non-zero, leading to the blue 3-cycle in the graph. For L_{53}, 4 additional flops are required since $L_{51}, L_{31} \neq 0$ as well as $L_{52}, L_{32} \neq 0$, resulting in the two green 3-edged cycles.

It is reassuring to note that the bounds from theorem 3.2 are in accordance with the well-known bounds for dense matrices: In a dense matrix $\|\boldsymbol{L}\|_0 = \mathcal{O}(n^2)$ and thus the complexity is $\geq \mathcal{O}(\left[n^2\right]^{1.5}) = \mathcal{O}(n^3)$. This gives a hint when the computational cost of calculating $\boldsymbol{L}$ can be of $\mathcal{O}(\|\boldsymbol{L}_0\|^{1.5})$.

Proposition 1. Suppose the Cholesky factor $\boldsymbol{L} \in \mathcal{R}^{n\times n}$ of a matrix $\boldsymbol{Q}$ is sparse, i.e. has $\|\boldsymbol{L}\|_0 = \mathcal{O}(n)$ non-zero entries and the following structure: The first $\sqrt{n}$ rows of the lower triangular structure of $\boldsymbol{L}$ and the complete diagonal are non-zero, but all other entries are zero. Then the computational cost f of calculating the Cholesky factor is

$$f = \mathcal{O}(\|\boldsymbol{L}\|_0^{1.5}) \tag{5}$$

.

Proof. First it is to note that $\boldsymbol{L}$ indeed is a sparse matrix: The number of non-zeros is of order $\mathcal{O}(n+\sqrt{n}^2) = \mathcal{O}(n)$, because there are order $\sqrt{n} \times \sqrt{n}$ non-zero entries on the off-diagonal. The off-diagonal non-zero entries of $\boldsymbol{L}$ however just form a dense block within $\boldsymbol{L}$. It is well known that the complexity of calculating the cholesky of a dense matrix is cubic in its dimension. This implies that the cost of calculating the off-diagonal entries is of

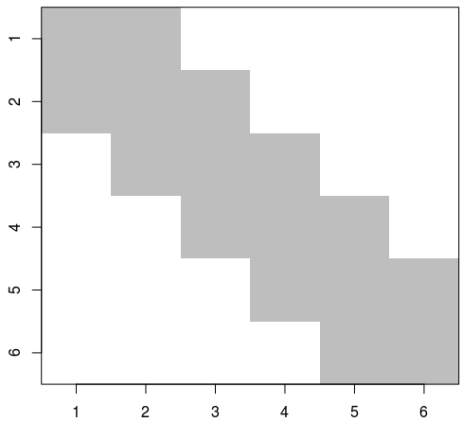
$$\mathcal{O}(\sqrt{n}^3) = \mathcal{O}(n^{1.5}) = \mathcal{O}(\|\boldsymbol{L}\|_0^{1.5})$$

which is just the upper bound from corollary 3.2.1. □

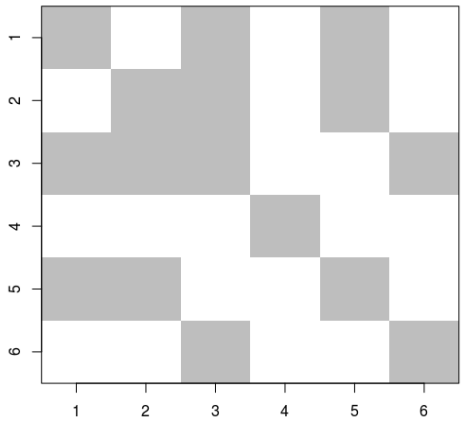
Proposition 1 indicates that the upper bound from corollary 3.2.1 might not be very tight, unless the sparsity structure of $\boldsymbol{L}$ is such that it partly resembles that of a dense matrix.

For the results of this section we assumed that the sparsity structure of $\boldsymbol{L}$ is exactly known. This is usually not the case. The reason for that is that numerical cancellation typically cannot be predicted from the structure of $\boldsymbol{Q}$. It is only found after the corresponding element of $\boldsymbol{L}$ is calculated exactly. Since a full calculation of the element L_{ij} is necessary in order to check whether numerical cancellation occurs, these

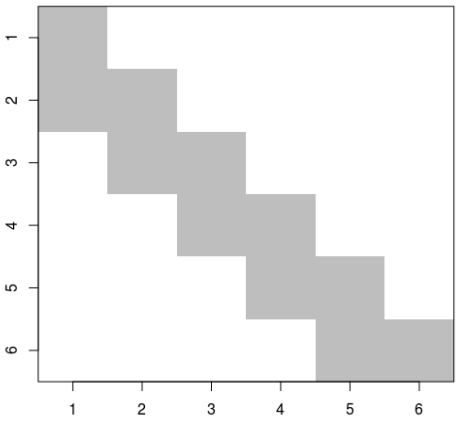
zero-values in $\boldsymbol{L}$ have to be treated like non-zero elements if one is interested in predicting the sparsity structure of $\boldsymbol{L}$. Methods, some of which we introduce in the next section, therefore usually anticipate the sparsity structure of $\boldsymbol{L}$ such that one can distinguish between elements that are zero and elements that *can* be non-zero and thus have to be calculated. The elements that *can* be non-zero might still turn out to be zero, due to numerical cancellation, but this does not help for the reduction of computational cost.



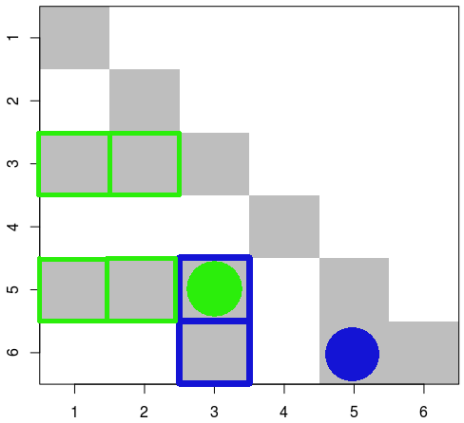
(a) Precision matrix 1.



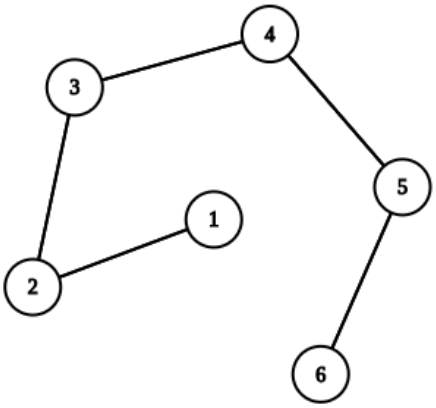
(b) Precision matrix 2.



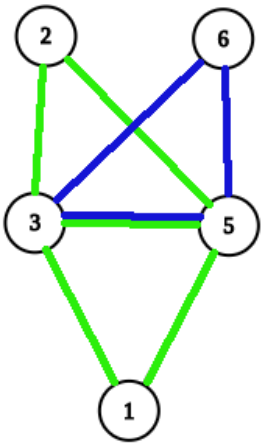
(c) The Cholesky factor of precision matrix 1, $\|\boldsymbol{L}\|_0 = 11$.



(d) The Cholesky factor of precision matrix 2, $\|\boldsymbol{L}\|_0 = 13$. Squares of the same color are needed to calculate the dots of the same color.



(e) The graph of the Cholesky factor 1. It does not contain any cycles.



(f) The graph of the Cholesky factor 2. It contains 3 cycles.

Figure 2: Equivalence of number of 3-cycles in G_L and the number of flops required for the calculation of $\boldsymbol{L}$. In the left case there are no 3-cycles in the graph of the Cholesky factor and hence no product L_{ik}, L_{jk} has to be calculated. In the right case, there are 3 3-cycles. The one marked blue represents the fact that $L_{53}, L_{63} \neq 0$ which leads to 2 additional flops in the calculation of L_{65} The ones marked in green represent the fact that $L_{31}, L_{51} \neq 0$ which leads to 4 additional flops in the calculation of L_{53}.

4 Gaussian Graphical Models and Sparse Cholesky Factors

It is well known that the sparsity of a precision matrix $\boldsymbol{Q}$ in Gaussian models follows directly from the conditional independence structure. It is therefore useful to consider the conditional independence graph G of the parameters. The Graph consists of the set of labelled nodes or vertices $V = \{1, 2, ..., n\}$ and the set of edges E. The parameters of the model, which we denote with $\boldsymbol{x} = (x_1, ..., x_n)$, are represented by the vertices in V. E connects the vertices such that there is no edge between node i and j if and only if

$$x_i \quad \perp \quad x_j \quad | \quad \boldsymbol{x}_{-ij}$$

which is known as Markov property. It is well known that conditional independence manifests itself with a zero-entry in the precision matrix (theorem 2.2 in Rue and Held [2004]) and hence

$$x_i \perp x_j \mid \boldsymbol{x}_{-ij} \quad \iff \quad Q_{ij} = 0 \quad \iff \quad \textit{no edge between nodes i and j in G} \tag{6}$$

This makes it easy to deduce the graph structure from a precision matrix and vice versa. Now let $\boldsymbol{L}$ be the Cholesky factor of $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$. In order to make statements about the sparsity of $\boldsymbol{L}$ it is useful to define for $1 \leq i < j \leq n$ the future of i except j as the set of indices

$$F(i, j) = \{i + 1, .., j - 1, j + 1, ...n\}$$

It can be shown (2.8 in Rue and Held [2004]) that for $\boldsymbol{L}$ a statement regarding conditional independence similar to the one for $\boldsymbol{Q}$ can be made:

Theorem 4.1. *Let $\boldsymbol{x}$ be a random vector with mean μ, precision matrix $\boldsymbol{Q}$ and conditional independence graph G like in* (6) *(this is, $\boldsymbol{x}$ is a GMRF). Let $\boldsymbol{L}$ be the Cholesky factor of $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$. If and only if x_i and x_j are independent given $\boldsymbol{x}_{F(i,j)}$ then $L_{ij} = 0$.*

$$x_i \perp x_j \mid \boldsymbol{x}_{F(i,j)} \quad \iff \quad L_{ji} = 0$$

As an important corollary they derive a simple and sufficient condition for $L_{ji} = 0$:

Corollary 4.1.1.

$$\textit{F(i,j) blocks all paths from i to j in G} \quad \implies \quad L_{ji} = 0$$

This means that for a given graph we can make use of the future set and infer the structure of $\boldsymbol{L}$. We can immediately tell which entries L_{ij} are zero and which *can* be non-zero and thus have to be calculated according to equation (2). As explained, these entries can still be zero, but this cannot be anticipated by the future set. It is therefore helpful to define the fill of $\boldsymbol{Q}$ as the set of pairs of vertices corresponding to possible non-zero entries in $\boldsymbol{L}$:

Definition 4.1. The fill of $\boldsymbol{Q}$ is defined as

$$fill(\boldsymbol{Q}) = \{(i, j) : \exists \text{ path from i to j in } G \text{ whose vertices are all less than min}(i, j)\}$$

and the fill-graph of $\boldsymbol{Q}$ is the undirected graph implied by $fill(\boldsymbol{Q})$.

Here it is to note that the fill contains pairs of vertices which can be understood as directed edges in a graph. Since $(i, j) \in fill(\boldsymbol{Q}) \iff (j, i) \in fill(\boldsymbol{Q})$, the resulting fill-graph is however indeed undirected. If we refer to the number of edges or cycles in the fill-graph, we therefore implicitly mean the number of undirected edges/cycles. In particular, (i, j) and (j, i) for $i \neq j$ are counted as a single, undirected edge.

Definition 4.2. $n_{\boldsymbol{L}}$ denotes the number of undirected edges in the fill-graph of $\boldsymbol{Q}$.

It is important to note that the fill contains all pairs of vertices for which there is a path that cannot be blocked by the future set. It in particular also contains all direct paths, i.e. the ones corresponding to non-zero entries in $\boldsymbol{Q}$. Using corollary 4.1.1 one can therefore understand the edges in the fill-graph as the possible non-zero entries in $\boldsymbol{L}$ that have to be computed. $n_{\boldsymbol{L}}$ hence counts precisely those entries that are relevant for the computational cost of obtaining $\boldsymbol{L}$. Theorem 3.2 and corollary 3.2.1 are easily extended

to this concept: If instead of the exact non-zero structure of $\boldsymbol{L}$ only the fill is known, one has to take the number of 3-cycles in the fill-graph instead of the number of 3-cycles in in the Graph implied by $\boldsymbol{L}$ for a bound on computational cost. In the same spirit, the exact number of non-zeros $\|\boldsymbol{L}\|_0$ can be replaced with $n_{\boldsymbol{L}}$. $n_{\boldsymbol{L}}$ is therefore a crucial quantity to monitor when assessing the computational cost of Cholesky factorization.

Corollary 4.1.2. *For a matrix $\boldsymbol{Q} \in \mathcal{R}^{n\times n}$ with non-zero diagonal elements and Cholesky factor $\boldsymbol{L}$ the number of possible non-zeros is*

$$n_{\boldsymbol{L}} = \frac{1}{2}(|fill(\boldsymbol{Q})| + n)$$

Proof. $fill(\boldsymbol{Q})$ contains all pairs of vertices i, j for which there exists a path that cannot be blocked by the corresponding future set. For $i > j$ both (i, j) and (j, i) are in $fill(\boldsymbol{Q})$, but only L_{ij} can be non-zero. For each of the n diagonal elements L_{ii} there is only one edge (i, i) in $fill(\boldsymbol{Q})$. The total number of possible non-zero elements is thus $n_{\boldsymbol{L}} = \frac{1}{2}(|fill(\boldsymbol{Q})| + n)$. □

4.1 Fill-In Ratio

It is now clear that in order to assess the computational cost of calculating $\boldsymbol{L}$ the relevant measure is the size of the fill. For sparse matrices a convenient way of expressing it is the fill-in ratio

$$fir = \frac{n_{\boldsymbol{L}}}{n_{\boldsymbol{Q}}} \tag{7}$$

where $n_{\boldsymbol{Q}}$ is the number of non-zero entries in the lower triangular part of $\boldsymbol{Q}$, i.e. $n_{\boldsymbol{Q}} = \frac{1}{2}(\|\boldsymbol{Q}\|_0 + n)$. The fill-in ratio hence is just the ratio between the number of possible non-zero elements in $\boldsymbol{L}$ and the number of non-zero elements in the lower triangular part of $\boldsymbol{Q}$. Of special interest is how the fill-in ratio scales with the dimension of $\boldsymbol{Q}$ and parameters of a underlying model. We will provide explicit results for special designs of the 2-factor crossed random effects model in Section 6.

Corollary 4.1.3. *For a matrix $\boldsymbol{Q}$ with Cholesky factor $\boldsymbol{L}$ the fill-in-ratio is always greater or equal to 1.*

Proof. Since a non-zero entry in $\boldsymbol{Q}$ implies that the corresponding edges are in $fill(\boldsymbol{Q})$, it follows directly that $n_{\boldsymbol{Q}} \leq n_{\boldsymbol{L}}$. The equality only holds if $\boldsymbol{Q}$ and $\boldsymbol{L}$ have the same sparsity structure. □

It should again be stressed that this does not mean that $\boldsymbol{L}$ is always more or equally dense than $\boldsymbol{Q}$. There can be less non-zero entries in $\boldsymbol{L}$, but these cannot be anticipated by the future set.

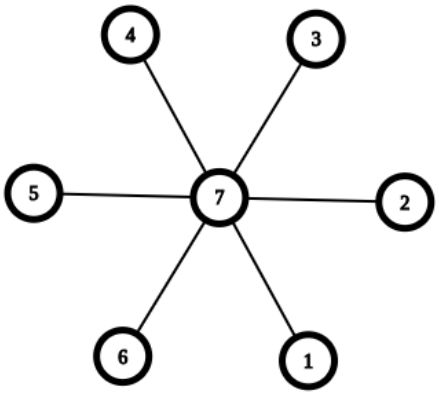
4.2 Sparsity and Labelling of Vertices

It should be noted that what the future set is depends on the labelling of the vertices, which can be chosen freely. This means, what is called node 1 and what is called node 7 does not matter since the resulting graph will remain the same. It does however matter for the sparsity of $\boldsymbol{L}$. This is straightforward to see from a simple example (2.4.2 in Rue and Held [2004]).

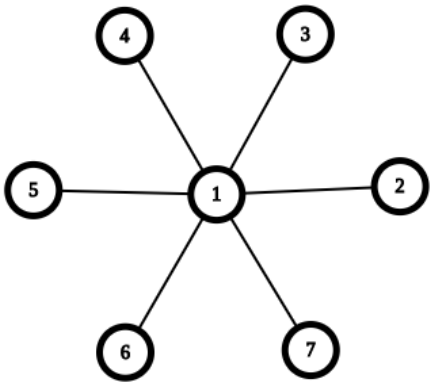
The two graphs in Figure 3 are equivalent. The only difference is that in the left graph the central node is labelled as 7 and in the right graph as 1. For both graphs $\boldsymbol{Q}$ is equally sparse, but $\boldsymbol{L}$ is not. In the left graph, for any pair of nodes $i \neq 7, j \neq 7$ and $i < j$, the central node 7 lies in $F(i, j)$. As node 7 blocks all paths between other nodes, according to equation (4.1.1), $L_{ij} = 0$ for these pairs of i, j. In the right graph, however, the central node is labelled 1 and thus never lies in any future set. Consequently, the future set of any $i < j$ cannot block the path between i and j. (i, j) and (j, i) are thus in $fill(\boldsymbol{Q})$ and $L_{ij} \neq 0$ has to be assumed.
This illustrates that permuting the precision matrix can increase the sparsity of $\boldsymbol{L}$. Most algorithms for sparse calculation of the Cholesky factor therefore rely on finding a permutation with permutation matrix $\boldsymbol{P}$ such that

$$\boldsymbol{Q}_P = \boldsymbol{P}\boldsymbol{Q}\boldsymbol{P}^T$$

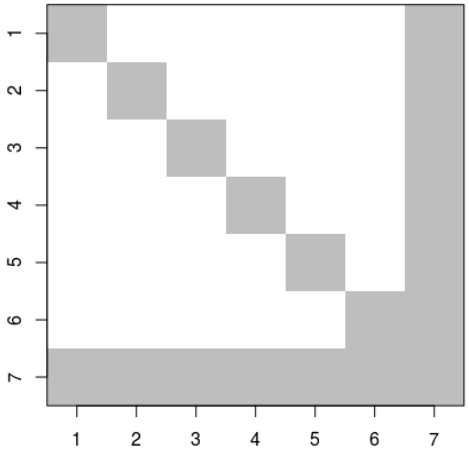
and its corresponding Cholesky factor have a reduced fill-in ratio.



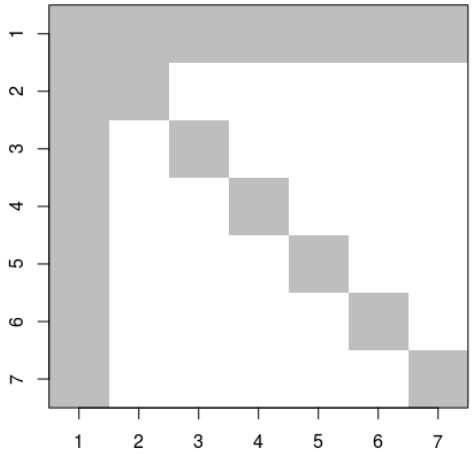
(a) The central node is 7 and it lies in $F(i,j)$ for all $i,j \neq 7$.



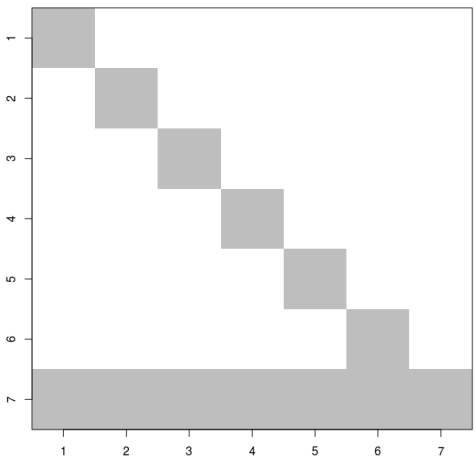
(b) The central node is 1 and it lies in no $F(i,j)$ for any $i,j \neq 1$.



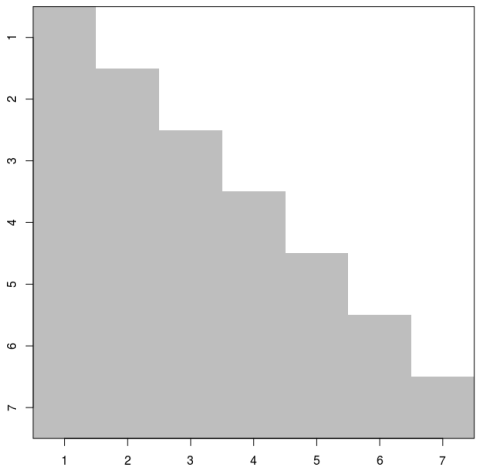
(c) The Precision Matrix corresponding to graph with 7 as central node.



(d) The Precision Matrix corresponding to graph with 1 as central node.



(e) The Cholesky factor of the precision matrix corresponding to the graph with 7 as central node is maximally sparse.

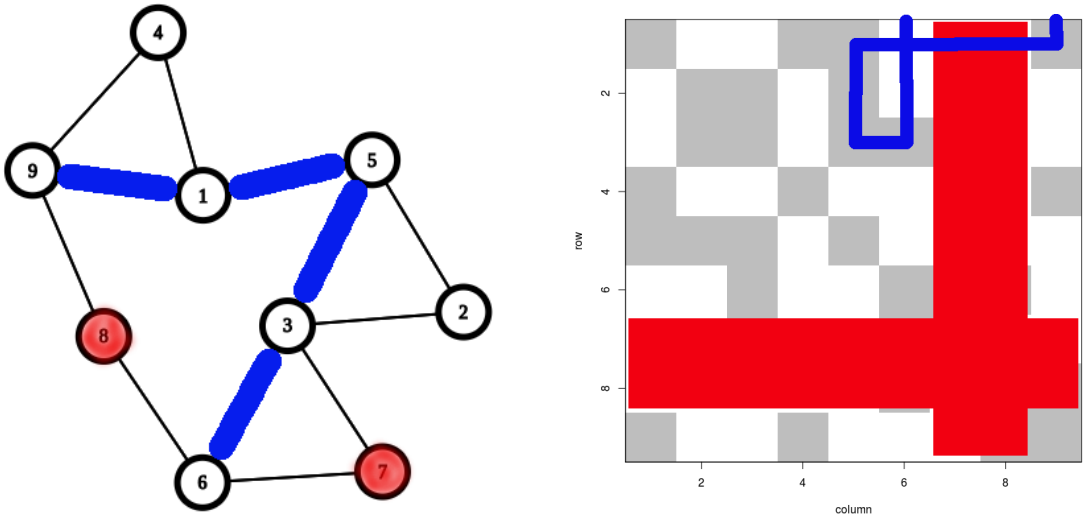


(f) The Cholesky factor of the precision matrix corresponding to the graph with 1 as central node is full.

Figure 3: The ordering of nodes matters for the sparsity of $\boldsymbol{L}$: Every pair of nodes is independent given their future set in the left graph, but no pair of nodes in the right graph. The Cholesky factor of the left precision matrix is thus maximally sparse, whereas the one of the right matrix is full.

4.3 Equivalence of Paths in Q and L

In order to get an intuition of what conditional independence given the future set means, it is instructive to look at how a path in the graph can be understood in the precision matrix and how the future set relates to it. We already know that an edge between node i and j is just an entry in Q_{ij}. If now in the graph also k is connected to j, but not to i, one could move from i to k via j. In $\boldsymbol{Q}$ this means moving from the non-zero entry Q_{ij} to Q_{jk} or Q_{kj} which is just moving along the row or column from Q_{ij}. In other words, moving from node to node via edges is just moving from one non-zero entry in $\boldsymbol{Q}$ to another one along the rows or columns of the matrix. If one now wants to condition on the future set or equivalently check whether $(i, j) \in fill(\boldsymbol{Q})$, this simply means not using the corresponding rows and columns. An example is shown in Figure 4, where for a given graph one is interested if L_{96} can be non-zero. This principle is useful if one wants to make explicit statements about the sparsity of $\boldsymbol{L}$ for cases where the precision matrix has a pattern that cannot be represented instructively in the conditional independence graph. In particular, we will take advantage of it in Section 6.4.



(a) Graph

(b) Precision matrix

Figure 4: Equivalence of a path in a conditional independence graph and the corresponding precision matrix: We are interested in whether $L_{96} \neq 0$, i.e. if $F(6, 9)$ blocks all paths between 6 and 9 or if $(9, 6) \in fill(\boldsymbol{Q})$. In the graph of $\boldsymbol{Q}$ the future set is immediate to see, it corresponds to the red nodes. We can see that there exists a path between nodes 6 and 9 that does not pass a node in the future set. $(6, 9)$ and $(9, 6)$ are thus in $fill(\boldsymbol{Q})$. An edge in the graph corresponds to a non-zero entry in the precision matrix. Two edges are thus connected if they share a common node, i.e. are in the same row or column in the precision matrix. If one wants to find a path in the precision matrix, one can therefore move from non-zero entry to non-zero entry along rows and columns. In the figure we can for example start at node 6 and move to row 3, because there is a non-zero entry at Q_{36}. From there, we can move along the row to Q_{35}, then to Q_{15} and finally to Q_{19}. The path is marked blue and is equivalent to the path in the graph. As we did not use columns or rows that belong to the future set (marked red), we can conclude that $(9, 6) \in fill(\boldsymbol{Q})$ and $L_{96} \neq 0$ has to be assumed.

5 Special Cases of the 2-Factor Crossed Random Effects Model

For the rest of this paper we will focus on making explicit statements regarding the sparsity structure of $\boldsymbol{Q}$ and $\boldsymbol{L}$ for the 2-factor random crossed effects model. Since it is hard to derive results for the most general case, we will introduce special arrangements and designs which allow to derive explicit results and give a

framework that simplifies sampling. These special cases are then investigated analytically in Section 6 and numerically in Section 7. We will start by first deriving the general structure of the precision matrix $\boldsymbol{Q}$ of the posterior distribution.

5.1 Calculating $\boldsymbol{Q}$

Recalling the definition of the 2-factor random crossed effects model in equation (1) we can explicitly calculate the non-zero entries of $\boldsymbol{Q}$ by applying Bayes theorem:

$$\begin{aligned} p(\boldsymbol{\beta}|\boldsymbol{y}) = & \\ &= \frac{p(\boldsymbol{y}|\boldsymbol{\beta})p(\boldsymbol{\beta})}{p(\boldsymbol{y})} \\ &\propto p(\boldsymbol{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}) \\ &\propto \exp\left(\boldsymbol{\beta}^T\boldsymbol{Q}\boldsymbol{\beta}\right) \end{aligned} \tag{8}$$

This can be used to read off the entries of the precision matrix $\boldsymbol{Q}$. The detailed calculation can be found in Appendix A. For convenience, we define $n_{ij} = 1_{(i,j)\in S}$ as 1 if we have an observation for a_i and b_j and 0 else. $N = \sum_{i,j} n_{ij}$ is then the total number of observations, $n_{i:} = \sum_j n_{ij}$ the number of times a_i was observed and $n_{:j} = \sum_i n_{ij}$ the number of times b_j was observed. The entries of the precision matrix $\boldsymbol{Q}$ for $\boldsymbol{\beta}|\boldsymbol{y}$ are then:

$$\begin{aligned} Q_{\mu\mu|\boldsymbol{y}} &= N\tau_e \\ Q_{\mu a_i} &= n_{i:}\tau_e \\ Q_{\mu b_j} &= n_{:j}\tau_e \\ Q_{a_i a_i} &= \tau_a + \tau_e n_{i:} \\ Q_{b_j b_j} &= \tau_b + \tau_e n_{:j} \\ Q_{a_i b_j} &= \tau_e \end{aligned} \tag{9}$$

where $i \in (1, ..., I)$ and $j \in (1, ..., J)$. It is to note that conditional on $\boldsymbol{\mu}$ the conditional independence graph of $\boldsymbol{Q}$ is bipartite, i.e. there are no direct links between customers or products. This implies a natural ordering of the model parameters:

5.2 $ab\mu$ Ordering

Assigning lower labels to customers a_i than to products b_j and the highest label to μ will be referred to as $ab\mu$ ordering. In particular this means that customers have labels $\{1, ..., I\}$, products have labels $\{I + 1, ..., I + J\}$ and μ has label $I + J + 1$. In Section 6.2 we will prove that assigning this label to μ is optimal for reduced fill-in.

As a consequence of $ab\mu$ ordering, the precision matrix for the balanced level case with $I = J$ can be split into 4 blocks that fully describe the matrix:

$$\begin{aligned} \boldsymbol{Q}_{aa} &= \boldsymbol{Q}_{1:I,1:I} \\ \boldsymbol{Q}_{bb} &= \boldsymbol{Q}_{I+1:2I,I+1:2I} \\ \boldsymbol{Q}_{ab} &= \boldsymbol{Q}_{1:I,I+1:2I} \\ \boldsymbol{Q}_{\mu} &= \boldsymbol{Q}_{1:2I+1,2I+1} \end{aligned} \tag{10}$$

The $\boldsymbol{Q}_{aa}$ block describes all direct links between customers, the $\boldsymbol{Q}_{bb}$ part the ones between products and the $\boldsymbol{Q}_{ab}$ part all direct links between customers and products. Due to the bipartite structure of the model both $\boldsymbol{Q}_{aa}$ and $\boldsymbol{Q}_{bb}$ are diagonal matrices. The two precision matrices in 5 are examples of $ab\mu$ ordering.

5.3 Balanced Levels

Balanced levels mean that each customer a_i is linked to exactly $n^{(a)}$ products and similarly, each product b_j is linked to exactly $n^{(b)}$ customers. For many derivations we will additionally assume $I = J$, i.e.

$$n^{(a)} = n^{(b)} \equiv d \tag{11}$$

The conditional independence graph is then d-regular.

5.4 Markovian Design

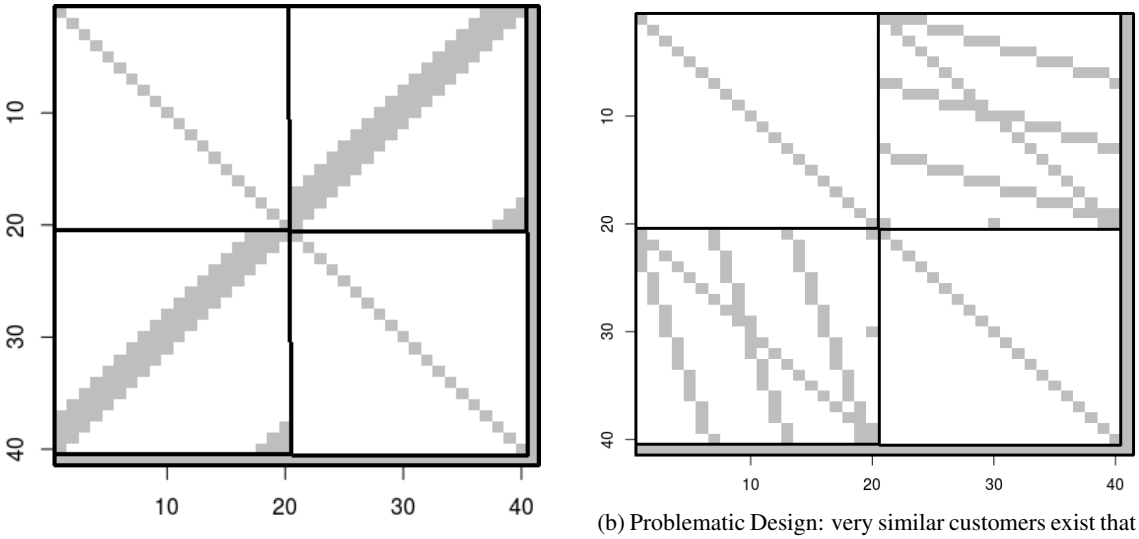
In this arrangement we assume balanced levels, $I = J$ and $ab\mu$ ordering. We furthermore require that customer 1 has rated the last d products ($\{2I-d, ..., 2I\}$), customer 2 the products $\{2I-d-1, ..., 2I-1\}$, ..., customer I the products $\{I+1, 2I-d+1, ..., 2I\}$. Then the customer-product block $\boldsymbol{Q}_{ab}$ has anti-diagonal structure. Note that this arrangement within the given class of graphs means that neighbouring customers have maximal overlap in the products they rated. The corresponding precision matrix can be seen in Figure 5 for $I = 20, d = 4$. This design is referred to as Markovian Design because the conditional independence graph is similar to a Markov chain with periodic boundary conditions.

5.5 Problematic Design

In this arrangement we again assume balanced levels, $I = J$ and $ab\mu$ ordering. The precision matrix of this case can be seen in Figure 5. The arrangement corresponds to a case where neighbouring customers have almost minimal overlap in their rated products. In general similar customers exist, but every customer has an almost individual product (on the diagonal) that for many customers is fairly far away from the rest of their products, which are aligned next to each other. For given I, d the customer-product matrix $\boldsymbol{Q}_{ab}$ can be obtained from the following algorithm:

Result: Fills the customer-product part of the precision matrix according to the Problematic Design
Initialize $\boldsymbol{c}$ as an empty array;
Initialize $pivot$=1;
Initialize $\boldsymbol{Q}_{ab}$ as an $I \times I$ matrix of zeros;
for *r=1:(I-1)* **do**
  $\boldsymbol{Q}_{ab}(r, r) = 1$;
  for *i in 1:d-1* **do**
    if $r == pivot$ **then**
      append $pivot$ to $\boldsymbol{c}$;
      $pivot = pivot + 1$;
    end
    if $pivot == I + 1$ **then**
      $pivot = 1$;
    end
    $\boldsymbol{Q}_{ab}(r, pivot) = 1$;
    $pivot = pivot + 1$;
  end
end
Set $\boldsymbol{Q}_{ab}(I, \boldsymbol{c}) = 1$;

The design is referred to as problematic because it creates a fill-in ratio that increases linear with I (see Section 6.4) and thus a possibly dense Cholesky factor.



(a) Markovian Design

(b) Problematic Design: very similar customers exist that differ mostly by one product.

Figure 5: Precision matrices for Markovian Design and Problematic Design for $I = 20, d = 4$. The black lines show the separation that is induced by the $ab\mu$ ordering: The left upper block is the $\boldsymbol{Q}_{aa}$ block, right next to it is the $\boldsymbol{Q}_{ab}$ block. Below the $\boldsymbol{Q}_{ab}$ block is the $\boldsymbol{Q}_{bb}$ block. The last row/column corresponds to the $\boldsymbol{Q}_{\mu}$ blocks.

5.6 Erdös-Renyi

The Erdös-Renyi case can be seen as the probabilistic version of the balanced-level-case, because we only require d-regularity in expectation. For the case of $I = J$, this is achieved if there is a direct link between every customer a_i and every product b_j with probability p and there is no direct link with probability $1 - p$, where

$$p = \frac{d}{I} \tag{12}$$

Now the number of products a customer has rated and vice versa will follow a Binomial distribution with parameters I and p. This implies that in expectation every product will be rated by $I \times p = d$ customers and every customer will be linked to d products. Random draws from the precision matrix of the Erdös-Renyi case are easy to generate and therefore especially interesting for numerical experiments.

6 Sparsity of Special Arrangements of the 2-Factor Crossed Random Effects Model

In this section we want derive explicit results for the 2-factor crossed random effects models. For this we will mostly consider the special arrangements that were introduced in Section 5.

Theorem 6.1. *In the balanced-levels-case, the precision matrix is sparse for $I = J$ and $d \ll I$. More specifically,*

$$\|\boldsymbol{Q}\|_0 = I(6 + 2d) + 1$$

Proof. In the balanced level-case for $I = J$ we have $n_{i:} = n_{:j} = d$. As customers and products are only linked to themselves directly, they create $2I$ entries. μ is linked to all customers and all products as well as to itself and creates thus $4I + 1$ entries in $\boldsymbol{Q}$. The customer-product links are equivalent to the total number of observations $N = \sum_{i,j} n_{ij} = Id$ which leads to $2Id$ entries in $\boldsymbol{Q}$. Summing up these entries gives

$$\|\boldsymbol{Q}\|_0 = I(6 + 2d) + 1$$

from which $n_{\boldsymbol{Q}}$ follows trivially. As the number of non-zero entries is of $\mathcal{O}(I)$ for $d \ll I$, $\boldsymbol{Q}$ is a sparse matrix. □

6.1 The Position of μ

Even for general crossed random effects models a strong statement about the position of μ can be obtained:

Theorem 6.2. *The fill-in ratio is a non-increasing function of the label of μ.*

Proof. Let u be the label of μ and ν be the parameter corresponding to label $u+1$. As the number of non-zero entries of $\boldsymbol{Q}$ is independent of the labelling of nodes, it is sufficient to show that $n_{\boldsymbol{L}}$ is a non-increasing function of u by showing that $|fill(\boldsymbol{Q})|$ is a non-increasing function of u. Since μ is connected to all other nodes there is a path between any nodes $i \neq j$ that only passes through μ. All pairs of nodes (i,j) for which $u < min(i,j)$ will hence be in $fill(\boldsymbol{Q})$. Now consider the case where μ and ν swap labels, i.e. μ has label $u' = u+1$ and ν has label u. The resulting fill is denoted with $fill'(\boldsymbol{Q})$. Then all pairs of nodes (i,j) for which $u' < min(i,j)$ are in $fill'(\boldsymbol{Q})$ and

$$i,j \in fill'(\boldsymbol{Q}) \implies i,j \in fill(\boldsymbol{Q})$$

because all other pairs of nodes that are connected through paths not involving μ are not affected by the swap. Hence,

$$|fill(\boldsymbol{Q})| \geq |fill'(\boldsymbol{Q})|$$

□

From theorem 6.2 it follows immediately that in order to reduce the fill-in ratio one should always assign the highest possible label to μ, which is the case for the $ab\mu$ ordering. Then only the direct links to μ will create non-zero entries in $\boldsymbol{L}$. These sum up to $I+J+1$ (counting the $\mu-\mu$ entry as well).

6.2 Fill-In of $ab\mu$ Ordering with Balanced Levels

It is convenient to consider the $ab\mu$ ordering with balanced levels and $I=J$, because it gives a framework where it is possible to explicitly calculate the fill-in-ratio of certain configurations:

Theorem 6.3. *Whether or not a precision matrix $\boldsymbol{Q}$ with balanced levels, $ab\mu$ ordering and $I=J$ can have a sparse Cholesky factor depends only on product-product paths. In particular, the number of possible non-zero entries in the Cholesky factor is*

$$n_{\boldsymbol{L}} = I(3+d) + 1 + n_{bb}$$

where n_{bb} are the number of possible non-zero entries that are created through paths from one product to another.

Proof. The proof relies on theorem 4.1.1. With the $ab\mu$ ordering, all products lie in the future set of any two customers. As there are no direct links between customers and products, the future set thus blocks every path from one customer to another. Consequently, the only customer-customer entries in the Cholesky factor will be the diagonal ones. Customer-customer paths hence create I possible non-zero entries in $\boldsymbol{L}$.

In the same way, all customer-product entries in $\boldsymbol{L}$ are the ones corresponding to the direct edges between a_i and b_j, because in order to reach any other product that is not directly linked to a customer, one would have to move through the directly linked products. These, however, always lie in the future set of a customer and another product. The customer-product paths hence create Id non-zero entries in $\boldsymbol{L}$.

Since we already know from Section 6.2 that the connections with μ create $2I+1$ entries, the only part that is left to analyze are thus the connections between products and products. These are non-trivial, because no customer is part of the future set of two products and therefore every customer can be used to connect products. If we denote the number of possible non-zero entries from product-product paths with n_{bb}, we can just sum up the non-zero entries from the above calculations:

$$\|\boldsymbol{L}\|_0 = (2I+1) + (I) + (Id) + n_{bb} = I(3+d) + 1 + n_{bb}$$

For sparsity of $\boldsymbol{L}$ we need this to be of $\mathcal{O}(I)$, which is true if and only if n_{bb} is of $O(I)$ and $d \ll I$. □

Please note that the $ab\mu$ ordering is not necessarily optimal. It however provides a convenient framework for determining explicit results, because one can focus on the product-product paths. As these paths will go through customers, the $\boldsymbol{Q}_{ab}$ block is of predominant interest. In the following, we will focus on what we introduced as Markovian and Problematic Design in Section 5, because they are both arrangements within the framework of $ab\mu$ ordering and we can make explicit statements about the sparsity of their Cholesky factor.

6.3 Fill-In of Markovian Design

In this section we will derive the exact fill-in-ratio of the Markovian Design.

Theorem 6.4. *The fill-in ratio of the Markovian Design is*

$$fir = \frac{I(3d+2) - (d-1)(2d-1) + 1}{I(4+d) + 1} \tag{13}$$

Proof. Since the Markovian Design is a special case of $ab\mu$ ordering we can use theorem 6.3 and just calculate n_{bb}, i.e. the number non-zero entries in $\boldsymbol{L}$ that result from paths starting and ending at products. In the following we will count the product-product paths that cannot be blocked by the future set. Recall that for any two products, no customer is in the respective future set. From the definition of Markovian Design one then observes that every product is connected via one customer to $2(d-1)$ other (neighbouring) products. In Figure 6 this is shown for product 12 (green) that is connected to its 4 neighbours (blue), because they were rated by the same customer. Customer 12 could for example not reach product 9, because therefore the path would have to go through either node 10 or node 11, which both lie in $F(9, 12)$. Taking into account also the diagonal entries, i.e. the links between a product and itself, these direct links thus create d unique non-zero entries in $\boldsymbol{L}$ per product. This sums up to a total of $I \times d$.

The only other paths between two products that cannot be blocked by the future set are paths to the last $d-1$ products (yellow in Figure 6): Since the future set of a product j and one of the last $d-1$ products does not involve products with labels smaller than j, these lower-labelled products can be used to move until the last customer. The last customer, however, is directly linked to the last $d-1$ products. In other words, every product is also connected to the last $d-1$ products through a path that cannot be blocked by the future set. This can again best be seen from the exemplary node 12: In order to reach the last customer, only products with labels lower than 12 are required and these do not lie in the future set of node 12 and one of the last two products. These connections to the last products create additional $I(d-1) - (2d-1)(d-1)$ entries. The subtraction comes from the fact that some of these paths were already counted in the direct links above. All product-product paths together sum up to $I(2d-1) - (d-1)(2d-1)$. Using theorem 6.3 this yields

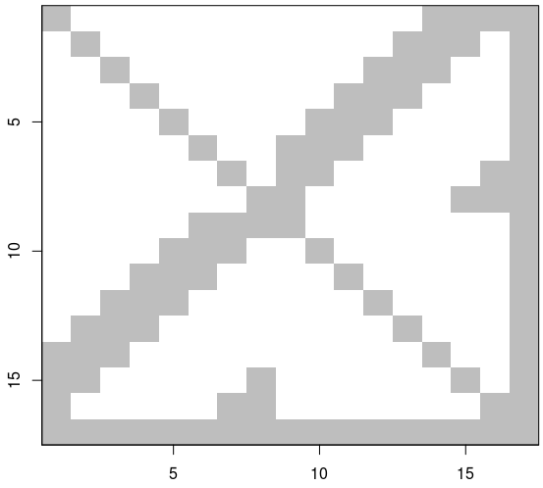
$$n_{\boldsymbol{L}} = I(3d+2) - (d-1)(2d-1) + 1$$

This together with equation (6.1) allows to calculate the fill-in ratio as

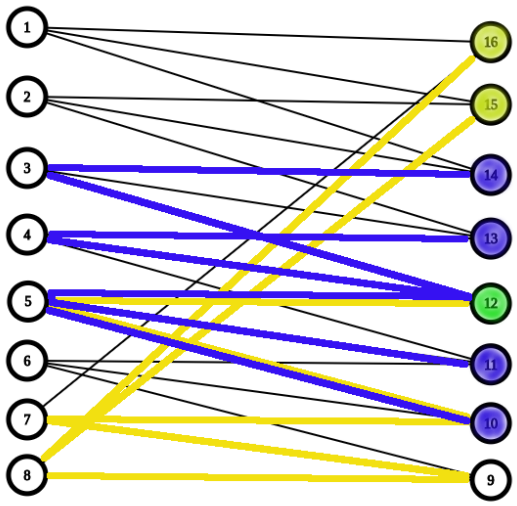
$$fir = \frac{I(3d+2) - (d-1)(2d-1) + 1}{I(4+d) + 1} \tag{14}$$

□

Equation (14) is especially interesting, because for large I and fixed d, the fill-in ratio becomes constant, i.e. (3d+2)/(4+d). This means that for this arrangement the Cholesky factor is sparse, because the number of non-zeros is of the same order as the number of non-zeros in $\boldsymbol{Q}$.



(a) Precision Matrix



(b) Graph with paths from product 12 that create non-zero entries in $\boldsymbol{L}$.

Figure 6: Precision matrix and conditional independence graph of Markovian Design. Neighbouring customers have maximal overlap in their products.

6.4 Fill-In of Problematic Design

For the Problematic Design of Section 5.5 we will not derive the exact fill-in-ratio, but only a lower bound. This, however, is enough to show that the Cholesky factor of $\boldsymbol{L}$ is possibly dense.

Theorem 6.5. *The fill-in-ratio for the precision matrix $\boldsymbol{Q}$ with the Problematic Design of Section 5.5 is $\mathcal{O}(I)$.*

Proof. First note that the above theorem is equivalent to $n_{\boldsymbol{L}}$ being of $\mathcal{O}(I^2)$. For the proof, we will again focus on the customer-product part $\boldsymbol{Q}_{ab}$ of the precision matrix and derive that the product-product interactions create $\mathcal{O}(I^2)$ entries in the Cholesky factor $\boldsymbol{L}$. For this, we will use the equivalence of a path in a graph and in the precision matrix that was explained in Section 4.3. In order to move from one node to another we can move between non-zero entries along rows and columns in the precision matrix. Assume that customers are labelled like $a_1, ..., a_I$ and products are labelled like $b_1, ..., b_I$. Let i denote the index/row of customers in $\boldsymbol{Q}_{ab}$ and j denote the index/column of the products in $\boldsymbol{Q}_{ab}$. Note that in the full precision matrix $\boldsymbol{Q}$ customer i would correspond to row/column i whereas product j would correspond to row/column $j + I$. Furthermore, define $r(j)$ as the lowest customer-index that is directly linked to product j (see orange dots in Figure 7). From the construction of the precision matrix (Algorithm 2) we can see that

$$\begin{aligned} j > r(j) \quad &\textbf{if} \quad j \geq 2 \\ j = r(j) \quad &\textbf{if} \quad j = 1 \end{aligned} \tag{15}$$

and furthermore

$$\frac{j-1}{d-1} \leq r(j) \leq \frac{j-1}{d-1} + 1. \tag{16}$$

This can also be seen intuitively from the precision matrix in Figure 7. We will now show that from product j there exists a path to every $\jmath' \in \{r(j), ..., j-1\}$ that cannot be blocked by the corresponding future set and hence (j, j') and $(j', j) \in fill(\boldsymbol{Q})$.

Every product j is linked to customer $r(j)$ by definition. Customer $r(j)$, however, is also linked to product $r(j)$ by construction of $\boldsymbol{Q}_{ab}$ (the diagonal is full). Product $r(j)$ is again linked to customer $r(r(j))$ who is linked to product $r(r(j))$. Using equation (15) one notes that this procedure can be repeated until

product 1 is reached. In other words, from every product j one can reach the first product without using the products indexed with $S_j \equiv \{r(j)+1, ..., j-1, j+1, ..., I\}$. Now define

$$T_j \equiv \{r(j), ..., j-1\}.$$

If we choose some $j' \in T_j$ we note that

$$F(j', j) \subseteq S_j \quad \forall \quad j' \in T_j.$$

This means that there exists a path from product j to product 1 that does not contain edges in $F(j', j)$. We however also note that we can reach product 1 from product j' by the exact same procedure that we applied for product j, i.e. by moving to product $r(j')$, then to product $r(r(j'))$ and so on. We also note that the path from j' to product 1 only involves products with indices smaller or equal to j' (equation (15)), i.e. indices that are not in $F(j', j)$. This means that for every j and $j' \in T_j$ there exists a path between product j and product j' through product 1 that cannot be blocked by $F(j', j)$. Hence, (j, j') and $(j', j) \in fill(\boldsymbol{Q})$. An illustration for the definition of $r(j), S_j, T_j$ can be seen in Figure 7.

For every j, there are at least $2|T_j| = 2(j - r(j))$ elements in $fill(\boldsymbol{Q})$. We are thus interested in $\sum_{j=1}^{I}(j - r(j))$ which can be bounded by

$$\begin{aligned}
\sum_{j=1}^{I} j - r(j) &\leq \sum_{j=1}^{I} \left[j - \frac{j-1}{d-1} \right] && \textit{by eq. (16)} \\
&\leq \sum_{j=1}^{I} \left[j(1 - \frac{1}{d-1}) + \frac{1}{d-1} \right] \\
&\leq (1 - \frac{1}{d-1}) \sum_{j=1}^{I} j \\
&= (1 - \frac{1}{d-1}) \frac{I(I+1)}{2} && \textit{by Gaussian formula for sums} \\
&= \mathcal{O}(I^2) \quad \textit{as } I \to \infty
\end{aligned}$$

Consequently, the size of $fill(\boldsymbol{Q})$ is $\mathcal{O}(I^2)$ and the fill-in ratio is $\mathcal{O}(I)$. □

7 Numerical Studies

In this section we will look at the performance of conventional R packages for our designs of interest. We want to confirm the results derived in Section 6 and understand if one can improve algorithmic performance by ordering $\boldsymbol{Q}$ appropriately. For the simulation of matrices from the 2-factor crossed-random effects model all precision parameters were set to 1: $\tau_a = \tau_b = \tau_e = 1$.

7.1 Spam and Matrix Packages

For the numerical computation of the Cholesky factor of matrices the two *R*-libraries *Spam* and *Matrix* were used. While *Matrix* is a package with a wide range of methods for both sparse and dense matrices, *Spam* is a set of sparse matrix functions for spatial statistics with special emphasis on Markov Chain Monte Carlo type calculations within the framework of GMRFs (Furrer and Sain [2010]).

In *Spam*, the function *chol* can be used for the calculation of the Cholesky factor. It relies on a left-looking block-sparse algorithm developed by Ng and Peyton [1991]. The base-algorithm creates blocks by grouping columns with similar sparsity structure together and then efficiently calculates the non-zero entries of $\boldsymbol{L}$ according to equation (2). In addition, one can choose to apply a permutation before running the algorithm by setting a value for $pivot$. There are two built-in options: *MMD* is the default option and applies the multiple minimum degree algorithm (George and Liu [1989]). The other option is the reverse

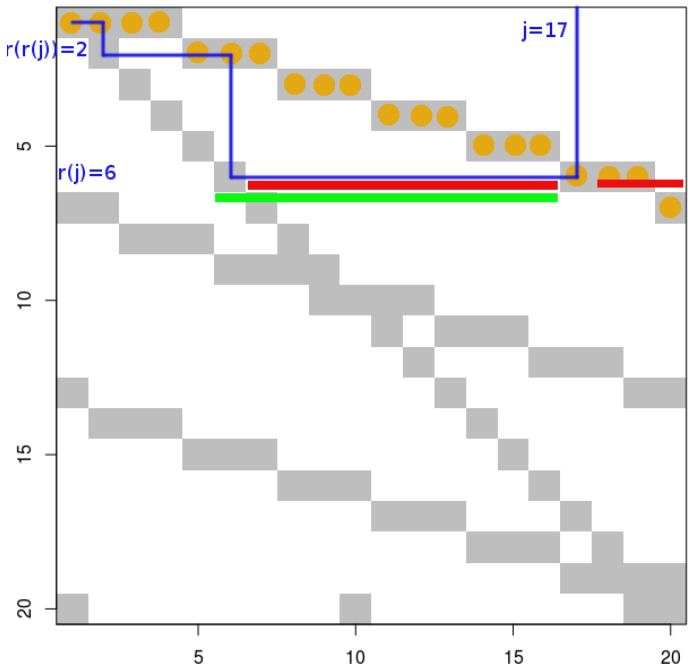


Figure 7: $\boldsymbol{Q}_{ab}$ for the Problematic Design. The orange dots mark the customer-product links with lowest customer index for each product. The row index of these dots corresponds to $r(j)$. An exemplary path from product $j = 17$ to product 1 is shown. The columns marked red correspond to the set S_j and the columns marked green to the set T_j. Note that for every product in T_j there exists a path to product $j = 17$ via product 1 that cannot be blocked by the corresponding future set, because the future set will be a subset of S_j.

Cuthill-McKee routine *RCM*, which tries to permute in order to make $\boldsymbol{Q}$ as banded as possible (11.1.5 in Golub and van Loan [2013]). The user can also specify their own permutation by passing the corresponding vector as $pivot$ argument. Note that even though one might set $pivot = FALSE$, the base-algorithm for the calculation of the Cholesky factor might still permute the input matrix since it tries to group similar columns together. Throughout this numerical analysis it was however never observed that this base permutation had an effect on the resulting fill-in. It seemed like it is more of a useful step for efficient computations that does not effect the overall sparsity of $\boldsymbol{L}$. Even though *Matrix* also comes with a sparse Cholesky routine, only the dense formalism was used, since this ensures that no base-permutation is applied. Because the dense formalism is computationally more expensive than *Spam*s algorithms, it could however not be applied to all cases of interest.

In order to assess the sparsity of $\boldsymbol{L}$ it is crucial to count the number of non-zero entries of a matrix (sparse or dense). In sparse matrix packages, the non-zero entries are stored in a vector. It turns out that *Spam* returns a vector of length $n_{\boldsymbol{L}}$, which supports the claim that $n_{\boldsymbol{L}}$ is an appropriate quantity to monitor when studying the computational cost of Cholesky factorization. In particular, the vector can contain zeros at the positions where the package assumed a non-zero element that turned out to be zero after calculating it, i.e. at numerical zeros. *Matrix* returns a matrix in dense format that does not allow to detect numerical zeros. Only $\|\boldsymbol{L}\|_0$ can be obtained. In other words, *Matrix* allows us to assess the true number of non-zero elements in $\boldsymbol{L}$ and *Spam* the possible non-zeros implied by $fill(\boldsymbol{Q})$.

7.2 Position of μ

First we check if the permutations provided by *Spam* are capable of understanding that for our setup μ should always be put at the last position. In Figure 8 the fill-in ratio is shown as function of initial μ-position for Markovian Design and $I = 30, d = 4$. For both the *MMD* and the *RCM* permutations the fill-in

is independent of the position of μ and fairly close to the $pivot = FALSE$ option for the last positions. In other words, both permutation algorithms understand that μ has to be put at a later position. The base-algorithm, however, does not change the position of μ which leads to a highly increased fill-in if μ is put at the first positions. For $I = 30, d = 4$ a fill-in-ratio of 8 means a full Cholesky factor. It can also be seen that applying *Spam*s pre-permutation is actually slightly worse than just leaving the initial Markovian Design with μ at the last position. In Figure 9 the re-ordered $\boldsymbol{Q}$ matrix is shown. The permutations obtained from the base- and the *RCM* algorithm lead to fairly similar matrices since they both seem to try to create a banded matrix. The *RCM* algorithm, however, does not assign the highest possible label to μ, which is the reason for the slightly higher fill-in ratio in Figure 8. Unsurprisingly, no algorithm preserves the $ab\mu$ ordering.

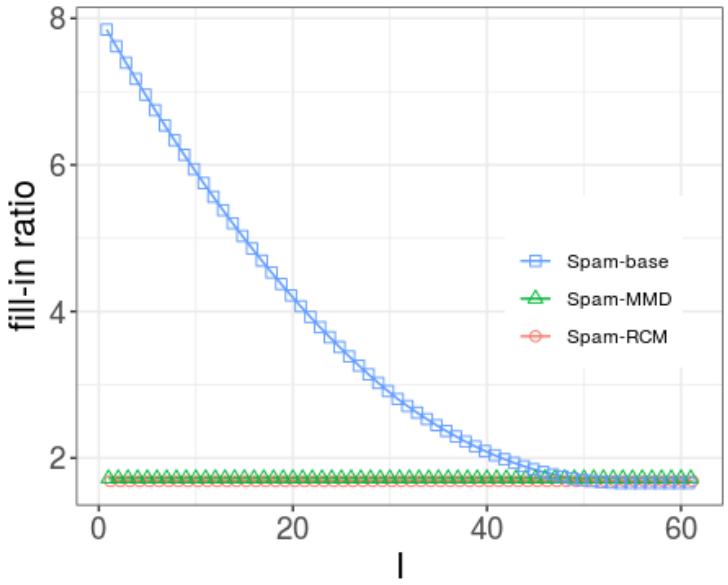


Figure 8: Fill-in-ratio with *Spam*s pivot options for different positions of μ for Markovian Design.

7.3 Markovian Design

In Figure 10a the fill-in ratio obtained with the *Spam* routines as well as the one obtained with the dense *Matrix* routine are shown for different values of I and constant $d = 4$ for Markovian Design. It is to note that *Spam* does not find a permutation that leads to a lower fill-in-ratio than the one calculated explicitly

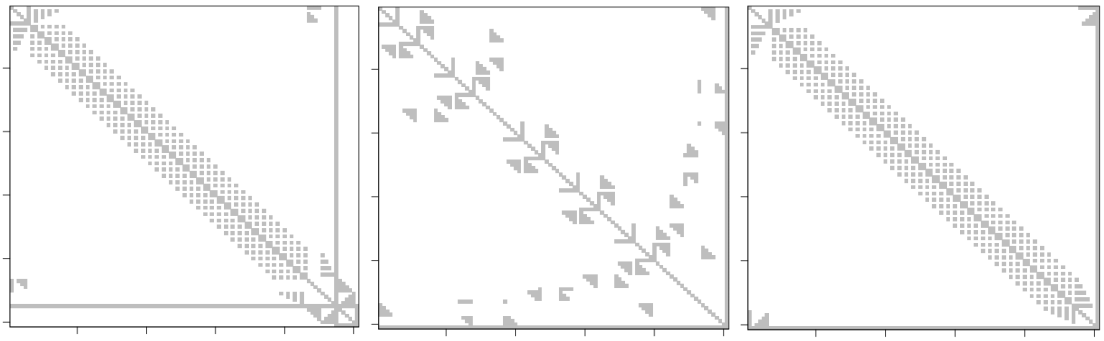
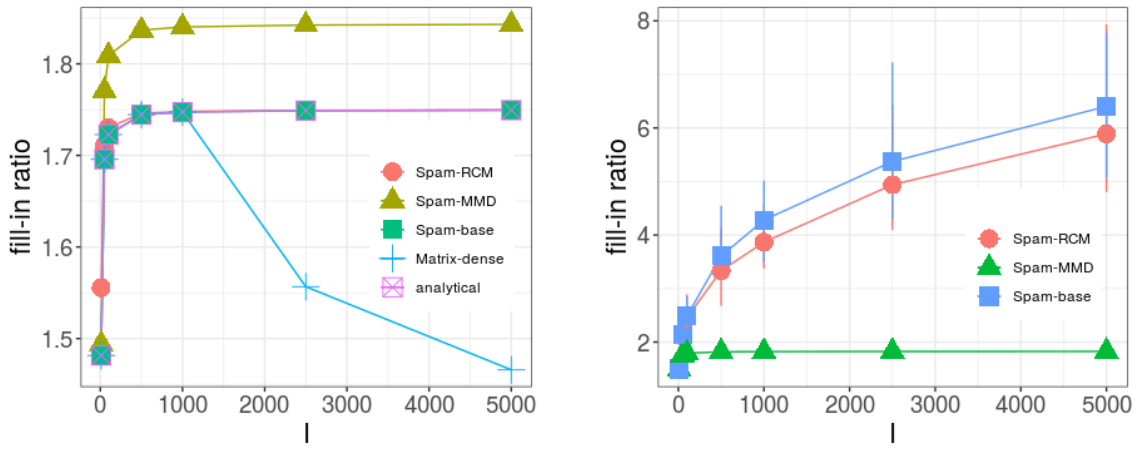


Figure 9: Ordering of $\boldsymbol{Q}$ obtained from *RCM* (left) and *MMD* (middle) and the *Spam* base algorithm for Markovian Design.



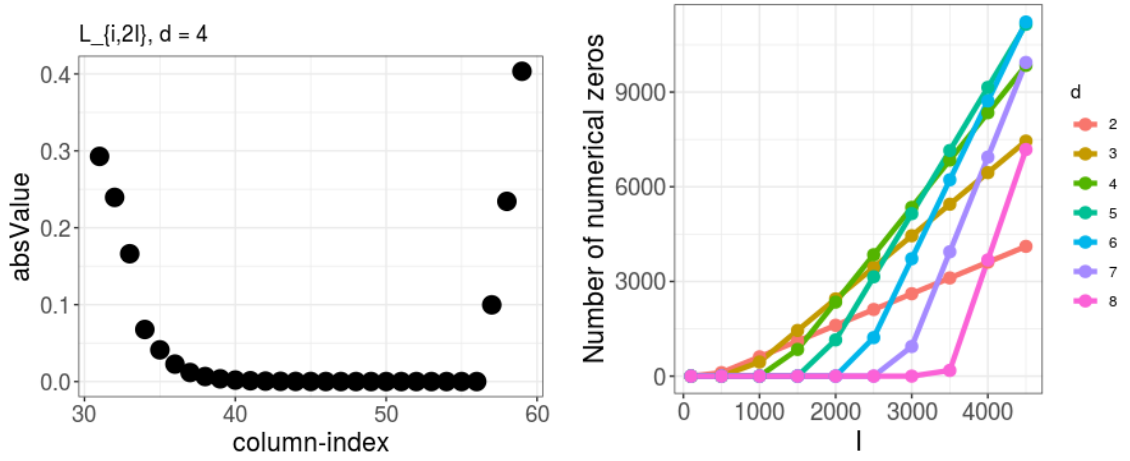
(a) Unpermuted $\boldsymbol{Q}_{ab}$

(b) Permuted $\boldsymbol{Q}_{ab}$

Figure 10: Fill-in-ratio of Markovian Design with *Spam*s pivot options for unpermuted and permuted $\boldsymbol{Q}_{ab}$ matrix. Bars indicate minimum and maximum values. In the permuted case only *MMD* is capable of recovering a constant fill-in ratio.

(see equation (14)). It is also to note that *Spam*s base-algorithm performs best and returns the exact fill-in ratio that was obtained analytically. The default permutation, *MMD*, leads to a slightly increased fill-in and the *RCM* permutation yields a fill-in that is very close to the one of the base-algorithm. Most importantly, the fill-in ratio remains constant for large I for all *Spam* routines, which is the crucial part for efficient calculation of $\boldsymbol{L}$. What calls the attention is that the fill-in ratio for dense matrices is significantly lower than the one obtained from *Spam* for large I. As explained, this means that the true number of non-zero values is much lower than the one anticipated by *Spam*. These zero-values, however, are not typical numerical zeros in the sense that $Q_{ij} \approx \sum_k L_{ik} L_{jk}$. This was seen by adding random noise to all entries of the precision matrix. Since the number of observed zeros remained the same, we concluded that these non-zero values cannot be due to conventional numerical cancellation, but they are rather the result of weak dependence. In fact, all these zeros are at positions that correspond to product-product paths involving the last $d-1$ products. Recall that many products are only connected to the last products through many other products of lower label and many customers, i.e. through many other nodes. The reason for these values being so small is thus that the division from equation (2) is applied many times which leads to values that are so small that they become numerically zero. A confirmation for this can be seen in Figure 11, where on the left the absolute value of the entries of the Cholesky factor corresponding to paths between products and the last product are shown for $I = 30, d = 4$. Since products 1,2,3,27,28,29 are connected through one customer only, their corresponding entry is large. For products with label > 3 the absolute value of the entry in the Cholesky seems to decay exponentially with product index. On the right of Figure 11 one can see the number of numerical zeros as a function of I for different values of d, which for large enough I seems to be linear. Since also the number of non-zero values in $\boldsymbol{L}$ grows linearly in I, the computational cost of calculating these values is possibly non-negligible. Furthermore, there are not only these values that are stored as exact numerical zeros, but there is also a big number of very small entries in $\boldsymbol{L}$. One could possibly obtain a 'good enough' approximation of the Cholesky factor by sparing to compute these values. This relates directly to approximate Cholesky computations and will be elaborated further in Section 9.

In order to find out if the initial arrangement of the Markovian Design is preferable over other orderings of the same class a permutation was applied. The permutation was performed such that the $ab\mu$ block structure of the customers and products remained, i.e. customers were only permuted with other customers and products only with other products. This leads to the $\boldsymbol{Q}_{ab}$ block not being anti-diagonal any more, whereas all other blocks keep their structure. The result of the performance of the different $\boldsymbol{Spam}$ routines on the permuted Markovian Design can be seen in Figure 10b, where average fill-in ratios for 25 random permutations of the Markovian Design are shown. The bars indicate min/max values. Only *MMD* is capable of recovering a ordering such that the fill-in ratio is equal to the one it obtained in the non-permuted case



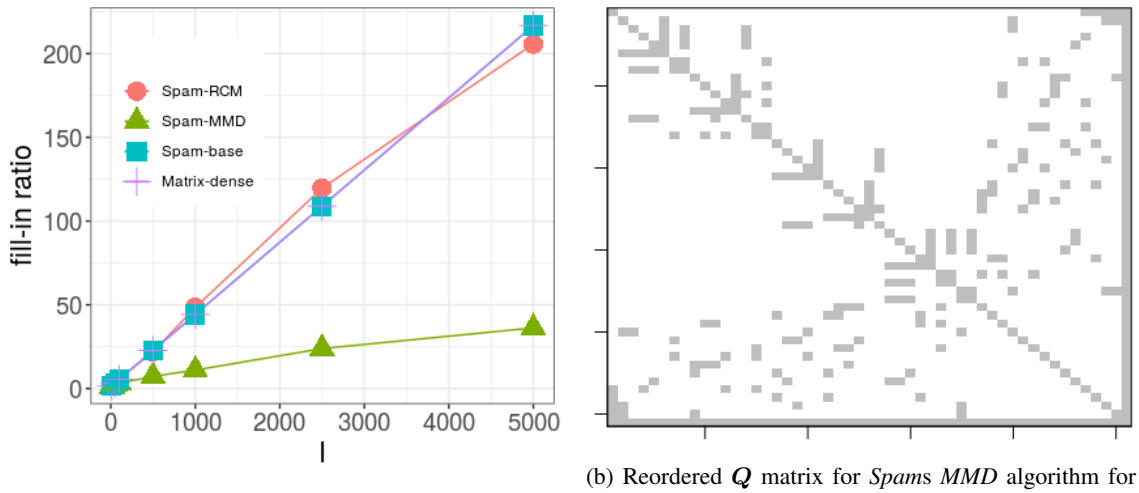
(a) Absolute values of entries in $\boldsymbol{L}$

(b) Number of Numerical zeros

Figure 11: a): Absolute value of last product row of $\boldsymbol{L}$ for $I = 30, d = 4$, i.e. $\boldsymbol{L}_{2I,(I+1):2I}$. These entries correspond to paths between products and the last product. The diagonal entry is not shown.
b): Number of numerical zeros as a function of I for different values of d.

(which in particular is constant for large I). *RCM* performs surprisingly bad and is hardly able to improve the fill-in compared to the base-algorithm. Even though it cannot be confirmed that the fill-in obtained from *RCM* is in fact increasing linearly with I, it is obvious that there are permutations of the Markovian Design leading to increased fill-in. In other words, for all what we observed, the Markovian Design was optimal for its class of graphs.

7.4 Problematic Design



(a) Fill-in-ratio with *Spam* permutations.

(b) Reordered $\boldsymbol{Q}$ matrix for *Spams MMD* algorithm for $I = 25, d = 4$.

Figure 12: Problematic Design

The performance of the *Spam* routines on the Problematic Design introduced in Section 6.4 is shown in Figure 12a. The fill-in ratio grows linearly in I if one applies the *RCM* routine or decides not to pre-permute. In particular, the fact that the base-algorithm leads to a fill-in ratio that scales linearly in I can be seen as a confirmation of the result derived in Section 6.4. Even though *MMD* performs significantly

better than the other routines, the fill-in ratio still increases linearly in I, but with a much smaller slope. Nevertheless it is interesting to look at the permutation obtained from *MMD* since it provides by far the best results. In Figure 12b the re-ordered $\boldsymbol{Q}$ matrix can be seen. Even though it has to be taken into account that this ordering also includes the permutations applied by the base-algorithm, it is to note that the resulting matrix looks fairly similar to the $ab\mu$ ordering. In particular, μ is at the last position and there is a quadratic block with only diagonal elements on the lower right part. The Cholesky factorization of the Problematic Design created no exact numerical zeros. However, it was found that a big fraction of the entries in $\boldsymbol{L}$ are very small. For $I = 5000, d = 4$ for example approximately 98.5% of the entries are smaller than 10^{-4} in absolute value.

7.5 Erdös-Renyi

Finally, the performance of the *Spam* routines on the Erdös-Renyi design was investigated. The resulting average fill-in ratios are shown in Figure 13. Min/Max values are indicated with bars, but deviate so little from the average value that they can be barely noticed. Again, the fill-in ratio grows linearly in I for all methods and *MMD* outperforms the other methods. It is interesting to see that the fill-in ratio obtained from *RCM* is significantly larger than the one from the base-algorithm. In the Erdös-Renyi design, apart from d-regularity in expectation, no structure on the connections between customers and products is assumed, making it a very relevant design for realistic 2-factor crossed random effects models. However, all methods lead to a dense Choleksy factor, which indicates that it might be hard or even impossible to find a general ordering that yields a sparse $\boldsymbol{L}$. For the Erdös-Renyi design no numerical zeros were found in the Cholesky factor. However, for for a matrix with $I = 5000, d = 4$ more than 90% of all entries are smaller than 10^{-4} in absolute value.

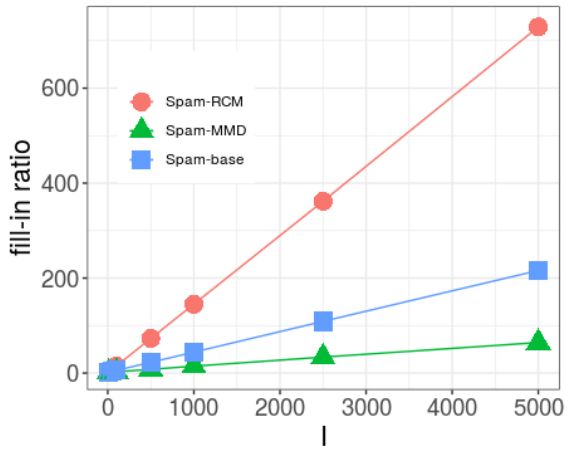


Figure 13: Fill-in ratio for Erdös-Renyi design.

8 Summary

In this thesis we focused on analyzing 2-factor crossed random effects models and their scalability for inference. We showed that the computational bottleneck for MCMC based methods is the calculation of the Cholesky factor and introduced methods for the calculation of $\boldsymbol{L}$ in the case of sparsity. We in particular showed that the crucial part for efficient calculation is the sparsity structure of $\boldsymbol{L}$ itself. We expressed the computational cost through the number of 3-cycles in the fill-graph of the precision matrix and proved upper and lower bounds. We then introduced graphical models in order to assess conditions for sparsity that can be derived with probabilistic arguments. In particular, we found that reordering the precision matrix can increase the sparsity of the Cholesky factor crucially. We introduced the Markovian Design and the Problematic Design as 2 exemplary designs for which we showed sparsity and density of the Cholesky

factor, respectively. We then numerically analyzed the performance of the *Spam* package on these designs and the Erdös-Renyi design. We found indications that in general it might be hard or even impossible to find a generic rule for ordering the precision matrix of 2-factor crossed random effects models such that the respective Cholesky factor is sparse. We however also found that possibly a big part of the non-zero entries in $\boldsymbol{L}$ is small in absolute value. In the presentation of possible future work in Section 9 we perform preliminary experiments indicating that neglecting those small values might only lead to negligible changes in the Cholesky factor (according to the Frobenius norm, which we consider as a metric). For the Markovian Design we could connect the appearence of zero and close-to-zero entries to the notion of weak dependence.

9 Future Work

9.1 Approximate Cholesky Methods

Since the numerical studies showed that a big fraction of the entries in $\boldsymbol{L}$ is either exactly zero or close to zero, this opens the door for approximate Cholesky methods. If at the price of a small error in $\boldsymbol{Q}$ one can spare to calculate these values, computational cost might be reduced greatly. It is therefore crucial to find out how many small values should be expected and whether their position can be estimated with probabilistic arguments. In Figure 14 the relative error which corresponds to neglecting a certain fraction of small values can be seen. For a given $\boldsymbol{Q}$ matrix the cholesky factor was computed. Then, a certain fraction of small entries (in absolute value) was set to zero in $\boldsymbol{L}$ and $\boldsymbol{Q}$ was reconstructed from this simplified Cholesky factor. The relative change in the Frobenius norm of $\boldsymbol{Q}$ then serves as a measure of the price one has to pay for the approximation. It can be seen that if one allows a relative error of 10^{-6}, for $I = 5000$ about 99% of the values in $\boldsymbol{L}$ can be neglected. It is also to note that the fraction of negligible values increases with I for a fixed relative error. Consequently, allowing a small error might not only decrease the number of elements that have to be computed by a prefactor, but also change how they scale with I. Future work should therefore include research on how the proportion of negligible values scales exactly with I and d and to which extent this can be used for improving the computational cost of the Cholesky factorization for different designs.

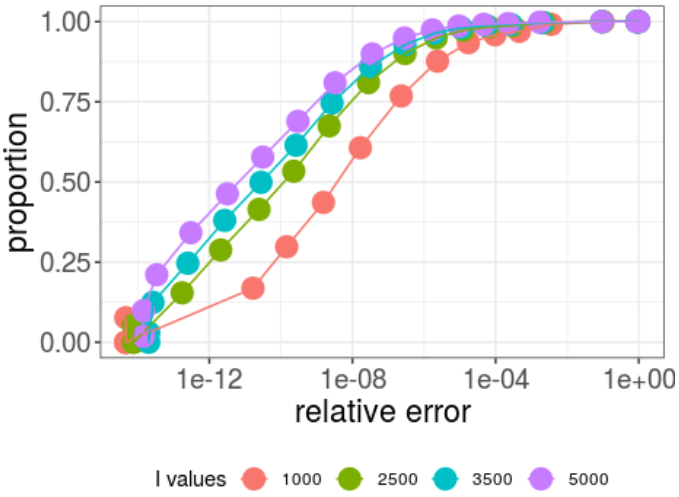


Figure 14: Problematic Design, d=4: Proportion of Values than can be neglected as a function of the resulting relative error in the Frobenius norm.

9.2 Exploration of Full Cost for MCMC Algorithm

In this thesis we investigated the computational cost of computing the Cholesky factor of a precision matrix, which is a crucial step for sampling from a normal distribution. For MCMC algorithms however, one also

has take into account that the parameters and hyperparameters of the model have to be updated regularly. We believe that this can be done efficiently, but nevertheless it should be confirmed how a fully costed MCMC algorithm works out for the case of 2-factor crossed random effects models.

9.3 Simulation of Balanced Random Designs

Since in this thesis only random graphs that were d-regular in expectation (i.e. Erdös-Renyi graphs) were investigated, another possible step in the analysis of 2-factor crossed random effects models is the investigation of random graphs that are exactly balanced. We therefore propose a modification of the configuration model as a method that can be used for drawing samples from a d-regular graph: We create two vectors of indices, where each vector corresponds to one factor. Each vector then contains every index d times. For $d = 3$ we thus have

$$\begin{aligned}\boldsymbol{a}_{ind} &= \{1, 1, 1, 2, 2, 2, ..., I, I, I\} \\ \boldsymbol{b}_{ind} &= \{1, 1, 1, 2, 2, 2..., I, I, I\}\end{aligned}$$

We then shuffle $\boldsymbol{b}_{ind}$ and place an edge between nodes $\boldsymbol{a}_{ind}[i], \boldsymbol{b}_{ind}[i]$ for every i. The resulting graph is not necessarily d-regular, as some links might appear twice. We therefore reject every non-d-regular graph and shuffle again until we get the first d-regular one, which we accept. One can show that for large I the probability of having no double edges becomes independent of I and decreases exponentially with d. This implies that this form of rejection sampling is only feasible for small d, which is typically the case for crossed random effects models.

Appendices

A Detailed Calculation of Q

Like in Section 5.1 we start by applying Bayes theorem to the 2-factor crossed random effects model:

$$\begin{aligned}
p(\boldsymbol{\beta}|\boldsymbol{y}) = & \\
&= \frac{p(\boldsymbol{y}|\boldsymbol{\beta})p(\boldsymbol{\beta})}{p(\boldsymbol{y})} \\
&\propto p(\boldsymbol{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}) \\
&= p(\boldsymbol{\beta}) \prod_{i,j} p(y_{ij}|\mu, a_i, b_j) \\
&= \prod_{i=1}^{I}\prod_{j=1}^{J} p(\mu)p(a_i)p(b_j)p(\epsilon_{ij}) \\
&= \exp(-\frac{1}{2}\sum_i a_i^2\tau_a)\exp(-\frac{1}{2}\sum_j b_j^2\tau_b)\exp(-\frac{1}{2}\sum_{i,j}(y_{ij}-\mu-a_i-bj)^2\tau_e) \\
&\propto \exp\left(-\frac{1}{2}\tau_a\sum_i a_i^2 - \frac{1}{2}\tau_b\sum_j b_j^2 - \frac{1}{2}\tau_e\sum_{i,j}(\mu^2+a_i^2+b_j^2+2\mu(a_i+b_j)+2a_ib_j\right) \\
&= \exp\left(-\frac{1}{2}(\tau_a+\tau_e n_{i:})\sum_i a_i^2 - \frac{1}{2}(\tau_b+\tau_e n_{:j})\sum_j b_j^2 - \frac{1}{2}\tau_e N\mu^2 - \tau_e\mu\left[\sum_i a_i n_{i:} + \sum_j b_j n_{:j}\right] - \tau_e\sum_{i,j} a_ib_j\right) \\
&\propto \exp\left(\boldsymbol{\beta}^T\boldsymbol{Q}\boldsymbol{\beta}\right)
\end{aligned}$$

where $n_{ij} = 1_{(i,j)\in S}$ is 1 if we have an observation for a_i and b_j and 0 else. $N = \sum_{i,j} n_{ij}$ is the total number of observations, $n_{i:} = \sum_j n_{ij}$ the number of times a_i was observed and $n_{:j} = \sum_i n_{ij}$ the number

of times b_j was observed. The entries of the precision matrix $\boldsymbol{Q}$ for $\boldsymbol{\beta}|\boldsymbol{y}$ can then be read off:

$$\begin{aligned} Q_{\mu\mu|\boldsymbol{y}} &= N\tau_e \\ Q_{\mu a_i} &= n_{i:}\tau_e \\ Q_{\mu b_j} &= n_{:j}\tau_e \\ Q_{a_i a_i} &= \tau_a + \tau_e n_{i:} \\ Q_{b_j b_j} &= \tau_b + \tau_e n_{:j} \\ Q_{a_i b_j} &= \tau_e \end{aligned} \tag{17}$$

where $i \in (1, ..., I)$ and $j \in (1, ..., J)$.

References

Reinhard Furrer and Stephan R. Sain. spam: A sparse matrix R package with emphasis on MCMC methods for Gaussian Markov random fields. *Journal of Statistical Software*, 36(10):1–25, 2010. doi: 10.18637/jss.v036.i10. URL http://www.jstatsoft.org/v36/i10/.

Alan George and Joseph W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31, 1989. URL http://www.jstor.org/stable/2030845.

John R. Gilbert. Predicting structure in sparse matrix computations. *SIAM Journal on Matrix Analysis and Applications*, 15(1):62–79, 1994. doi: 10.1137/S0895479887139455. URL https://doi.org/10.1137/S0895479887139455.

Gene H. Golub and Charles F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 2013.

Esmond G. Ng and Barry W. Peyton. Block sparse cholesky algorithms on advanced uniprocessor computers. *Mathematical Sciences Section, Oak Ridge National Laboratory*, 1991.

O Papaspiliopoulos, G O Roberts, and G Zanella. Scalable inference for crossed random effects models. *Biometrika*, 107(1):25–40, 11 2019. ISSN 0006-3444. doi: 10.1093/biomet/asz058. URL https://doi.org/10.1093/biomet/asz058.

Igor Rivin. Counting cycles and finite dimensional lp norms. *Advances in Applied Mathematics*, 29 (4):647 – 662, 2002. ISSN 0196-8858. doi: https://doi.org/10.1016/S0196-8858(02)00037-4. URL http://www.sciencedirect.com/science/article/pii/S0196885802000374.

Håvard Rue and Leonhard Held. *Gaussian Markov Random Fields*. Chapman Hall/CRC, Florida, 2004.