Penalized reduced rank regression for multidimensional survival data: new estimation approaches and simulation study.

Miltiadous, Myriana

Universiteit Leiden
The Netherlands

Leiden University Medical Center

Penalized reduced rank regression for multidimensional survival data: new estimation approaches and simulation study.

Myriana Miltiadous

Thesis advisors:
Dr. Mar Rodríguez-Girondo, LUMC
PhD candidate Marije Sluiskes, LUMC

Defended on February 10th, 2025

MASTER THESIS

STATISTICS AND DATA SCIENCE

UNIVERSITEIT LEIDEN

Contents

Abstract

This thesis investigates the use of reduced-rank regression (RRR) models in the context of survival analysis (survRRR models). RRR models have not been widely explored in this field; however, their use as a dimension reduction method to predict multiple related outcomes in multivariate statistics and machine learning has shown great potential in the context of linear models [1]. The analysis of longitudinal multimorbidity data, which implies the study of multiple time to-event outcomes (e.g. age at a certain disease onset) in a single individual, is the main motivation for the work presented in this thesis. More specifically, this research aims to identify relevant predictors for multimorbidity. Given the potentially high dimensionality of both candidate predictors and time-to-event outcomes, the focus is primarily on studying penalized survRRR models, particularly considering LASSO and group LASSO approaches. Notably, no simulation studies have investigated the empirical properties of penalized and non-penalized survRRR models.

Therefore, this thesis has two main objectives. First, an intensive simulation study is conducted to evaluate the performance of existing survRRR implementations in relevant practical settings. Second, this thesis focuses on developing a general estimation procedure for penalized survRRR models using the Alternating Direction Method of Multipliers (ADMM), which is appealing because it can effectively address complicated optimization problems by breaking them down into more manageable subproblems. The proposed algorithm is implemented in `R` for the LASSO penalty and compared with an existing implemented LASSO-penalized approach [2]. With these contributions, this work provides a general framework for fitting penalized survRRR models, using the ADMM algorithm and its implementation in `R`, with the potential to be expanded for more complicated penalties within the provided framework. Additionally, through simulation studies, this thesis offers a deeper understanding of the performance and behavior of survRRR models.

Chapter 1

Introduction

In recent years, considerable progress has been made in extending classical regression models to handle high-dimensional covariates in time-to-event data. High-dimensional data refers to datasets with a large number of predictor variables that can complicate classical regression techniques. One common approach to managing this complexity is the use of penalized regression models. In simple words, penalization strategies simplify and improve the generalizability of regression models by adding a constraint that shrinks some coefficients, discouraging the development of complex, overfitted models. The Least Absolute Shrinkage and Selection Operator, or LASSO, is one of these methods [3]. The LASSO technique shrinks some coefficients to zero, effectively selecting a simpler, easier-to-understand model with the exclusion of less important variables. Penalized models have been used for the prediction of univariate survival outcomes in many studies [4, 5].

Longitudinal studies, which involve repeated observations of the same subjects over time, have recently developed further to include multiple time-to-event outcomes for the same individual [6, 7]. For instance, within the same cohort, a study may monitor the duration between the diagnosis of cardiovascular disease and diabetes. This expansion has driven interest in identifying common factors of co-occurring diseases, which has increased the demand for regression models that can handle multiple outcomes at once.

To address this need, reduced rank regression for survival data offers a promising approach. In general, the goal of reduced-rank regression (RRR) is to model the relationships between predictors and multiple response variables while reducing the dimensionality of the problem by constraining the rank of the coefficient matrix. This is achieved by decomposing the coefficient matrix into a product of lower-dimensional matrices. The detailed procedure for this approach is outlined in the following sections of this chapter. Although this method has been extensively researched for continuous outcomes [8, 1], its application to survival analysis remains uncommon.

Fiocco et al. introduced a reduced rank proportional hazards model for competing risks

in a survival analysis setting, demonstrating its applicability in a breast cancer trial [9, 10]. Competing risks occur when participants in a study are at risk of one or more distinct events, and the occurrence of one event precludes the occurrence of another. The model proposed by Fiocco et al. (2005) can also be applied to scenarios where individuals remain at risk of experiencing other events after experiencing a particular type of event, as is often the case in studies involving multi-morbidity data.

Our work focuses on extending the Fiocco et al. model by incorporating penalized reduced-rank regression (RRR) models for multivariate survival data. The thesis has two main aims.

First, we investigate a recently proposed LASSO approach [2], which is based on the same type of algorithm as the original Fiocco et al. model. Through an extensive simulation study, we compare the performance of both the original unpenalized model and the newly proposed LASSO approach across a wide range of scenarios. Of note, no prior simulation experiments have been conducted for RRR models in the context of survival analysis.

The second aim of this thesis is to develop a general method able to handle more general penalization types. The aforementioned LASSO approach is implemented using existing `R` functions, limiting its applicability to a small subset of penalty types, namely LASSO, ridge, and elastic net. For instance, group penalization methods, such as group LASSO, cannot be easily incorporated. Group LASSO and other group-based penalties are particularly appealing in cases where predictors exhibit group-wise patterns, as is often observed in omics data. Recent work by Qian et al. has demonstrated the advantages of the group LASSO penalty in large-scale multivariate sparse linear regression using datasets like the UK Biobank [11].

To extend the range of penalized RRR models for survival data, we propose a new estimation approach based on the Alternating Direction Method of Multipliers (ADMM), a widely-used optimization method for solving complex convex optimization problems. By breaking the optimization process into smaller, more manageable subproblems, ADMM simplifies computation while allowing the inclusion of sophisticated penalty structures, such as LASSO and group LASSO. This versatility makes ADMM a suitable choice for our study.

The rest of the thesis is organized as follows: first, within chapter 1 a comprehensive presentation of the basic concepts of survival analysis and other relevant underlying methods is provided. Chapter 2 explains RRR models for survival analysis in detail, both unpenalized and penalized; how ADMM works, and how it can be used to consider RRR models for survival data with a generic penalty. Chapter 3 contains the simulation study conducted to compare the performance of unpenalized and penalized models in controlled scenarios. Chapter 4 presents experimental results from the comparison of the newly implemented algorithm in `R` using the novel approach based on ADMM, with the existing LASSO approach that is based on existing `R` functions. Chapter 5 includes a general discussion, including limitations of the conducted research and some directions for future work. The link for the GitHub repository with the `R` code developed for this thesis can be found in the Appendix.

1.1 Survival analysis

Survival analysis provides a powerful statistical framework for investigating time-to-event data, particularly in longitudinal research contexts. In this section, we present some important concepts [12].

Time-to-event data refers to data that measure the time duration until a specific event of interest occurs, such as death, disease onset, or recovery. The time-to-event is typically denoted by T ($T \geq 0$), which represents the random variable describing the time from a defined starting point t_0 (e.g., study enrollment) to the occurrence of the event of interest. Time-to-event random variables can be described using different functions, with the survival function being one of the most important. It measures the probability of surviving beyond a given time t:

$$S(t) = P(T > t). \tag{1.1}$$

Next, the *hazard function* is used to calculate the instantaneous rate of event occurrence at time t, given that the event has not occurred before t :

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \,|\, T \geq t)}{\Delta t}, \quad h(t) \geq 0. \tag{1.2}$$

The hazard function provides insights into the instantaneous risk of experiencing the event at any particular time point. For example, in a scenario of disease progression, the instantaneous risk of developing a disease at time t, given that the individual has not yet developed the disease can be represented by the hazard function. The hazard is closely associated with the survival function $S(t)$, which gives the probability that a person has not experienced the event of interest up to time t. The relationship between survival and hazard function is expressed as:

$$S(t) = \exp\left(-\int_0^t h(u)\, du\right).$$

A higher hazard, $h(t)$, corresponds to a faster decline in the survival function, $S(t)$. Another important function is the *cumulative hazard function*, which determines the accumulated hazard up to time t :

$$\Lambda(t) = \int_0^t h(s)\, ds. \tag{1.3}$$

Finally, the *distribution function* represents the probability of the event occurring by time t and is expressed as:

$$F(t) = P(T \leq t) = \int_0^t f(s)\, ds, \tag{1.4}$$

where $f(t)$ is the probability density function of T. All these functions are interconnected and can be derived from one another. For example:

$$h(t) = -\frac{d}{dt} \log(S(t)) = \frac{f(t)}{S(t)},$$

and the survival function is linked to the cumulative hazard as:

$$S(t) = \exp(-\Lambda(t)).$$

One distinctive feature of survival analysis techniques is their ability to incorporate censoring, which occurs when the exact time of an event is unknown. This can happen for several reasons: the event may not have occurred by the end of the study (*right censoring*), it may have occurred before observation began (*left censoring*), or it may only be known to fall within a specific time interval (*interval censoring*). The most popular type is *right censoring* and it is the only type considered in this thesis. To define the censoring time for right-censoring, a random variable C is often used. The observed survival time $\tilde{T}$ is the minimum of T and C, ($\tilde{T} = min(T, C)$). Given a sample of n individuals, the data for each individual is represented by (t_i, δ_i), where t_i is the time of the event or censoring, and δ_i is an indicator variable that equals 1 if the event has been observed and 0 if the data is censored.

1.1.1 The Cox proportional hazards model

One commonly used method for regression modeling in survival analysis is the Cox proportional hazards model [13]. This model is popular because it can handle censoring and allows researchers to study how covariates influence the hazard function without assuming a particular distribution for survival times. The hazard function, $h(t)$, is expressed in the Cox proportional hazards model as:

$$h(t|\mathbf{Z}) = h_0(t) \exp(\beta_1 Z_1 + \beta_2 Z_2 + \cdots + \beta_p Z_p) = h_0(t) \exp(\boldsymbol{\beta}^\top \mathbf{Z}). \tag{1.5}$$

In Equation (1.5), $h(t|\mathbf{Z})$ represents the hazard function at time t given the covariates $Z_1, \ldots, Z_p$. Regarding $h_0(t)$, it is the baseline hazard function that reflects the hazard when all covariates are zero and $\beta_1, \ldots, \beta_p$ are the regression coefficients of the outcome for the covariates $Z_1, \ldots, Z_p$.

The Cox model relies on the proportional hazards assumption, which states that the ratio of the hazard functions for any two individuals remains constant across time. In other words, the impact of a covariate on the hazard is multiplicative and remains constant throughout time. The data is used to estimate the above-mentioned baseline hazard non-parametrically. This implies that the baseline hazard $h_0(t)$ is determined directly from the observed survival times and events, as opposed to selecting a particular functional form. This makes it a semi-parametric model.

In scenarios where thousands of predictors and more than one outcome exist, the procedure of estimating the coefficients becomes extremely complicated, like models taking an unfeasible amount of time to converge or fail to converge at all. Having that in mind, researchers have developed techniques to deal with such situations.

1.2 Penalized Cox proportional hazards model

Two common challenges in applying the Cox proportional hazards model are multicollinearity, which occurs when predictors are highly correlated and makes it difficult to separate their individual effects, and overfitting, where the model captures noise rather than meaningful patterns, leading to poor generalizability to new data [14]. Overfitting is especially problematic in high-dimensional settings, where the number of predictors (p) exceeds the number of observations (n). To address these issues, penalized models are often employed, as they add a penalty term to the likelihood, helping to reduce complexity and improve model performance.

Penalization techniques have been successfully applied to Cox proportional hazards regression in the context of survival analysis [15, 16].

Next, we describe two types of penalization techniques: the LASSO and its extension, the group LASSO.

1.2.1 LASSO

LASSO was first proposed by Tibshirani in 1996 [3]. It utilizes the ℓ_1 penalty, which is defined as the sum of the absolute values of the regression coefficients multiplied by a tuning parameter, λ, that controls the size of that penalty, balancing model fit and sparsity (the bigger the λ, the bigger the penalty, the smaller the coefficients are associated with the predictors). Mathematically, the ℓ_1 penalty can be expressed as $\sum_{j=1}^{p} |\beta_j|$, where β_j are the regression coefficients and p is the number of predictors.
With this in mind, the optimization problem in general terms, that LASSO solves is:

$$\hat{\boldsymbol{\beta}}^{\text{LASSO}} = \arg\min_{\boldsymbol{\beta}} \left\{ \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{\beta})) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

In the above expression, $\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{\beta}))$ is the loss function and it represents the model's error with regards to the response variable $\boldsymbol{y}$, the predictors $\boldsymbol{x}$, and the model parameters $\boldsymbol{\beta}$. Depending on the model, this could be the residual sum of squares in ordinary least squares (OLS) regression, the negative log-likelihood in generalized linear models, or the partial likelihood in the context of the Cox proportional hazards model.

1.2.2 Group LASSO

The group LASSO [17], is an extension of the LASSO and it is more suited for situations where predictors are naturally grouped. For instance genetic markers, such as single nucleotide polymorphisms (SNPs) are naturally categorized by their chromosomal locations or related biological processes. The main idea behind this method, is to either include or completely exclude entire groups of coefficients from the model. Thus, all coefficients within a group of variables are set to

zero if that group of variables is considered irrelevant. After deciding which groups are included and which are not, the group LASSO penalizes the coefficients within the included groups.

The general group LASSO optimization problem is:

$$\hat{\boldsymbol{\beta}}^{\text{Group LASSO}} = \arg\min_{\boldsymbol{\beta}} \left\{ \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{\beta})) + \lambda \sum_{g=1}^{G} \sqrt{d_g} \|\boldsymbol{\beta}_g\|_2 \right\}$$

Here, $\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{\beta}))$ is again the general loss function. G is the number of groups, $\boldsymbol{\beta}_g$ represents the vector of coefficients within group g, d_g is the size of group g and $\|\boldsymbol{\beta}_g\|_2$ is the ℓ_2-norm (Euclidean norm) of the coefficients in group g. This guarantees that the group of coefficients as a whole can be chosen or eliminated.

1.3 Reduced rank regression models

As already mentioned, multi-outcome survival data is becoming increasingly available, and building regression models that can handle several survival outcomes simultaneously is the focus of this master thesis.

To tackle this issue, reduced-rank regression (RRR) for survival data is a suitable method. Despite their long history, RRR models have not been widely used with survival data. Originally introduced by Anderson for multivariate regression [18], this method has attracted a lot of interest due to its ability to effectively capture underlying data structures while reducing the complexity of the predictor space [19, 20, 21, 22].

In traditional regression models involving K outcomes and p predictors, the resulting $p \times K$ coefficient matrix $\mathbf{B}$ is typically of full rank, with coefficients equivalent to those obtained when separate univariate models for each outcome are fitted. RRR, however, imposes a rank (R) constraint on the $p \times K$ matrix $\mathbf{B}$, with $R \leq min(p, K)$. To achieve this, $\mathbf{B}$ is factorized into two lower-dimensional matrices: $\mathbf{B} = \mathbf{A}\boldsymbol{\Gamma}^\top$, where $\mathbf{A}$ is a $p \times R$ matrix and $\boldsymbol{\Gamma}$ is a $K \times R$ matrix. With this procedure, the model introduces some hidden variables, known as latent factors, which summarize the effect of the predictors. The $\mathbf{A}$ matrix consists of the effects of the predictors on the latent factors, while the $\boldsymbol{\Gamma}$ matrix represents the impacts of the latent factors on the outcomes.

To make it even clearer, rank (R) represents the dimension of the column space that $\mathbf{B}$ should have (i.e. number of linearly independent columns). So, if $\mathbf{A}$ and $\boldsymbol{\Gamma}$ matrices are generated with one column ($R = 1$), $\mathbf{B}$ is expected to have one linearly independent column and when they are generated with two columns ($R = 2$), $\mathbf{B}$ is expected to have two linearly independent columns. A reduction in the variables that the model needs to estimate is therefore achieved since the model only needs to estimate the smaller matrices $\mathbf{A}$ and $\boldsymbol{\Gamma}$, rather than the entire $\mathbf{B}$ matrix. As a result of the rank constraint forced on $\mathbf{B}$, the number of coefficients to be estimated decreases significantly from $p \times K$ to $R \times (p + K - R)$.

Chapter 2

Methods

2.1 Reduced rank regression for multivariate survival data (survRRR)

2.1.1 Unpenalized survRRR

As explained in chapter 1, RRR aims to reduce the rank of the $p \times K$ coefficient matrix of interest, denoted as $\mathbf{B}$ in the context of regression modeling with a multivariate outcome of dimension K and p regressors. Therefore, $\mathbf{B}$ is deconstructed into $\mathbf{A}$ and $\mathbf{\Gamma}$ matrices ($\mathbf{B} = \mathbf{A}\mathbf{\Gamma}^\top$) with respective dimensions $p \times R$ and $K \times R$. $\mathbf{B}$ is then of max rank R. To apply this idea to survival analysis, there are some important steps that Fiocco et al. took [9]. First of all, the function of the proportional hazards model takes as input, $\mathbf{Z}$, a p$\times$1 covariate vector of event k, $k = 1, ..., K$ [13]. This is given in the following Equation 2.1:

$$h_k(t|\mathbf{Z}) = h_{k0}(t) \exp(\beta_{k1} Z_1 + \beta_{k2} Z_2 + \cdots + \beta_{kp} Z_p) = h_{k0}(t) \exp(\boldsymbol{\beta}_k^\top \mathbf{Z}), \tag{2.1}$$

where $h_k(t|\mathbf{Z})$ is the hazard function for the k^{th} outcome at time t, $h_{k0}(t)$ is the baseline hazard specific to the k^{th} outcome, and $\boldsymbol{\beta}_k$ is the $p \times 1$ vector of coefficients for the covariates $Z_1, Z_2, ..., Z_p$ for the k^{th} outcome. Each outcome has a unique baseline hazard $h_{k0}(t)$ and unique covariate effects $\boldsymbol{\beta}_k$, which gives the model flexibility and is particularly helpful in situations when there are major differences in the relationships between covariates and outcomes. The specific time to event variable associated to each of the $k = 1, ..., K$ outcomes under consideration is denoted by T_k.

For estimating the parameters of $\mathbf{B}$ matrix (K vectors with p elements each denoted for each outcome k as $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, ..., \boldsymbol{\beta}_K$), the Cox model uses the *partial likelihood function* (instead of a

full likelihood function). This partial likelihood function of the Cox model is defined for the K considered outcomes in the following Equation (2.2):

$$L(\mathbf{B}) = \prod_{k=1}^{K} \prod_{i=1}^{D_k} \frac{\exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_{ki})}{\sum_{l \in R_k(t_{ki})} \exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_l)}, \tag{2.2}$$

where, t_{ki} ($i = 1, ..., D_k$ and $k = 1, ..., K$) denotes the D_k times of outcome k, $\mathbf{Z}_{ki}$ is the related regression variable (vector of covariates) for the subject who fails at t_{ki}, and $R_k(t_{ki})$ denotes the group of subjects at risk for outcome k just before time t_{ki}.

For the estimation of the covariate effects, the matrix $\mathbf{B}$ that maximizes the partial likelihood function should be calculated. The idea behind this function is for the Cox model to determine the likelihood that the event will occur based on comparing the event probability of an individual i who experiences the event at time t_i with the likelihoods of all other individuals who were still 'at risk' of experiencing the event at that time. This essentially means that, given that the event did occur at time t_i, the model indicates the probability that it will occur to individual i rather than any other individual in the risk set. Consequently, compared to the full likelihood used in other methods, the partial likelihood does not require knowing the exact event time for every subject of the study and it only depends on the sequence of the events.

The motivation for the described proposed partial likelihood, was the need for not having to model the baseline hazard function $h_0(t)$ directly. By avoiding the need to specify $h_0(t)$, this method is less restrictive than a full likelihood approach, significantly simplifying the computational process while allowing the model to efficiently handle censored data. Individuals who are censored—that is, who have not yet experienced the event by the end of the follow-up period—still provide information to the model. That is by including every person in the risk setup until the moment they are censored, which means that they are taken into account when determining the probability of other individuals' events until their last known time in the study. This eliminates the need to assume when censored people would finally experience the event and allows the model to estimate the impact of variables on the hazard rate based on the sequence of observed events.

Denoting the matrices of interest $\mathbf{A}$, $\boldsymbol{\Gamma}$ and $\mathbf{B}$ as vectors—$\mathbf{A} = [\boldsymbol{\alpha}_1][\boldsymbol{\alpha}_2]...[\boldsymbol{\alpha}_R]$, where $\boldsymbol{\alpha}_r$ for $r = 1, 2, ..., R$ are $p \times 1$ vectors, $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1][\boldsymbol{\gamma}_2]...[\boldsymbol{\gamma}_R]$, where $\boldsymbol{\gamma}_r$ for $r = 1, 2, ..., R$ are $K \times 1$ vectors, $\mathbf{B} = [\boldsymbol{\beta}_1][\boldsymbol{\beta}_2]...[\boldsymbol{\beta}_K]$, where $\boldsymbol{\beta}_k$ for $k = 1, 2, ..., K$ are $p \times 1$ vectors—, function $h_k(t)$ for the RRR model is given by:

$$h_k(t) = h_{k0}(t) \exp\left\{ \sum_{r=1}^{R} \gamma_{kr} \boldsymbol{\alpha}_r^\top \mathbf{Z} \right\} = h_{k0}(t) \exp(\boldsymbol{e}_k^\top \mathbf{B}^\top \mathbf{Z}), \tag{2.3}$$

where γ_{kr} is the k^{th} element of the $\boldsymbol{\gamma}_r$ vector and $\boldsymbol{e}_k$ is the vector indicating each outcome k (zero everywhere except for element k, which is equal to 1).

The partial likelihood of 2.2 can be rewritten using $\mathbf{A}$ and $\boldsymbol{\Gamma}$ instead of $\mathbf{B}$, since $\mathbf{B} = \mathbf{A}\boldsymbol{\Gamma}^\top$. This is given in Expression 2.4:

$$L(\mathbf{A}) = L(\mathbf{\Gamma}) = \prod_{k=1}^{K} \prod_{i=1}^{D_k} \frac{\exp(\sum_{r=1}^{R} \gamma_{kr} \boldsymbol{\alpha}_r^\top \mathbf{Z}_{ki})}{\sum_{l \in R_k(t_{ki})} \exp(\sum_{r=1}^{R} \gamma_{kr} \boldsymbol{\alpha}_r^\top \mathbf{Z}_l)}. \tag{2.4}$$

Fiocco et al. proposed an iterative procedure to simultaneously optimize the partial likelihood concerning $\mathbf{A}$ and $\mathbf{\Gamma}$, meaning that firstly $\mathbf{\Gamma}$ is considered fixed and the maximum likelihood solution for $\mathbf{A}$ is found [9]. Afterwards, the solution found for $\mathbf{A}$ is considered fixed, and the maximum likelihood solution for $\mathbf{\Gamma}$ is found. This type of algorithm is also known under the names criss-cross [23] or NIPALS [24]. Concretely, the steps are presented in Algorithm 1, with $\ell(\mathbf{A}) = logL(\mathbf{A})$ and $\ell(\mathbf{\Gamma}) = logL(\mathbf{\Gamma})$.

Algorithm 1 Estimation procedure for unpenalized survRRR model

1. Initial estimate for $\mathbf{\Gamma}$ any $K \times R$ matrix of rank R.
2. For given $\mathbf{\Gamma}$, estimate $\mathbf{A}$:
   $$h_k(t|\mathbf{Z}) = h_{k0}(t) \exp\left(\sum_{r=1}^{R} \boldsymbol{\alpha}_r^\top [\gamma_{kr} \mathbf{Z}]\right),$$
   where $\ell(\mathbf{A}) = \sum_{k=1}^{K} \ell_k(\mathbf{A})$ and $\mathbf{A} = \arg\max_{\mathbf{A}} \ell(\mathbf{A}) = \arg\min_{\mathbf{A}} -\ell(\mathbf{A})$
3. For given $\mathbf{A}$, estimate $\mathbf{\Gamma}$:
   $$h_k(t|\mathbf{Z}) = h_{k0}(t) \exp\left(\sum_{r=1}^{R} \gamma_{kr} \left[\boldsymbol{\alpha}_r^\top \mathbf{Z}\right]\right),$$
   where $\ell(\mathbf{\Gamma}) = \sum_{k=1}^{K} \ell_k(\mathbf{\Gamma})$ and $\mathbf{\Gamma} = \arg\max_{\mathbf{\Gamma}} \ell(\mathbf{\Gamma}) = \arg\min_{\mathbf{\Gamma}} -\ell(\mathbf{\Gamma})$
4. Steps 2 and 3 are repeated until the log-partial likelihoods of the following iterations diverge by no more than a chosen convergence tolerance.

The RRR model given by 2.3 has a uniquely defined matrix $\mathbf{B}$. Nevertheless, $\mathbf{B} = \mathbf{A}\mathbf{\Gamma}^\top$ is not a unique decomposition. This is due to the fact that the decomposition $\mathbf{B} = \mathbf{A}\mathbf{M}\mathbf{M}^{-1}\mathbf{\Gamma}^\top = (\mathbf{A}\mathbf{M})(\mathbf{\Gamma}\mathbf{M}^{-1})$ still represents a valid reduced rank model for any non-singular $R \times R$ rotation matrix $\mathbf{M}$. Further restrictions are needed to obtain a unique set of parameter estimates for $\mathbf{A}$ and $\mathbf{\Gamma}$ from the elements of $\mathbf{B}$.

2.1.2 Penalized survRRR

The estimation procedure described in 2.1.1 can fail in some relevant situations, resulting to the coefficient matrix $\mathbf{B}$ becoming undefined. Applying penalization to this procedure, can be a proper way of solving this problem.

Recently, an extension of Algorithm 1 to include a LASSO penalty has been proposed [2] and is given in Algorithm 2. The change made in the penalized algorithm compared to the unpenalized one is the addition of the penalty term to the objective functions. So, instead of minimizing $-\ell(\mathbf{A})$ (and $-\ell(\mathbf{\Gamma})$), the objective should include a penalty term, and hence minimize $-\ell(\mathbf{A}) + \lambda pen(\mathbf{A})$ (and $-\ell(\mathbf{\Gamma}) + \lambda pen(\mathbf{\Gamma})$). The scalar λ represents the penalty strength applied

in the objective function and pen() defines the penalty term. The function used for deciding the convergence is also different because of the penalty. This is given by:

$$-\ell(\mathbf{B}) + \lambda pen(\mathbf{A}) + \lambda pen(\mathbf{\Gamma}).$$

Algorithm 2 Estimation procedure penalized survRRR model

1: Let $k = 0$, initialize $\mathbf{\Gamma}^{(0)}$, initialize objective function $O^{(0)} = 100$.
2: **while** $k = 0$ or $|O^{(k+1)} - O^{(k)}| \geq \epsilon$ do **do**
3: Fix $\mathbf{\Gamma}^{(0)}$, solve $\mathbf{A}$:
4: Fit a penalized Cox model, stratified by outcome:

$$h_k(t|\mathbf{Z}) = h_{k0}(t) \exp\left(\sum_{r=1}^{R} \boldsymbol{\alpha}_r^\top \gamma_{kr} \mathbf{Z}\right), \tag{2.5}$$

s.t. the penalized negative partial loglikelihood is minimized, i.e.

$$\text{minimize}_{\mathbf{A}} \left\{-\ell(\mathbf{A}) + \lambda \, \text{pen}(\mathbf{A})\right\}, \tag{2.6}$$

p any penalty.
5: Fix $\mathbf{A}^{(k+1)}$, solve $\mathbf{\Gamma}^{(k+1)}$:
6: Fit a penalized Cox model, stratified by outcome:

$$h_k(t|\mathbf{Z}) = h_{k0}(t) \exp\left(\sum_{r=1}^{R} \gamma_{kr} [\boldsymbol{\alpha}_r^\top \mathbf{Z}]\right), \tag{2.7}$$

s.t. the penalized negative partial loglikelihood is minimized, i.e.

$$\text{minimize}_{\mathbf{\Gamma}} \left\{-\ell(\mathbf{\Gamma}) + \lambda pen(\mathbf{\Gamma})\right\}, \tag{2.8}$$

p any penalty.
7: Calculate objective function $O^{(k+1)} = -\ell(\mathbf{B}^{(k+1)}) + \lambda pen(\mathbf{A}^{(k+1)}) + \lambda pen(\mathbf{\Gamma}^{(k+1)})$.
8: **end while**

The structure of Algorithm 2, which involves fitting penalized models stratified by outcome, enables the use of existing software for certain types of penalties. Notably, Sluiskes et al. [2] implemented Algorithm 2 with a LASSO penalty using the widely-used `glmnet` package [25, 26] of `R` that is based on coordinate descent algorithm for optimization [27].

2.2 Alternating Direction Method of Multipliers (ADMM)

2.2.1 Motivation and backround

As mentioned earlier, the recently developed LASSO survRRR implementation based on `glmnet` cannot be easily extended to accommodate other types of penalties. To address this limitation,

we have developed a new algorithm for penalized survRRR that supports arbitrary penalties using the Alternating Direction Method of Multipliers (ADMM).

ADMM combines the advantages of the dual ascent and the method of multipliers and it is particularly beneficial for solving large-scale optimization problems, especially those that can be divided into smaller sub-problems that can be solved without the strict conditions that the dual ascent method relies on [28]. That ability of ADMM makes it particularly well-suited to our problem, as it enables separate handling of the penalty and the stratification. For more information about the development of ADMM please refer to the Appendix section 6.1.

2.2.2 ADMM algorithm

The general form of the optimization problem that ADMM solves is:

$$\min_{\boldsymbol{x},\boldsymbol{z}} f(\boldsymbol{x}) + g(\boldsymbol{z}) \tag{2.9}$$

$$\text{subject to } \mathbf{M}\boldsymbol{x} + \mathbf{D}\boldsymbol{z} - \boldsymbol{c} = 0 \tag{2.10}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{z} \in \mathbb{R}^m$, $\mathbf{M} \in \mathbb{R}^{p\times n}$, $\mathbf{D} \in \mathbb{R}^{p\times m}$, and $\boldsymbol{c} \in \mathbb{R}^p$. The functions f and g are assumed to be convex. The ADMM algorithm iteratively solves this problem with the following procedure.

It first defines the augmented Lagrangian of the specific problem:

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \frac{\rho}{2}||\mathbf{M}\mathbf{x} + \mathbf{D}\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{y}||_2^2 \tag{2.11}$$

where $\boldsymbol{y}$ is known as the scaled dual variable (essential for the algorithm) and $\rho > 0$ is known as the ADMM step size, that influences the convergence properties of the algorithm. Then the algorithm repeats the following steps until convergence is achieved:

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}} L_\rho(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{y}^k) \tag{2.12}$$

$$\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} L_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{y}^k) \tag{2.13}$$

$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \mathbf{M}\mathbf{x}^{k+1} + \mathbf{D}\boldsymbol{z}^{k+1} - \boldsymbol{c}. \tag{2.14}$$

The algorithm is considered to have converged when both primal and dual residuals are sufficiently small. Those are defined as:

$$\text{Primal Residual} : \boldsymbol{r}^{k+1} = \mathbf{M}\mathbf{x}^{k+1} + \mathbf{D}\boldsymbol{z}^{k+1} - \boldsymbol{c} \text{ and Dual Residual} : \boldsymbol{s}^{k+1} = \rho\mathbf{M}^\top\mathbf{D}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \tag{2.15}$$

Therefore, the convergence criteria are set as:

$$\|\boldsymbol{r}^{k+1}\|_2 \leq \epsilon^{\text{pri}} \text{ and } \|\boldsymbol{s}^{k+1}\|_2 \leq \epsilon^{\text{dual}} \tag{2.16}$$

where ϵ^{pri} and ϵ^{dual} are small positive tolerances (predetermined small values, close to zero).

2.3 General penalized survRRR using ADMM

Now we present the ADMM implementation to estimate penalized survRRR models, considering an arbitrary penalty. The resulting, new algorithm which can fit penalized survRRR models, is hence given in Algorithm 3.

2.3.1 Preliminary considerations

Algorithm 3, follows the steps presented in Algorithm 2, but additions are made to it, regarding how the objective functions in steps 4 and 6 of Algorithm 2 are minimized. Before explaining the resulting algorithm in detail, it is essential to introduce some mathematical expressions and explain how they are derived. First of all, to be able to construct the algorithm it is necessary to derive the (log) partial likelihoods in terms of the matrices $\mathbf{A}$ and $\mathbf{\Gamma}$, $\ell(\mathbf{A})$ and $\ell(\mathbf{\Gamma})$ respectively since the objective functions for the optimization problems that are needed to be solved are based on them. For doing so, the partial likelihood of matrix $\mathbf{B}$ (Equation (2.2)) is utilized. The procedure for deriving those expressions is shown in section 2.3.4.1. The derived partial log-likelihoods of the matrices $\mathbf{A}$ and $\mathbf{\Gamma}$ are given in equations 2.17 and 2.18 respectively. Moreover, the gradient of $\ell(\mathbf{A})$ in terms of $\boldsymbol{\alpha}$, which is a $(pR) \times 1$ vector with all the elements of the $\mathbf{A}$ matrix, and the gradient of $\ell(\mathbf{\Gamma})$ in terms of $\boldsymbol{\gamma}$, which is a $(KR) \times 1$ vector with all elements of the $\mathbf{\Gamma}$ matrix, should also be derived to solve the augmented Lagrangians presented in steps 4 and 14 of Algorithm 3. The calculations for those expressions are derived in section 2.3.4.2. So, $\ell(\mathbf{A})$ and $\ell(\boldsymbol{\alpha})$ denote the same expression, but $\ell(\mathbf{A})$ is written in terms of $\mathbf{A}$, whereas $\ell(\boldsymbol{\alpha})$ is written in terms of $\boldsymbol{\alpha}$. This also holds for $\ell(\mathbf{\Gamma})$ and $\ell(\boldsymbol{\gamma})$. Finally, the expressions for the update steps 7,8 and 17, 18 are derived and provided in section 2.3.4.3.

Algorithm 3 Estimation procedure for penalized RRR model for survival data using ADMM

Let $q = 0$, initialize $\mathbf{\Gamma}^{(0)}$, initialize objective function $0^{(0)} = 100$

while $q = 0$ **or** $|\mathcal{O}^{(q+1)} - \mathcal{O}^{(q)}| \geq \epsilon_1$ **do** ▷ $O^{(q)} = -logL(\mathbf{B}^{(q)}) + \lambda\|\mathbf{A}^{(q)}\|_L + \lambda\|\mathbf{\Gamma}^{(q)}\|_N$

Fix $\mathbf{\Gamma}^{(0)}$, solve $\mathbf{A}$: ▷ $\mathbf{\Gamma}^{(0)}$ any $K \times R$ matrix with rank R

For given $\mathbf{\Gamma}$ with elements $\boldsymbol{\gamma}_{kr}$, fit a penalized Cox model, stratified by outcome:

$$h_k(t) = h_{k0}(t)\exp\left(\sum_{r=1}^{R}\boldsymbol{\alpha}_r^\top \mathbf{W}_{kr}\right), \text{where } \mathbf{W}_{kr} = \gamma_{kr}^\top \mathbf{Z}$$

s.t. the penalized negative partial log-likelihood is minimized, i.e.:

$$\min_{\mathbf{A}}\left\{-\log L(\mathbf{A}) + \lambda\sum_{r=1}^{R}\|\boldsymbol{\alpha}_r\|_L\right\}, \quad L \text{ any norm}$$

Convert to ADMM: minimize $\min_{\boldsymbol{\alpha},\mathbf{d}}\left\{-logL(\boldsymbol{\alpha}) + \lambda\sum_{r=1}^{R}\|\mathbf{d}_r\|_L\right\}$ subject to $\boldsymbol{\alpha} - \mathbf{d} = 0$

$$\mathcal{L}(\boldsymbol{\alpha}, \mathbf{d}, \mathbf{u}) = -logL(\boldsymbol{\alpha}) + \lambda\sum_{r=1}^{R}\|\mathbf{d}_r\|_L + \frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d} + \mathbf{u}\|_2^2 = -\ell(\boldsymbol{\alpha}) + \lambda\sum_{r=1}^{R}\|\mathbf{d}_r\|_L + \frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d} + \mathbf{u}\|_2^2$$

Initialize $\boldsymbol{\alpha}^{(0)}$, $\mathbf{d}^{(0)}, \mathbf{u}^{(0)}$ to 0-filled vectors, $r_1^{(0)}, s_1^{(0)} = 100, n = 0$

while $\| \boldsymbol{r}_1^{(n+1)}\|_2 \geq \epsilon_2$ or $\| \boldsymbol{s}_1^{(n+1)}\|_2 \geq \epsilon_3$ **do**

$\boldsymbol{\alpha}^{(n+1)} = argmin_{\boldsymbol{\alpha}}(\mathcal{L}(\boldsymbol{\alpha}, \mathbf{d}^{(n)}, \mathbf{u}^{(n)}))$

$\mathbf{d}^{(n+1)} = argmin_{\mathbf{d}}(\mathcal{L}(\boldsymbol{\alpha}^{(n+1)}, \mathbf{d}, \mathbf{u}^{(n)}))$

$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \boldsymbol{\alpha}^{(n+1)} - \mathbf{d}^{(n+1)}$

Calculate residuals $\boldsymbol{r}_1^{(n+1)} = \boldsymbol{\alpha}^{(n+1)} - \mathbf{d}^{(n+1)}$, $\boldsymbol{s}_1^{(n+1)} = \rho(\mathbf{d}^{(n+1)} - \mathbf{d}^{(n)})$

$n = n + 1$

end while

Fix $\mathbf{A}^{(n+1)}$, solve $\mathbf{\Gamma}^{(n+1)}$:

For given $\mathbf{A}$ with columns $\boldsymbol{\alpha}_r$, $r = 1, \ldots, R$, fit a penalized Cox model, for each outcome, $k = 1, \ldots, K$ separately, with $V_r = \boldsymbol{\alpha}_r^\top \mathbf{Z}$ as covariates, i.e.:

$$h_k(t) = h_{k0}(t)\exp\left(\sum_{r=1}^{R}\gamma_{kr}V_r\right)$$

s.t. the penalized negative partial loglikelihood is minimized, i.e.:

$$\min_{\boldsymbol{\gamma}}\left\{-logL(\mathbf{\Gamma}) + \lambda\sum_{r=1}^{R}\|\boldsymbol{\gamma}_r\|_N\right\}, \; N \text{ any norm}$$

Convert to ADMM: minimize $\min_{\boldsymbol{\gamma},\mathbf{t}}\left\{-logL(\boldsymbol{\gamma}) + \lambda\sum_{r=1}^{R}\|\mathbf{t}_r\|_N\right\}$ subject to $\boldsymbol{\gamma} - \mathbf{t} = 0$

$$\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{z}, \mathbf{u}) = -logL(\boldsymbol{\gamma}) + \lambda\sum_{r=1}^{R}\|\mathbf{t}_r\|_N + \frac{\rho}{2}\|\boldsymbol{\gamma} - \mathbf{t} + \mathbf{u}\|_2^2 = -\ell(\boldsymbol{\gamma}) + \lambda\sum_{r=1}^{R}\|\mathbf{t}_r\|_N + \frac{\rho}{2}\|\boldsymbol{\gamma} - \mathbf{t} + \mathbf{u}\|_2^2$$

Initialize $\boldsymbol{\gamma}^{(0)}$, $\mathbf{t}^{(0)}, \mathbf{u}^{(0)}$ to 0-filled vectors, $r_2^{(0)}, s_2^{(0)} = 100, n = 0$

while $\| \boldsymbol{r}_2^{(n+1)}\|_2 \geq \epsilon_4$ or $\| \boldsymbol{s}_2^{(n+1)}\|_2 \geq \epsilon_5$ **do**

$\boldsymbol{\gamma}^{(n+1)} = argmin_{\boldsymbol{\gamma}}(\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{z}^{(n)}, \mathbf{u}^{(n)}))$

$\mathbf{t}^{(n+1)} = argmin_{\mathbf{t}}(\mathcal{L}(\boldsymbol{\gamma}^{(n+1)}, \boldsymbol{z}, \mathbf{u}^{(n)}))$

$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \boldsymbol{\gamma}^{(n+1)} - \mathbf{t}^{(n+1)}$

Calculate residuals $\boldsymbol{r}_2^{(n+1)} = \boldsymbol{\gamma}^{(n+1)} - \mathbf{t}^{(n+1)}$, $\boldsymbol{s}_2^{(n+1)} = \rho(\mathbf{t}^{(n+1)} - \mathbf{t}^{(n)})$

$n = n + 1$

end while

Calculate objective function $|\mathcal{O}^{(q+1)} - \mathcal{O}^{(q)}|$ ▷ $\mathbf{B}^{(q+1)} = \mathbf{A}^{(q+1)}\mathbf{\Gamma}^{T(q+1)}$

$q = q + 1$

end while

ρ: positive scalar parameter, which controls the penalty for the primal and dual variables' disagreement across the ADMM iterations.

λ : regularization parameter in the penalization problem, which controls the trade-off between the solution's simplicity and the model's accuracy in representing the data.

$\boldsymbol{\alpha}$: $(pR) \times 1$ vector of all the elements of matrix $\mathbf{A}$

$\boldsymbol{\gamma}$: $(KR) \times 1$ vector of all the elements of matrix $\boldsymbol{\Gamma}$

$\|\cdot\|_L$, $\|\cdot\|_N$: The L-norm and N-norm respectively, which depend on the desired penalization (f-norm of a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is defined as: $\|\boldsymbol{x}\|_f = \left(\sum_{i=1}^{n} |x_i|^f\right)^{\frac{1}{f}}$).

2.3.2 Description of the ADMM algorithm

Algoritm 3 is now explained in detail. It begins by initializing a variable q, which denotes the number of the total outer iterations for the algorithm to converge. Within the outer while loop there are two inner while loops to incorporate the desired penalty using ADMM. The logic behind those while loops is to apply the specific penalty independently on both the matrices $\mathbf{A}$ and $\boldsymbol{\Gamma}$ like in Algorithm 2, but with the use of ADMM. The formulation of the problem in ADMM notation can be explicitly seen in steps 4 and 14 of the Algorithm 2.

2.3.2.1 Estimating A

Concerning the first inner while loop (lines 3-12), it is implemented by considering matrix $\boldsymbol{\Gamma}$ ($K \times R$ matrix) as a fixed, randomly initialized matrix and iterates until convergence for the matrix $\mathbf{A}$ ($p \times R$ matrix). Convergence is checked by calculating the 2-norm of first and second order residuals $\boldsymbol{r}_1$ and $\boldsymbol{s}_1$. Here ϵ^{pri} is represented by ϵ_2 and ϵ^{dual} by ϵ_3. Within this while loop the following steps are considered: first for the given $\boldsymbol{\Gamma}$ matrix, a penalized Cox model stratified by outcome is fitted, subject to the minimization of the penalized negative partial likelihood. In step 4 of the algorithm, $\lambda \sum_{r=1}^{R} \|\boldsymbol{\alpha}_r\|_L$ represents the chosen penalty incorporated in the model ($\boldsymbol{\alpha}_r$ represents the r^{th} column of matrix $\mathbf{A}$ and is a $p \times 1$ vector). The letter L is used to denote an arbitrary norm here. For example, if $L = 1$, then the LASSO penalty is applied. The auxiliary variable $\mathbf{d}$, which is a $pR \times 1$ vector, is also introduced to be used in the constraint for the ADMM method (In fact $\mathbf{d}$ is derived after considering a $p \times R$ matrix $\mathbf{D}$, which is can be represented as $[\mathbf{d}_1, .., \mathbf{d}_R]$, where $\mathbf{d}_r$ is a $p \times 1$ vector that symbolizes the r^{th} column of that $p \times R$ matrix $\mathbf{D}$). So, the optimization problem that should be solved with ADMM is the following: $\min_{\boldsymbol{\alpha},\mathbf{d}} \left\{ -logL(\boldsymbol{\alpha}) + \lambda \sum_{r=1}^{R} \|\mathbf{d}_r\|_L \right\}$ subject to $\boldsymbol{\alpha} - \mathbf{d} = 0$. The augmented Lagrangian, $\mathcal{L}(\boldsymbol{\alpha}, \mathbf{d}, \mathbf{u})$, is defined as explained in 2.11. In the augmented Lagrangian, $\mathbf{u}$ is the scaled dual variable and ρ is the step size. Both those variables are essential for the ADMM method. Then the $\boldsymbol{\alpha}$, $\mathbf{d}$ and $\mathbf{u}$ vectors are initialized to zeros and the first and second order

residuals, r_1 and s_1, to 100. Also, n, which counts the iterations until convergence, is initialized to 0. After that $\boldsymbol{\alpha}$, $\mathbf{d}$ and $\mathbf{u}$ are updated iteratively, using the gradients of the denoted expressions in lines 7-10, until the residuals converge. Those steps are the conversion of the ADMM update steps (described by expressions 2.12-2.14) to our specific optimization problem of adding LASSO penalty to the survRRR model.

2.3.2.2 Estimating Γ

The same procedure follows within the second inner while loop (lines 12-20), whose goal is for the matrix $\boldsymbol{\Gamma}$ to converge. Here, the auxiliary variable is denoted by $\mathbf{t}$ (instead of $\mathbf{d}$ used in the first inner while loop). The three variables ($\boldsymbol{\gamma}$, $\mathbf{t}$ and $\mathbf{u}$) are again updated according to the values that minimize the augmented Lagrangian of the problem with fixed $\mathbf{A}$, whereas in the first while loop $\boldsymbol{\gamma}$ is considered fixed. The minimization problem that is solved here is $\min_{\boldsymbol{\gamma},\mathbf{t}} \left\{ -logL(\boldsymbol{\Gamma}) + \lambda \sum_{r=1}^{R} \|\mathbf{t}_r\|_N \right\}$ subject to $\boldsymbol{\gamma} - \mathbf{t} = 0$. For this problem, N-norm is used to represent the penalty, since it is not necessary that the same penalty, as the one applied to $\mathbf{A}$ matrix, should be applied to the $\boldsymbol{\Gamma}$ matrix. So, if $N = 1$, the LASSO penalty is considered. Combining that with $L = 1$ in the first loop, the algorithm should yield similar results with the `glmnet` based algorithm that applies the LASSO penalty [2].

2.3.2.3 Overall convergence

After both $\mathbf{A}$ and $\boldsymbol{\Gamma}$ matrix have converged ($\|\boldsymbol{r}_2\|_2$ and $\|\boldsymbol{s}_2\|_2$ become smaller than ϵ^{pri}, denoted as ϵ_4, and ϵ^{dual}, denoted as ϵ_5, respectively), the while loop ends and a check is performed in line 23 for the convergence of the objective function: $O^{(q+1)} = -logL(\mathbf{B}^{(q+1)}) + \lambda\|\mathbf{A}^{(q+1)}\|_L + \lambda\|\boldsymbol{\Gamma}^{(q+1)}\|_N$. Objective function concerns matrix $\mathbf{B}$ ($p \times K$ matrix) with the desired penalty separately applied to each of the matrices $\mathbf{A}$ and $\boldsymbol{\Gamma}$, $\lambda\|\mathbf{A}^{(q+1)}\|_L$ and $\lambda\|\boldsymbol{\Gamma}^{(q+1)}\|_N$. For the calculation of $\mathbf{B}$, the derived $\mathbf{A}$ and $\boldsymbol{\Gamma}$ matrices of the two inner loops are used ($\mathbf{B} = \mathbf{A}\boldsymbol{\Gamma}^\top$), which have R columns with p and K rows, respectively. If convergence of the objective function is not achieved then the two inner while loops are repeated until it converge.

2.3.3 Group LASSO survRRR using ADMM

One of the aims of this thesis is to lay the foundation for an alternative way to include penalties in the RRR model for survival data since there is no available package that can implement the group LASSO penalty. For incorporating the group LASSO penalty in the above-described Algorithm 3), some additions should be made. These are listed below:

1. The objective function for $\mathbf{B}$ matrix (comment in **step 2** of the Algorithm 3) should become:
   $O^{(q+1)} = -logL(\mathbf{B}^{(q+1)}) + \lambda \sum_{r=1}^{R} \sum_{g=1}^{G} \sqrt{p_g} \|\boldsymbol{\alpha}_{r_g}^{(q+1)}\|_2 + \lambda pen(\boldsymbol{\Gamma})$.

Here $g = 1, ..., G$ denotes the different groups that the p coefficients are divided into to form the group LASSO penalty. Also, p_g is the number of coefficients belonging to group g and finally $\boldsymbol{\alpha}_{r_g}$ represent the elements of $\boldsymbol{\alpha}_r$ column of $\mathbf{A}$ matrix that belong to group g. Term $pen(\boldsymbol{\Gamma})$ corresponds to the penalty that is chosen to be incorporated into the $\boldsymbol{\Gamma}$ matrix (e.g. $\|\boldsymbol{\Gamma}^{(q+1)}\|_N$), which can also be ignored if desired.

2. The first and second expressions subjected to minimization, in step **step 4** should become respectively:
   a)$\min_{\mathbf{A}} \left\{ -logL(\mathbf{A}) + \lambda \sum_{r=1}^{R} \sum_{g=1}^{G} \sqrt{p_g} \|\boldsymbol{\alpha}_{r_g}\|_2 \right\}$
   b)$\min_{\boldsymbol{\alpha},\mathbf{d}} \left\{ -logL(\boldsymbol{\alpha}) + \lambda \sum_{r=1}^{R} \sum_{g=1}^{G} \sqrt{p_g} \|\mathbf{d}_{r_g}\|_2 \right\}$
   Here $g = 1, ..., G$ again denotes the different groups that the p coefficients are divided into to form the group LASSO penalty. Also, p_g is again the number of coefficients belonging to group g. Regarding $\boldsymbol{\alpha}_{r_g}$, it represents the elements of $\boldsymbol{\alpha}_r$ that belongs to group g and regarding $\mathbf{d}_{r_g}$, it represents the elements of $\mathbf{d}_r$ that belongs to group g. Therefore, the new Lagrangian becomes:
   $\mathcal{L}(\boldsymbol{\alpha}, \mathbf{d}, \mathbf{u}) = -\ell(\boldsymbol{\alpha}) + \lambda \sum_{r=1}^{R} \sum_{g=1}^{G} \sqrt{p_g} \|\mathbf{d}_{r_g}\|_2 + \mathbf{u}^\top (\boldsymbol{\alpha} - \mathbf{d}) + \frac{\rho}{2} \|\boldsymbol{\alpha} - \mathbf{d}\|_2^2$
   and **step 8** becomes:
   $\mathbf{d}^{(n+1)} = argmin_{\mathbf{d}}(\mathcal{L}(\boldsymbol{\alpha}^{(n+1)}, \mathbf{d}, \mathbf{u}^{(n)}))$, using this new augmented Langrngian.

3. Concerning the second inner loop, where a penalization is added to $\boldsymbol{\Gamma}$ matrix, it is decided not make any changes for incorporating any group-penalization, since we wanted the group LASSO penalty only to be applied to the p predictors, and not to the outcomes.

   Thus, the second while loop can either be ignored when group LASSO is applied to the model or taken into consideration for incorporating e.g. a LASSO penalty. For the second option, the steps are the same as the ones described for the general penalty and thus all the relevant expressions remain the same with those.

The gradient expressions for steps 7 and 8 in the group LASSO case are derived in section 2.3.4.3. It is important to note that for step 7, the expression is the same for both the group LASSO and the general case, as it is independent of the penalty. However, for step 8, the expressions differ between the general case and the group LASSO penalty, with the general case expression also presented in this thesis below the group LASSO ones. The expressions for steps 17 and 18 (associated with matrix $\boldsymbol{\Gamma}$-second inner loop-) also provided in section 2.3.4.3, are only relevant to the general case of penalization since, as previously mentioned, we decided that the group LASSO only concerns the p predictors.

2.3.4 Mathematical derivations

This section explains how crucial expressions of Algorithm 3 are calculated. For more context revisit section 2.3.1 with preliminary considerations.

2.3.4.1 Derivation of partial log likelihoods

The given expression in 2.2 is the starting point.

Firstly, the natural logarithm of $L(\mathbf{B})$ is taken:

$$\log L(\mathbf{B}) = \log \left(\prod_{k=1}^{K} \prod_{i=1}^{D_k} \frac{\exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_{ki})}{\sum_{l \in R_k(t_{ki})} \exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_l)} \right).$$

Then, using the property of logarithms, $\log(ab) = \log a + \log b$, it is derived that:

$$\log L(\mathbf{B}) = \sum_{k=1}^{K} \sum_{i=1}^{D_k} \log \left(\frac{\exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_{ki})}{\sum_{l \in R_k(t_{ki})} \exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_l)} \right).$$

Now, applying the logarithm property $\log\left(\frac{a}{b}\right) = \log a - \log b$ it is obtained:

$$\log L(\mathbf{B}) = \sum_{k=1}^{K} \sum_{i=1}^{D_k} \left(\log\left(\exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_{ki})\right) - \log \left(\sum_{l \in R_k(t_{ki})} \exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_l) \right) \right).$$

Using $\log\left(\exp(x)\right) = x$:

$$\log L(\mathbf{B}) = \sum_{k=1}^{K} \sum_{i=1}^{D_k} \left(\boldsymbol{\beta}_k^\top \mathbf{Z}_{ki} - \log \left(\sum_{l \in R_k(t_{ki})} \exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_l) \right) \right).$$

Therefore, the log of the given expression $L(\mathbf{B})$ is:

$$\ell(\mathbf{B}) = \log L(\mathbf{B}) = \sum_{k=1}^{K} \sum_{i=1}^{D_k} \left(\boldsymbol{\beta}_k^\top \mathbf{Z}_{ki} - \log \left(\sum_{l \in R_k(t_{ki})} \exp(\boldsymbol{\beta}_k^\top \mathbf{Z}_l) \right) \right).$$

So since, $\boldsymbol{\beta}_k^\top \mathbf{Z} = \sum_{r=1}^{R} \boldsymbol{\alpha}_r^\top \mathbf{W}_{kr} = \sum_{r=1}^{R} \boldsymbol{\gamma}_{kr} V_r$, where $W_{kr} = \gamma_{kr} Z$ and $V_r = \boldsymbol{\alpha}_r^\top \mathbf{Z}$, the desired expressions are:

$$\ell(\boldsymbol{\alpha}) = \ell(\mathbf{A}) = \log L(\mathbf{A}) = \sum_{k=1}^{K} \sum_{i=1}^{D_k} \left(\sum_{r=1}^{R} \boldsymbol{\alpha}_r^\top \mathbf{W}_{kr}(ki) - \log \left(\sum_{l \in R_k(t_{ki})} \exp(\sum_{r=1}^{R} \boldsymbol{\alpha}_r^\top \mathbf{W}_{kr}(I)) \right) \right), \tag{2.17}$$

$$\ell(\boldsymbol{\gamma}) = \ell(\boldsymbol{\Gamma}) = \log L(\boldsymbol{\Gamma}) = \sum_{k=1}^{K} \sum_{i=1}^{D_k} \left(\sum_{r=1}^{R} \gamma_{kr} V_r(ki) - \log \left(\sum_{l \in R_k(t_{ki})} \exp(\sum_{r=1}^{R} \gamma_{kr} V_r(I)) \right) \right), \tag{2.18}$$

where K represents the number of outcomes, and D_k denotes the number of failure times of outcome k. The parameter R indicates the number of ranks, while $R_k(t_{ki})$ is the set of subjects at risk for failure of outcome k just prior to time t_{ki}.

In this context, γ_{kr} refers to the element in the k^{th} row and r^{th} column of the matrix $\mathbf{\Gamma}$, and $V_r(ki)$ is defined as $\boldsymbol{\alpha}_r^\top \mathbf{Z}_{ki}$, where $\boldsymbol{\alpha}_r$ represents the r^{th} column of matrix $\mathbf{A}$ and $\mathbf{Z}_{ki}$ is the regression variable for the individual failing at time t_{ki}. Therefore, $\mathbf{Z}_I$ is the corresponding regression variable for the individuals who belong in the I^{th} group. The terms $\mathbf{W}_{kr}(ki)$ and $\mathbf{W}_{kr}(I)$ are equal to $\boldsymbol{\gamma}_{kr}\mathbf{Z}_{ki}$ and $\boldsymbol{\gamma}_{kr}\mathbf{Z}_I$, respectively, and $V_r(I)$ is computed as $\boldsymbol{\alpha}_r^\top \mathbf{Z}_I$.

2.3.4.2 Derivation of gradients: $\nabla_{\boldsymbol{\alpha}}\ell(\mathbf{A})$ and $\nabla_{\boldsymbol{\gamma}}\ell(\mathbf{\Gamma})$

Derivation of $\nabla_{\boldsymbol{\alpha}}\ell(\mathbf{A})$: We start from Equation (2.17). Since $\mathbf{A}$ is a $pR \times 1$ vector and $\ell(\mathbf{A})$ is a scalar function, $\nabla_{\boldsymbol{\alpha}}\ell(\mathbf{A})$ has the form $\nabla_{\boldsymbol{\alpha}}\ell(\mathbf{A}) = [\frac{d\ell(\mathbf{A})}{da_{11}}, ..., \frac{d\ell(\mathbf{A})}{da_{pR}}]$. In order to be able to derive this expression $\ell(\mathbf{A})$ had to be converted to be in terms of the a_{jr}, where a_{jr} is the element of j^{th} row and r^{th} column of the matrix $\mathbf{A}$.

Given: $\mathbf{A} = [\boldsymbol{\alpha}_1][\boldsymbol{\alpha}_2]...[\boldsymbol{\alpha}_R]$, where $\boldsymbol{\alpha}_r$ are $p \times 1$ vectors, with this structure:

$$\boldsymbol{\alpha}_r = \begin{pmatrix} a_{1r} \\ a_{2r} \\ \vdots \\ a_{pr} \end{pmatrix}$$

The elements of $\mathbf{A}$ matrix can be expressed as a vector in this form:

$$\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \\ \boldsymbol{\alpha}_R \end{pmatrix} = \begin{pmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \\ \vdots \\ \alpha_{1R} \\ \alpha_{2R} \\ \vdots \\ \alpha_{pR} \end{pmatrix}$$

Given also the expression in Equation (2.17):

$$\ell(\boldsymbol{\alpha}) = \ell(\mathbf{A}) = \sum_{k=1}^{K}\sum_{i=1}^{D_k}\left(\sum_{r=1}^{R}\boldsymbol{\alpha}_r^\top \mathbf{W}_{kr}(ki) - \log\left(\sum_{l\in R_k(t_{ki})}\exp\left(\sum_{r=1}^{R}\boldsymbol{\alpha}_r^\top \mathbf{W}_{kr}(I)\right)\right)\right).$$

The inner product $\boldsymbol{\alpha}_r^\top \mathbf{W}_{kr}(ki)$ is first rewritten in terms of the elements α_{jr}:

$$\boldsymbol{\alpha}_r^\top \mathbf{W}_{kr}(ki) = \begin{pmatrix} a_{1r} & a_{2r} & \cdots & a_{pr} \end{pmatrix} \begin{pmatrix} W_{kr1}(ki) \\ W_{kr2}(ki) \\ \vdots \\ W_{krp}(ki) \end{pmatrix}$$

This simplifies to:

$$\boldsymbol{\alpha}_r^\top \mathbf{W}_{kr}(ki) = \sum_{j=1}^{p} a_{jr} W_{krj}(ki)$$

So, in terms of a_{jr}, the expression becomes:

$$\ell(\mathbf{A}) = \sum_{k=1}^{K} \sum_{i=1}^{D_k} \left(\sum_{r=1}^{R} \sum_{j=1}^{p} a_{jr} W_{krj}(ki) - \log \left(\sum_{l \in R_k(t_{ki})} \exp \left(\sum_{r=1}^{R} \sum_{j=1}^{p} a_{jr} W_{krj}(I) \right) \right) \right)$$

Now, the gradient can be derived:

$$\nabla_{\boldsymbol{\alpha}} \ell(\mathbf{A}) = [\frac{\partial \ell(\mathbf{A})}{\partial a_{11}}, ..., \frac{\partial \ell(\mathbf{A})}{\partial a_{pR}}]$$

To have the general expression of $\nabla_{\boldsymbol{\alpha}} \ell(\mathbf{A})$, the derivative of $\ell(\mathbf{A})$ is derived with respect to the element $\boldsymbol{\alpha}_{nm}$ $(\frac{\partial \ell(\mathbf{A})}{\partial a_{nm}})$ and then by substituting n and m accordingly, $\nabla_{\boldsymbol{\alpha}} \ell(\mathbf{A})$ can be derived. Next, the steps of applying the chain rule are given. $\ell(\mathbf{A})$ consists of two main parts: a linear term and a logarithmic term. They are considered separately.

1. Linear term:$\sum_{r=1}^{R} \sum_{j=1}^{p} a_{jr} W_{krj}(ki)$
2. Logarithmic term:$\log \left(\sum_{l \in R_k(t_{ki})} \exp \left(\sum_{r=1}^{R} \sum_{j=1}^{p} a_{jr} W_{krj}(I) \right) \right)$

Step 1: Differentiate the linear term with respect to a_{nm} The linear term is:

$$\sum_{r=1}^{R} \sum_{j=1}^{p} a_{jr} W_{krj}(ki)$$

The partial derivative of this term with respect to a_{nm} is straightforward. It is be zero unless $j = n$ and $r = m$:

$$\frac{\partial}{\partial a_{nm}} \left(\sum_{r=1}^{R} \sum_{j=1}^{p} a_{jr} W_{krj}(ki) \right) = W_{kmn}(ki)$$

Step 2: Differentiate the logarithmic term with respect to a_{nm}. The logarithmic term is:

$$\log \left(\sum_{l \in R_k(t_{ki})} \exp \left(\sum_{r=1}^{R} \sum_{j=1}^{p} a_{jr} W_{krj}(I) \right) \right)$$

Let:

$$g_k(i) = \sum_{l \in R_k(t_{ki})} \exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right)$$

So, the logarithmic term becomes $\log(g_k(i))$. The derivative of $\log(g_k(i))$ is:

$$\frac{\partial}{\partial a_{nm}} \log(g_k(i)) = \frac{1}{g_k(i)} \frac{\partial g_k(i)}{\partial a_{nm}}$$

Now, $\frac{\partial g_k(i)}{\partial a_{nm}}$ is needed:

$$g_k(i) = \sum_{l \in R_k(t_{ki})} \exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right)$$

The derivative of $g_k(i)$ with respect to a_{nm} is:

$$\frac{\partial g_k(i)}{\partial a_{nm}} = \sum_{l \in R_k(t_{ki})} \exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right) \frac{\partial}{\partial a_{nm}} \left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right)$$

Since $\frac{\partial}{\partial a_{nm}} \left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right)$ is $W_{kmn}(I)$, it means:

$$\frac{\partial g_k(i)}{\partial a_{nm}} = \sum_{l \in R_k(t_{ki})} \exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right) W_{kmn}(I)$$

Putting it all together:

$$\frac{\partial}{\partial a_{nm}} \log(g_k(i)) = \frac{1}{g_k(i)} \sum_{l \in R_k(t_{ki})} \exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right) W_{kmn}(I)$$

Step 3: Combining the derivatives from the linear and logarithmic terms, the partial derivative of $\ell(\mathbf{A})$ with respect to a_{nm} is:

$$\frac{\partial \ell(\mathbf{A})}{\partial a_{nm}} = \sum_{k=1}^{K}\sum_{i=1}^{D_k} \left(W_{kmn}(ki) - \frac{1}{g_k(i)} \sum_{l \in R_k(t_{ki})} \exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right) W_{kmn}(I)\right)$$

This can be simplified to:

$$\frac{\partial \ell(\mathbf{A})}{\partial a_{nm}} = \sum_{k=1}^{K}\sum_{i=1}^{D_k} \left(W_{kmn}(ki) - \sum_{l \in R_k(t_{ki})} \frac{\exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right)}{g_k(i)} W_{kmn}(I)\right)$$

After writing out $g_k(i)$ this expression is obtained:

$$\frac{\partial \ell(\mathbf{A})}{\partial a_{nm}} = \sum_{k=1}^{K}\sum_{i=1}^{D_k} \left(W_{kmn}(ki) - \sum_{l \in R_k(t_{ki})} \frac{\exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right)}{\sum_{l \in R_k(t_{ki})} \exp\left(\sum_{r=1}^{R}\sum_{j=1}^{p} a_{jr} W_{krj}(I)\right)} W_{kmn}(I)\right) \tag{2.19}$$

By using Equation (2.19), $\nabla_{\boldsymbol{\alpha}} \ell(\mathbf{A}) = [\frac{\partial \ell(\mathbf{A})}{\partial a_{11}}, ..., \frac{\partial \ell(\mathbf{A})}{\partial a_{pR}}]$ can easily be calculated.

Derivation of $\nabla_{\boldsymbol{\gamma}}\ell(\boldsymbol{\Gamma})$: Here the starting point is Equation (2.18). Since $\boldsymbol{\Gamma}$ is a $KR\times 1$ vector and $\ell(\boldsymbol{\Gamma})$ is a scalar function, $\nabla_{\gamma}\ell(\boldsymbol{\Gamma})$ has the form $\nabla_{\boldsymbol{\gamma}}\ell(\boldsymbol{\Gamma}) = [\frac{d\ell(\boldsymbol{\Gamma})}{d\gamma_{11}}, ..., \frac{d\ell(\boldsymbol{\Gamma})}{d\gamma_{KR}}]$. In order to be able to derive this expression, the general derivative of $\ell(\boldsymbol{\Gamma})$ concerning $\boldsymbol{\gamma}_{nm}$ should be derived. Given: $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1][\boldsymbol{\gamma}_2]...[\boldsymbol{\gamma}_R]$, where $\boldsymbol{\gamma}_r$ are $K\times 1$ vectors, and

$$\boldsymbol{\gamma}_r = \begin{pmatrix} \gamma_{1r} \\ \gamma_{2r} \\ \vdots \\ \gamma_{Kr} \end{pmatrix}$$

Putting all elements of $\boldsymbol{\Gamma}$ matrix in one vector, gives:

$$\boldsymbol{\gamma} = \begin{pmatrix} \boldsymbol{\gamma}_1 \\ \boldsymbol{\gamma}_2 \\ \vdots \\ \boldsymbol{\gamma}_R \end{pmatrix} = \begin{pmatrix} \gamma_{11} \\ \gamma_{21} \\ \gamma_{31} \\ \vdots \\ \gamma_{1R} \\ \gamma_{2R} \\ \vdots \\ \gamma_{KR} \end{pmatrix}$$

So, given Equation (2.18):

$$\ell(\boldsymbol{\gamma}) = \ell(\boldsymbol{\Gamma}) = \sum_{k=1}^{K}\sum_{i=1}^{D_k}\left(\sum_{r=1}^{R}\gamma_{kr}V_r(ki) - \log\left(\sum_{I\in R_k(t_{ki})}\exp\left(\sum_{r=1}^{R}\gamma_{kr}V_r(I)\right)\right)\right)$$

The first aim is to derive the derivative of $\ell(\boldsymbol{\Gamma})$ with respect to γ_{nm}.

1. **First Term Derivative**:

$$\frac{\partial}{\partial\gamma_{nm}}\sum_{k=1}^{K}\sum_{i=1}^{D_k}\left(\sum_{r=1}^{R}\gamma_{kr}V_r(ki)\right)$$

The derivative of the first term for γ_{nm} is not equal to zero if $r = m$ and $k = n$. To indicate this, the Kronecker delta is used, δ, which is defined as:

$$\delta_{ab} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}$$

Using the Kronecker delta, it is obtained that:

$$\frac{\partial}{\partial\gamma_{nm}}\sum_{k=1}^{K}\sum_{i=1}^{D_k}\sum_{r=1}^{R}\gamma_{kr}V_r(ki) = \sum_{k=1}^{K}\sum_{i=1}^{D_k}\sum_{r=1}^{R}\delta_{kn}\delta_{rm}V_r(ki) = \sum_{i=1}^{D_n}V_m(ni)$$

2. **Second Term Derivative**:

$$\frac{\partial}{\partial \gamma_{nm}} \left(-\sum_{k=1}^{K} \sum_{i=1}^{D_k} \log \left(\sum_{I \in R_k(t_{ki})} \exp \left(\sum_{r=1}^{R} \gamma_{kr} V_r(I) \right) \right) \right)$$

Defining the function:

$$f(\boldsymbol{\gamma}) = -\sum_{k=1}^{K} \sum_{i=1}^{D_k} \log \left(\sum_{I \in R_k(t_{ki})} \exp \left(\sum_{r=1}^{R} \gamma_{kr} V_r(I) \right) \right)$$

The aim is to find the partial derivative of $f(\gamma)$ with respect to γ_{nm}.
It is now defined:

$$S_{ki} = \sum_{I \in R_k(t_{ki})} \exp \left(\sum_{r=1}^{R} \gamma_{kr} V_r(I) \right)$$

Thus, the function can be rewritten as:

$$f(\boldsymbol{\gamma}) = -\sum_{k=1}^{K} \sum_{i=1}^{D_k} \log(S_{ki})$$

Using the chain rule, the partial derivative of $f(\gamma)$ with respect to γ_{nm} is:

$$\frac{\partial f(\boldsymbol{\gamma})}{\partial \gamma_{nm}} = -\sum_{k=1}^{K} \sum_{i=1}^{D_k} \frac{1}{S_{ki}} \frac{\partial S_{ki}}{\partial \gamma_{nm}}$$

The derivative of each term in the sum of S_{ki} with respect to γ_{nm} is non-zero only when $k = n$.
So, for $k = n$, the term is:

$$\frac{\partial}{\partial \gamma_{nm}} \exp \left(\sum_{r=1}^{R} \gamma_{kr} V_r(I) \right) = \exp \left(\sum_{r=1}^{R} \gamma_{kr} V_r(I) \right) V_m(I)$$

Therefore,

$$\frac{\partial S_{ki}}{\partial \gamma_{nm}} = \sum_{I \in R_k(t_{ki})} \exp \left(\sum_{r=1}^{R} \gamma_{kr} V_r(I) \right) V_m(I) \delta_{kn}$$

Substituting $\frac{\partial S_{ki}}{\partial \gamma_{nm}}$ into the expression for $\frac{\partial f(\boldsymbol{\gamma})}{\partial \gamma_{nm}}$, it is derived:

$$\frac{\partial f(\boldsymbol{\gamma})}{\partial \gamma_{nm}} = -\sum_{k=1}^{K} \sum_{i=1}^{D_k} \frac{1}{S_{ki}} \sum_{I \in R_k(t_{ki})} \exp \left(\sum_{r=1}^{R} \gamma_{kr} V_r(I) \right) V_m(I) \delta_{kn}$$

The Kronecker delta δ_{kn} simplifies the sum over k by eliminating all terms for which $k \neq n$. Thus, it is derived:

$$\frac{\partial f(\boldsymbol{\gamma})}{\partial \gamma_{nm}} = -\sum_{i=1}^{D_n} \frac{1}{S_{ni}} \sum_{I \in R_n(t_{ni})} \exp\left(\sum_{r=1}^{R} \gamma_{nr} V_r(I)\right) V_m(I)$$

Recognizing that $S_{ni} = \sum_{I \in R_n(t_{ni})} \exp\left(\sum_{r=1}^{R} \gamma_{nr} V_r(I)\right)$, the fraction can be simplified to:

$$\frac{\partial f(\boldsymbol{\gamma})}{\partial \gamma_{nm}} = -\sum_{i=1}^{D_n} \sum_{I \in R_n(t_{ni})} \frac{\exp\left(\sum_{r=1}^{R} \gamma_{nr} V_r(I)\right)}{\sum_{I' \in R_n(t_{ni})} \exp\left(\sum_{r=1}^{R} \gamma_{nr} V_r(I')\right)} V_m(I)$$

Putting them all together, the partial derivative of the given function with respect to γ_{nm} is:

$$\frac{\partial f(\boldsymbol{\gamma})}{\partial \gamma_{nm}} = -\sum_{i=1}^{D_n} \sum_{I \in R_n(t_{ni})} \frac{\exp\left(\sum_{r=1}^{R} \gamma_{nr} V_r(I)\right)}{\sum_{I' \in R_n(t_{ni})} \exp\left(\sum_{r=1}^{R} \gamma_{nr} V_r(I')\right)} V_m(I)$$

Finally, the derivative of $\ell(\boldsymbol{\Gamma})$ with respect to γ_{nm} (combining the derivatives of the first and second terms) is:

$$\frac{\partial \ell(\boldsymbol{\Gamma})}{\partial \gamma_{nm}} = \sum_{i=1}^{D_n} \left[V_m(ni) - \sum_{I \in R_n(t_{ni})} \frac{\exp\left(\sum_{r=1}^{R} \gamma_{nr} V_r(I)\right)}{\sum_{I' \in R_n(t_{ni})} \exp\left(\sum_{r=1}^{R} \gamma_{nr} V_r(I')\right)} V_m(I) \right] \tag{2.20}$$

Thus, by using Equation (2.20) , the expression for $\nabla_\gamma \ell(\boldsymbol{\Gamma}) = [\frac{\partial \ell(\boldsymbol{\Gamma})}{\partial \gamma_{11}}, ..., \frac{\partial \ell(\boldsymbol{\Gamma})}{\partial a_{KR}}]$ can easily be computed.

2.3.4.3 Derivation of expressions for the update of ADMM method

Step 7: Expression: $\boldsymbol{\alpha}^{(n+1)} = argmin_a(-\ell(\boldsymbol{\alpha}) + \frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}\|_2^2)$, which means that the gradient of $-\ell(\boldsymbol{\alpha}) + \frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}\|_2^2$ in terms of $\mathbf{A}$ should be set to zero and solved with respect to $\boldsymbol{\alpha}$.
Since the gradient of $\ell(\mathbf{A})$ has already been derived in the previous section, the focus is now on the expression:

$$\frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}\|_2^2 \tag{2.21}$$

where $\boldsymbol{\alpha}$, $\mathbf{d}^{(n)}$, and $\mathbf{u}^{(n)}$ are vectors, and ρ is a scalar.
To find the derivative of the expression

$$f(\boldsymbol{\alpha}) = \frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}\|_2^2,$$

with respect to $\boldsymbol{\alpha}$, recall that the Euclidean norm can be expressed as a dot product:

$$\|\boldsymbol{x}\|_2^2 = \boldsymbol{x}^\top \boldsymbol{x}$$

Thus, the original expression can be rewritten as:

$$f(\boldsymbol{\alpha}) = \frac{\rho}{2}(\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)})^\top(\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)})$$

Now this expression can be further expanded:

$$f(\boldsymbol{\alpha}) = \frac{\rho}{2}\left[\boldsymbol{\alpha}^\top\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top(\mathbf{d}^{(n)} - \mathbf{u}^{(n)}) + (\mathbf{d}^{(n)} - \mathbf{u}^{(n)})^\top(\mathbf{d}^{(n)} - \mathbf{u}^{(n)})\right]$$

Since the term $(\mathbf{d}^{(n)} - \mathbf{u}^{(n)})^\top(\mathbf{d}^{(n)} - \mathbf{u}^{(n)})$ is a constant with respect to $\boldsymbol{\alpha}$, it vanishes when taking the gradient. For computing the gradient of $f(\boldsymbol{\alpha})$, the following rules are applied: $\nabla_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}^\top\boldsymbol{\alpha}) = 2\boldsymbol{\alpha}$ and $\nabla_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}^\top\mathbf{b})$=$\mathbf{b}$. Thus,

$$\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) = \rho(\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)})$$

$\boldsymbol{\alpha}^{(n+1)} = argmin_{\boldsymbol{\alpha}}(-\ell(\boldsymbol{\alpha}) + \frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}\|_2^2)$
$\iff -\nabla_{\boldsymbol{\alpha}}\ell(\boldsymbol{\alpha}) + \nabla_{\boldsymbol{\alpha}}\left(\frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}\|_2^2\right) = 0$
$\iff -\nabla_{\boldsymbol{\alpha}}\ell(\boldsymbol{\alpha}) + \nabla_{\boldsymbol{\alpha}}\left(\frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}\|_2^2\right) = 0$
$\iff -[\frac{\partial\ell(\boldsymbol{\alpha})}{\partial a_{11}}, ..., \frac{\partial\ell(\boldsymbol{\alpha})}{\partial a_{pR}}] + \rho(\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}) = 0$, where $\frac{\partial\ell(\boldsymbol{\alpha})}{\partial a_{nm}}$ can be calculated from $\frac{\partial\ell(\boldsymbol{\alpha})}{\partial a_{nm}}$, given in expression 2.19, explained in the previous section. The expression $-\nabla_{\boldsymbol{\alpha}}\ell(\boldsymbol{\alpha}) + \nabla_{\boldsymbol{\alpha}}\left(\frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d}^{(n)} + \mathbf{u}^{(n)}\|_2^2\right) = 0$ should in fact be solved with respect to $\mathbf{A}$ and this would be the update step for $\boldsymbol{\alpha}^{(n+1)}$. But, there is no closed form for that. So, for deriving the update step for $\boldsymbol{\alpha}$, the `optim` function of the `stats` (4.3.3) package in `R` is used [29]. This function computes parameter estimates by minimizing the objective function.

Step 8: Expression: $\mathbf{d}^{(n+1)} = argmin_{\mathbf{d}}(\lambda\sum_{r=1}^{R}\sum_{g=1}^{G}\sqrt{p_g}\|\mathbf{d}_{r_g}\|_2 + \frac{\rho}{2}\|\boldsymbol{\alpha}^{(n+1)} - \mathbf{d} + \mathbf{u}^{(n)}\|_2^2)$
If Ω_1 and Ω_2 are defined as:

$$\Omega_1 = \lambda\sum_{r=1}^{R}\sum_{g=1}^{G}\sqrt{p_g}\|\mathbf{d}_{r_g}\|_2, \quad \Omega_2 \quad = \frac{\rho}{2}\|\boldsymbol{\alpha}^{(n+1)} - \mathbf{d} + \mathbf{u}^{(n)}\|_2^2, \tag{2.22}$$

it is derived that, $\nabla_{\mathbf{d}}(\lambda\sum_{r=1}^{R}\sum_{g=1}^{G}\sqrt{p_g}\|\mathbf{d}_{r_g}\|_2 + \frac{\rho}{2}\|\boldsymbol{\alpha} - \mathbf{d} + \mathbf{u}\|_2) = \nabla_{\mathbf{d}}\Omega_1 + \nabla_{\mathbf{d}}\Omega_2$.
The first one to be computed is $\nabla_{\mathbf{d}}\Omega_1$. Since, Ω_1 is a scalar and d is a vector, then :

$$\nabla_{\mathbf{d}}\Omega_1 = [\frac{\partial\Omega_1}{\partial d_{11}}, ..., \frac{\partial\Omega_1}{\partial d_{pR}}]. \tag{2.23}$$

Therefore, for Expression 2.23 to be fully obtained, the general expression of $\frac{\partial\Omega_1}{\partial d_{nm}}$ is desired. Setting $r = m$ in the Expression of Ω_1 in 2.22, the following expression is acquired:

$$\frac{\partial\Omega_1}{\partial d_{nm}} = \lambda\frac{\partial}{\partial d_{nm}}\sum_{g=1}^{G}\sqrt{p_g}\|\mathbf{d}_{m_g}\|_2. \tag{2.24}$$

Now, in order to continue it is crucial d to know in which group g, the element d_{nm} belongs to. If it is defined to belong to group g_1 then :

$$\mathbf{d}_{m_{g1}} = \begin{pmatrix} \vdots \\ d_{nm} \\ \vdots \end{pmatrix}$$

So, $\|\mathbf{d}_{m_{g_1}}\|_2 = \sqrt{... + |d_{nm}|^2 + ...}$ and $\frac{\partial(\|\mathbf{d}_{m_{g_1}}\|_2)}{\partial d_{nm}} = \frac{\partial(\sqrt{...+|d_{nm}|^2+...})}{\partial d_{nm}} = \frac{|d_{nm}|}{\sqrt{...+|d_{nm}|^2+...}}$
Thus for d_{nm} in group g_1:

$$\frac{\partial \Omega_1}{\partial d_{nm}} = \lambda \frac{\partial}{\partial d_{nm}} \sum_{g=1}^{G} \sqrt{p_g} \|\mathbf{d}_{m_g}\|_2 = \lambda \frac{\partial}{\partial d_{nm}} \sqrt{p_{g1}} \|\mathbf{d}_{m_{g1}}\|_2 = \lambda \sqrt{p_{g1}} \frac{|d_{nm}|}{\sqrt{... + |d_{nm}|^2 + ...}} \tag{2.25}$$

For Ω_2 the procedure is equivalent with the one described for expression 2.21. So,

$$\nabla_{\mathbf{d}} \Omega_2 = \nabla_{\mathbf{d}} \left(\frac{\rho}{2} \|\boldsymbol{\alpha}^{(n+1)} - \mathbf{d} + \mathbf{u}^{(n)}\|_2^2 \right) = -\rho(\boldsymbol{\alpha}^{(n+1)} - \mathbf{d} + \mathbf{u}^{(n)}) \tag{2.26}$$

So, the expression $[\frac{\partial \Omega_1}{\partial d_{11}}, ..., \frac{\partial \Omega_1}{\partial d_{pR}}] - \rho(\boldsymbol{\alpha}^{(n+1)} - \mathbf{d} + \mathbf{u}^{(n)}) = 0$ should be solved with respect to d and that would be the update step for d^{n+1} , where $\frac{\partial \Omega_1}{\partial d_{nm}}$ is given in expression 2.29. For that again the *optim* function can be used.

General case: The only part that is different for the general case of the algorithm in this step is Ω_1, which is:

$$\Omega_1 = \lambda \sum_{r=1}^{R} \|\mathbf{d}_r\|_L. \tag{2.27}$$

The expression has again the following form: $\nabla_{\mathbf{d}} \Omega_1 = [\frac{\partial \Omega_1}{\partial d_{11}}, ..., \frac{\partial \Omega_1}{\partial d_{pR}}]$.
And it is again desired to derive the general expression: $\frac{\partial \Omega_1}{\partial d_{nm}}$
This is for r=m: $\frac{\partial \Omega_1}{\partial d_{nm}} = \lambda \frac{\partial}{\partial d_{nm}} \|\mathbf{d}_m\|_L$.
Since,

$$\mathbf{d}_m = \begin{pmatrix} d_{1m} \\ \vdots \\ d_{nm} \\ \vdots \\ d_{pm} \end{pmatrix}$$

It is obtained that:

$$\|\mathbf{d}_m\|_L = \sum_{i=1}^{p} |d_{im}|^L)^{\frac{1}{L}} \tag{2.28}$$

and

$$\frac{\partial(\|\mathbf{d}_m\|_L)}{\partial d_{nm}} = \frac{\partial\left(\left(\sum_{i=1}^{p}|d_{im}|^L\right)^{\frac{1}{L}}\right)}{\partial d_{nm}} = \frac{1}{L}\left(\sum_{i=1}^{p}|d_{im}|^L\right)^{\frac{1}{L}-1}\frac{\partial}{\partial d_{nm}}|d_{nm}|^L$$
$$= \left(\sum_{i=1}^{p}|d_{im}|^L\right)^{\frac{1}{L}-1}|d_{nm}|^{L-1}\cdot\frac{\partial|d_{nm}|}{\partial d_{nm}}$$

The derivative of the absolute value function $|d_{nm}|$ with respect to d_{nm} is:

$$\frac{\partial|d_{nm}|}{\partial d_{nm}} = \begin{cases} 1 & \text{if } d_{nm} > 0 \\ -1 & \text{if } d_{nm} < 0 \\ 0 & \text{if } d_{nm} = 0 \end{cases}$$

This can be written as:

$$\frac{\partial|d_{nm}|}{\partial d_{nm}} = \operatorname{sgn}(d_{nm})$$

Putting it all together:

$$\frac{\partial(\|\mathbf{d}_m\|_L)}{\partial d_{nm}} = \left(\sum_{i=1}^{p}|d_{im}|^L\right)^{\frac{1}{L}-1}|d_{nm}|^{L-1}\operatorname{sgn}(d_{nm})$$

Thus,

$$\frac{\partial\Omega_1}{\partial d_{nm}} = \frac{\partial}{\partial d_{nm}}\lambda\sum_{r=1}^{R}\|\mathbf{d}_r\|_L = \lambda\frac{\partial}{\partial d_{nm}}\|\mathbf{d}_m\|_L = \lambda\left(\sum_{i=1}^{p}|d_{im}|^L\right)^{\frac{1}{L}-1}|d_{nm}|^{L-1}\operatorname{sgn}(d_{nm}) \quad (2.29)$$

Step 17: Expression: $\boldsymbol{\gamma}^{(n+1)} = argmin_{\boldsymbol{\gamma}}(-\ell(\boldsymbol{\gamma}) + \frac{\rho}{2}\|\boldsymbol{\gamma} - \mathbf{t}^{(n)} + \mathbf{u}^{(n)}\|_2^2)$
Deriving the gradient of the expression $\frac{\rho}{2}\|\boldsymbol{\gamma} - \mathbf{t}^{(n)} + \mathbf{u}^{(n)}\|_2^2$ with respect to $\boldsymbol{\gamma}$ is straightforward since it is equivalent to the expression in step 7. Thus, following the same procedure it is acquired that:

$$\nabla_{\boldsymbol{\gamma}}\left(\frac{\rho}{2}\|\boldsymbol{\gamma} - \mathbf{t}^{(n)} + \mathbf{u}^{(n)}\|_2^2\right) = \rho(\boldsymbol{\gamma} - \mathbf{t}^{(n)} + \mathbf{u}^{(n)}) \quad (2.30)$$

So, since $\nabla_{\boldsymbol{\gamma}}\ell(\boldsymbol{\gamma}) = [\frac{d\ell(\boldsymbol{\gamma})}{da_{11}}, ..., \frac{d\ell(\boldsymbol{\gamma})}{da_{KR}}]$, it is derived that:
$\boldsymbol{\gamma}^{(n+1)} = argmin_{\boldsymbol{\gamma}}(-\ell(\boldsymbol{\gamma}) + \frac{\rho}{2}\|\boldsymbol{\gamma} - \mathbf{t}^{(n)} + \mathbf{u}^{(n)}\|_2^2)$
$\iff -\nabla_{\boldsymbol{\gamma}}\ell(\boldsymbol{\gamma}) + \nabla_{\boldsymbol{\gamma}}\left(\frac{\rho}{2}\|\boldsymbol{\gamma} - \mathbf{t}^{(n)} + \mathbf{u}^{(n)}\|_2^2\right) = 0$
$\iff -\nabla_{\boldsymbol{\gamma}}\ell(\boldsymbol{\gamma}) + \nabla_{\boldsymbol{\gamma}}\left(\frac{\rho}{2}\|\boldsymbol{\gamma} - \mathbf{t}^{(n)} + \mathbf{u}^{(n)}\|_2^2\right) = 0$
$\iff -[\frac{d\ell(\boldsymbol{\gamma})}{d\gamma_{11}}, ..., \frac{d\ell(\boldsymbol{\gamma})}{d\gamma_{KR}}] + \rho(\boldsymbol{\gamma} - \mathbf{t}^{(n)} + \mathbf{u}^{(n)}) = 0$, where $\frac{\partial\ell(\boldsymbol{\gamma})}{\partial\gamma_{nm}}$ is given in expression 2.20, explained in the previous section. Solving that with respect to $\boldsymbol{\gamma}$ gives the update step for $\boldsymbol{\gamma}^{(n+1)}$, but since again no closed form for it can be acquired, the *optim* package is used.

Step 18: Expression: $\mathbf{t}^{(n+1)} = argmin_{\mathbf{t}}(\lambda \sum_{r=1}^{R} \|\mathbf{t}_r\|_N + \frac{\rho}{2}\|\boldsymbol{\gamma}^{(n+1)} - \mathbf{t} + \mathbf{u}^{(n)}\|_2^2)$
The procedure to derive the gradient of $(\lambda \sum_{r=1}^{R} \|t_r\|_N + \frac{\rho}{2}\|\boldsymbol{\gamma}^{(n+1)} - \mathbf{t} + \mathbf{u}^{(n)}\|_2^2)$ is equivalent to the one described for step 8.
So if it is defined that:

$$\Omega_3 = \lambda \sum_{r=1}^{R} \|\mathbf{t}_r\|_N, \quad \Omega 4 = \frac{\rho}{2}\|\boldsymbol{\gamma}^{(n+1)} - \mathbf{t} + \mathbf{u}^{(n)}\|_2^2, \tag{2.31}$$

Then the gradient of Ω_3 with respect to t is:

$$\nabla_{\mathbf{t}}\Omega_3 = [\frac{\partial \Omega_3}{\partial t_{11}}, ..., \frac{\partial \Omega_3}{\partial t_{KR}}] \tag{2.32}$$

It is again needed to derive the general expression: $\frac{\partial \Omega_3}{\partial t_{nm}}$
This is a non-zero value only when setting $r = m$ in the Expression for Ω_3 in 2.31. So,

$$\frac{\partial \Omega_3}{\partial t_{nm}} = \lambda \frac{\partial}{\partial t_{nm}} \|\mathbf{t}_m\|_N. \tag{2.33}$$

Now, in order to continue, $\mathbf{t}_m$ needs to be defined, as a $K \times 1$ vector.

$$\mathbf{t}_m = \begin{pmatrix} t_{1m} \\ \vdots \\ t_{nm} \\ \vdots \\ t_{Km} \end{pmatrix}$$

So, $\|\mathbf{t}_m\|_N = \sum_{k=1}^{K} |t_{km}|^N)^{\frac{1}{N}}$ and
$\frac{\partial(\|\mathbf{t}_m\|_N)}{\partial t_{nm}} = \frac{\partial\left(\left(\sum_{k=1}^{K} |t_{km}|^N\right)^{\frac{1}{N}}\right)}{\partial t_{nm}} = \frac{1}{N}\left(\sum_{k=1}^{K} |t_{km}|^N\right)^{\frac{1}{N}-1} \frac{\partial}{\partial t_{nm}}|t_{nm}|^N$
$= \left(\sum_{k=1}^{K} |t_{km}|^N\right)^{\frac{1}{N}-1} |t_{nm}|^{N-1} \cdot \frac{\partial |t_{nm}|}{\partial t_{nm}}$
The derivative of the absolute value function $|t_{nm}|$ with respect to t_{nm} is:

$$\frac{\partial |t_{nm}|}{\partial t_{nm}} = \begin{cases} 1 & \text{if } t_{nm} > 0 \\ -1 & \text{if } t_{nm} < 0 \\ 0 & \text{if } t_{nm} = 0 \end{cases}$$

This can be compactly written as:

$$\frac{\partial |t_{nm}|}{\partial t_{nm}} = \text{sgn}(t_{nm})$$

Putting it all together:

$$\frac{\partial(\|\mathbf{t}_m\|_N)}{\partial t_{nm}} = \left(\sum_{k=1}^{K} |t_{km}|^N\right)^{\frac{1}{N}-1} |t_{nm}|^{N-1}\,\mathrm{sgn}(t_{nm})$$

Thus,

$$\frac{\partial \Omega_3}{\partial t_{nm}} = \frac{\partial}{\partial t_{nm}} \lambda \sum_{r=1}^{R} \|\mathbf{t}_r\|_N = \lambda \frac{\partial}{\partial t_{nm}} \|\mathbf{t}_m\|_N = \lambda \left(\sum_{k=1}^{K} |t_{km}|^N\right)^{\frac{1}{N}-1} |t_{nm}|^{N-1}\,\mathrm{sgn}(t_{nm}) \tag{2.34}$$

For $\Omega 4$ the procedure is again equivalent with the one described for expression 2.21. So,

$$\nabla_{\mathbf{t}} \Omega 4 = \nabla_{\mathbf{t}} \left(\frac{\rho}{2} \|\boldsymbol{\gamma}^{(n+1)} - \mathbf{t} + \mathbf{u}^{(n)}\|_2^2\right) = -\rho(\boldsymbol{\gamma}^{(n+1)} - \mathbf{t} + \mathbf{u}^{(n)}) \tag{2.35}$$

So, $[\frac{\partial \Omega_3}{\partial t_{11}}, ..., \frac{\partial \Omega_3}{\partial t_{KR}}] - \rho(\boldsymbol{\gamma}^{(n+1)} - \mathbf{t} + \mathbf{u}^{(n)}) = 0$, where $\frac{\partial \Omega_3}{\partial t_{nm}}$ is given in expression 2.34. Solving that in terms of $\mathbf{t}$ gives update step for $\mathbf{t}^{(n+1)}$ and for that `optim` package can once again be used.

Chapter 3

Simulation study

3.1 Simulation study setup

In this section we present a simulation study. The description of its set-up as well as its results are provided. The simulation study is described using the ADEMP framework [30]. The aims, data-generating mechanisms, methods, estimands, and performance measures are described below. Two different simulation studies are conducted: one using the unpenalized survRRR model and one using the LASSO-penalized survRRR model. They are described separately.

3.1.1 Aims

3.1.1.1 Simulation study 1

The aim of simulation study 1 is to analyze how well unpenalized survRRR models perform in low-dimensional settings, in particular estimating the elements of the $p \times K$ matrix $\mathbf{B}$ and the elements of the $n \times K$ matrix $\mathbf{ZB}$ (linear predictor). Moreover, since the data are generated based on the reduced rank assumption, we also investigate whether the simulated number of ranks aligns with the optimal number of estimated ranks.

3.1.1.2 Simulation study 2

For simulation study 2, the aim is to examine the performance of the LASSO-penalized survRRR model, considering more complex scenarios with a higher number of predictors than in simulation 1, and including sparsity and correlation among the predictors in the simulated data. The correct estimation of regression coefficients and the associated linear predictors, together with the correct

estimation of the generated number of ranks, is the aim. Moreover, in this scenario, we investigate if penalization leads to better results than unpenalized models.

3.1.2 Data-generating mechanisms

3.1.2.1 Simulation study 1

Combinations of the following parameters are taken into consideration in order to assess the unpenalized RRR model's performance for survival data: sample sizes (n): 500, 1000, 5000; number of covariates (p): 5, 10; number of outcomes (K): 4, 8; and ranks for simulated matrices $\mathbf{A}$ and $\mathbf{\Gamma}$ (R): 1, 2. This results in $3 \times 2 \times 2 \times 2 = 24$ parameter combinations. For each combination, 200 datasets ($S = 200$) are simulated. The following model is used to generate data:

$$h_k(t|\mathbf{Z}) = h_{k0}(t) \exp(\boldsymbol{\beta}_k^\top \mathbf{Z}), \tag{3.1}$$

where $h_k(t|\mathbf{Z})$ is the hazard function for the k$^{\text{th}}$ outcome at time t, $h_{k0}(t)$ represents the baseline hazard for the k^{th} outcome, and vector $\boldsymbol{\beta}_k$ is the k^{th} column of the coefficient matrix $\mathbf{B}$. For simplicity, the baseline hazard for all the outcomes is set to $h_{k0}(t) = 1$. The way that matrix $\mathbf{B}$, matrix $\mathbf{Z}$ and survival times t are simulated is presented below. The simulation process involves two main steps:

1. The first step involves generating the $\mathbf{B}$ matrix (true coefficient matrix), which is fixed across all 200 datasets for each parameter combination. This matrix is generated by multiplying $\mathbf{A}$ and $\mathbf{\Gamma}^\top$ matrices ($\mathbf{B} = \mathbf{A}\mathbf{\Gamma}^\top$). For simulation 1, the $(p \times R)$ matrix $\mathbf{A}$ and the $(K \times R)$ matrix $\mathbf{\Gamma}$ are constructed with i.i.d. elements drawn from a uniform, U[0,1], distribution. This way of simulating the true $\mathbf{B}$ matrix is referred to as 'case 1' in this thesis.

2. The second step involves generating the covariate matrix $\mathbf{Z}$, the survival times t_{ij}, and the censoring times, c_{ij}, independently for each of the S Monte Carlo trials of each combination of parameters. The $n \times p$ covariate matrix $\mathbf{Z}$ is generated with i.i.d. elements drawn from a standard normal distribution, $N(0, 1)$ and its rows, $\mathbf{Z}_i$, $i = 1, ..., p$, are uncorrelated, since they are generated independently ($\mathbf{Z}_i \sim N(0, 1)^n$). Regarding the survival times per outcome, t_{ij}, where $i = 1, \ldots, n$ indexes subjects and $j = 1, \ldots, K$ indexes outcomes, are stored in an $n \times K$ matrix $\mathbf{T}$. Each element of $\mathbf{T}$ is generated from an exponential distribution with rate parameter $e^{\eta_{ij}}$ ($t_{ij} \sim \text{Exp}(e^{\eta_{ij}})$), where $\boldsymbol{\eta} = \mathbf{ZB}$ is the linear predictor. Finally, the vector of length n with the censoring times for each subject, $\mathbf{C}$, is generated from a uniform distribution ($\mathbf{C} \sim U(0, 1)^n$). This resulted to a proportion of censored events between 54% and 63%.

3.1.2.2 Simulation study 2

In simulation 2 data is generated using the same model 3.1 as in simulation 1, but the generation of matrices $\mathbf{Z}$ and $\mathbf{B}$ is more complex. The number of predictors (p) is fixed to be 300, the number of outcomes (K) is fixed to be 8 and the value of corelation between the covariates (cor) is chosen to be 0.7 instead of 0, used in study 1. The sample sizes (n) are limited to $n = 1000$ and $n = 5000$ (the $n = 500$ is excluded from simulation study 2 due to instability issues caused by the higher dimensionality of the datasets), while the rank parameter (R) for matrices $\mathbf{A}$ and $\mathbf{\Gamma}$ remains at either 1 or 2.

Two additional generation methods (cases) for matrices $\mathbf{A}$ and $\mathbf{\Gamma}$ are included in this simulation 2, apart from 'case 1', used in simulation 1. The most interesting of them forces $\mathbf{A}$, $\mathbf{\Gamma}$ and $\mathbf{B}$ matrices to have block structures and is referred to as 'case 2'. This generation approach is interesting because it challenges the model to determine which predictors are meaningful by introducing structured sparsity, in which some variables are set to zero for particular outcomes. The model must reflect this structure in the $\mathbf{A}$ and $\mathbf{\Gamma}$ matrices while discarding irrelevant predictors, since it is estimating these matrices that have fewer columns than the number of outcomes. More details about that way of generating $\mathbf{A}$ and $\mathbf{\Gamma}$ are given in the next paragraph. The second additional way is named 'case 3'. For that case, $\mathbf{A}$ and $\mathbf{\Gamma}$ matrices follow the random generation process of 'case 1', but with added sparsity - 10% of elements in each matrix (randomly selected elements set to zero). This case is added in order to include a case in between 'case 1', which has totally random numbers, and 'case 2', which has a very specific block structure with the inclusion of zeros. The added 'case 3' is studied for the same parameters (n, p, K, R) as the other cases, but with a different correlation of 0.1 and hence, considered separately from those.

So in simulation study 2, there are in total $2 \times 2 \times 2 = 8$ parameter combinations regarding 'case 1' and 'case 2' and $2 \times 2 = 4$ parameter combinations regarding 'case 3'. For each parameter combination, 100 ($S = 100$) Monte Carlo datasets are generated in this study. The two steps to generate the data presented in simulation 1 remain similar except for some minor modifications. In step 1, not all $\mathbf{B}$ matrices are generated using 'case 1', and also all the values of the simulated (under any case) $\mathbf{B}$ matrix are divided with 100. This adjustment is made based on the observation that the non-zero correlation among predictors ($\mathbf{Z}$) of this study can result in large values in either $\mathbf{B}$ or $\boldsymbol{\eta}$ leading to event times that are either excessively large (and thus censored) or close to zero (providing little information). Dividing the regression coefficients by 100 ensures that the resulting event times are not overly skewed. In simulation 1, where there are fewer predictors and no correlation between them, this is less of an issue. The effects tend to average out in such situations (for example, a large positive Z_1 and β_1 may be balanced by another large negative Z_n). However, most $\mathbf{Z}$-values align when predictors are correlated, increasing their effect instead of averaging out, requiring the adjustment in the $\mathbf{B}$ matrix. Regarding step 2, the change made is that covariate matrix $\mathbf{Z}$ is not generated using the same procedure, since in simulation 2 correlation is added. Below, we detail steps 1 and 2 for the generation of matrices $\mathbf{B}$ and $\mathbf{Z}$, respectively, in simulation 2.

1. In simulation 2, we generate matrix $\mathbf{B}$ according to three cases: 'case 1' and 'case 3' as previously explained, and 'case 2' which is detailed here. In this new 'case 2,' matrices $\mathbf{A}$ and $\mathbf{\Gamma}$ have a block structure. For simulation under this case, p and K should be even numbers. To be more specific regarding the meaning of block structure, for $R = 2$ both $\mathbf{A}$ and $\mathbf{\Gamma}$ matrices are generated such that the first column consists of i.i.d. values drawn from U[0,1] for rows 1, ..., $\frac{p}{2}$ and zeros for rows $\frac{p}{2} + 1$, ..., p, while the second column consists of zeros for rows 1, ..., $\frac{p}{2}$ and i.i.d. values drawn from U[0,1] for rows $\frac{p}{2} + 1$, ..., p. Regarding the structure for when $R = 1$, it is more simple, with rows 1, ... , $\frac{p}{2}$ of both $\mathbf{A}$ and $\mathbf{\Gamma}$ matrices filled with zeros and with rows $\frac{p}{2} + 1$, ... , p filled with i.i.d. values drawn from U[0,1]. Finally, for both values of R, the coefficient matrix $\mathbf{B}$ is obtained by multiplying $\mathbf{A}$ with the transpose of $\mathbf{\Gamma}$, which results in $\mathbf{B}$ having a block structure as well. The structures of the three matrices, when simulated under $R = 1$ and when simulated under $R = 2$ are given in Figure 1 and in Figure 2 respectively. There, rand represents the i.i.d. values drawn from U[0,1], column-wise.

$$\begin{pmatrix} 0_{11} \\ 0_{21} \\ \vdots \\ 0_{\frac{p}{2}1} \\ \text{rand}_{(\frac{p}{2}+1)1} \\ \vdots \\ \text{rand}_{(p-1)1} \\ \text{rand}_{p1} \end{pmatrix} \quad \begin{pmatrix} 0_{11} \\ 0_{21} \\ \vdots \\ 0_{\frac{K}{2}1} \\ \text{rand}_{(\frac{K}{2}+1)1} \\ \vdots \\ \text{rand}_{(K-1)1} \\ \text{rand}_{K1} \end{pmatrix} \quad \begin{pmatrix} 0_{11} & \cdots & 0_{1\frac{K}{2}} & 0_{1(\frac{K}{2}+1)} & \cdots & 0_{1K} & \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{\frac{p}{2}1} & \cdots & 0_{\frac{p}{2}\frac{K}{2}} & 0_{\frac{p}{2}(\frac{K}{2}+1)} & \cdots & 0_{\frac{p}{2}K} & \\ 0_{(\frac{p}{2}+1)1} & \cdots & 0_{(\frac{p}{2}+1)\frac{K}{2}} & \text{B}_{(\frac{p}{2}+1)(\frac{K}{2}+1)} & \cdots & \text{B}_{(\frac{p}{2}+1)K} & \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{p1} & \cdots & 0_{p\frac{K}{2}} & \text{B}_{p(\frac{K}{2}+1)} & \cdots & \text{B}_{pK} & \end{pmatrix}$$

Figure 1: This figure represents the case, for which $\mathbf{A}$ and $\mathbf{\Gamma}$ matrices are simulated under rank 1 (R=1), using block structure way of simulation ('case 2'). The leftmost matrix is the $\mathbf{A}$ $(p \times 1)$ matrix, the middle one is the $\mathbf{\Gamma}$ $(K \times 1)$ matrix and the most right is the resulted $\mathbf{B}$ $(p \times K)$ matrix ($\mathbf{B} = \mathbf{A}\mathbf{\Gamma}^\top$). The non-zero values in $\mathbf{A}$ and $\mathbf{\Gamma}$, are represented with rand and those are i.i.d. values drawn from $U[0, 1]$. The non-zero entries of $\mathbf{B}$ are products of the corresponding values in $\mathbf{A}$ and $\mathbf{\Gamma}$, represented with $B_{ij}, i = 1, .., p, j = 1, .., K$, where $B_{ij} = A_i\Gamma_j$.

$$\begin{pmatrix} \text{rand}_{11} & 0_{12} \\ \text{rand}_{21} & 0_{22} \\ \vdots & \vdots \\ \text{rand}_{\frac{p}{2}1} & 0_{\frac{p}{2}2} \\ 0_{(\frac{p}{2}+1)1} & \text{rand}_{\frac{p}{2}2} \\ \vdots & \vdots \\ 0_{(p-1)1} & \text{rand}_{\frac{p}{2}2} \\ 0_{p1} & \text{rand}_{\frac{p}{2}2} \end{pmatrix} \quad \begin{pmatrix} \text{rand}_{11} & 0_{12} \\ \text{rand}_{21} & 0_{22} \\ \vdots & \vdots \\ \text{rand}_{\frac{K}{2}1} & 0_{\frac{K}{2}2} \\ 0_{(\frac{K}{2}+1)1} & \text{rand}_{\frac{K}{2}2} \\ \vdots & \vdots \\ 0_{(K-1)1} & \text{rand}_{\frac{K}{2}2} \\ 0_{K1} & \text{rand}_{\frac{K}{2}2} \end{pmatrix} \quad \begin{pmatrix} B_{11} & \cdots & B_{1\frac{K}{2}} & 0_{1(\frac{K}{2}+1)} & \cdots & 0_{1K} & \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{\frac{p}{2}1} & \cdots & B_{\frac{p}{2}\frac{K}{2}} & 0_{\frac{p}{2}(\frac{K}{2}+1)} & \cdots & 0_{\frac{p}{2}K} & \\ 0_{(\frac{p}{2}+1)1} & \cdots & 0_{(\frac{p}{2}+1)\frac{K}{2}} & \text{B}_{(\frac{p}{2}+1)(\frac{K}{2}+1)} & \cdots & \text{B}_{(\frac{p}{2}+1)K} & \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{p1} & \cdots & 0_{p\frac{K}{2}} & \text{B}_{p(\frac{K}{2}+1)} & \cdots & \text{B}_{pK} & \end{pmatrix}$$

Figure 2: This figure represents the case, for which $\mathbf{A}$ and $\mathbf{\Gamma}$ matrices are simulated under rank 2 (R=2), using block structure way of simulation ('case 2'). The leftmost matrix is the $\mathbf{A}$ $(p \times 2)$ matrix, the middle one is the $\mathbf{\Gamma}$ $(K \times 2)$ matrix and the most right is the resulted $\mathbf{B}$ $(p \times K)$ matrix ($\mathbf{B} = \mathbf{A}\mathbf{\Gamma}^\top$). The non-zero values in $\mathbf{A}$ and $\mathbf{\Gamma}$, are represented with rand and those are i.i.d. values drawn from $U[0, 1]$. The non-zero entries of $\mathbf{B}$ are products of the corresponding values in $\mathbf{A}$ and $\mathbf{\Gamma}$, represented with $B_{ij}, i = 1, .., p, j = 1, .., K$, where $B_{ij} = A_{i1}\Gamma_{j1} + A_{i2}\Gamma_{j2}$.

2. For step 2 to generate $\mathbf{Z}$ with correlation, firstly, a $p \times p$ matrix, denoted as $\mathbf{\Sigma}$ is generated, with all diagonal values set to 1 and all off-diagonal elements set to the target correlation value. Using this correlation matrix and a mean vector of zeros ($\vec{0}$), n samples (n vectors of length p) are drawn from a multivariate normal distribution: $\mathbf{Z} \sim \mathcal{N}_p(\vec{0}, \Sigma)$. The resulting matrix $\mathbf{Z}$ is of dimension $n \times p$, where rows represent observations and columns represent covariates. Because of this generation procedure, every pair of columns (covariates) in $\mathbf{Z}$ has an estimated correlation that (asymptotically) matches the desired correlation.

The Figures regarding the simulated datasets, for both simulation studies 1 and 2, are shown in the Appendix and aim to confirm the conditions that data are supposed to follow. Specifically, Figures in Appendix sections 6.2.1.1 (simulation study 1) and 6.2.2.1 (simulation study 2) show the distribution of predictors (on the left of each sub-figure) and the distribution of censored times (on the right of each sub-figure) of the S generated datasets per scenario under rank 1 and 2 respectively. As illustrated all predictors follow very similar distribution, normal with mean 0 and standard deviation 1 for all the different combinations of parameters. The distribution of the time up until a subject is observed (Tstop) is right-skewed meaning that most observations have shorter times until an outcome occurs and fewer observations have later times. This distribution aligns with the initial goal to have between $54\% - 63\%$ of events occurring.

3.1.3 Estimands

For both simulation studies 1 and 2, the primary estimands comprise the fitting rank (R) of the model, the elements of the coefficient matrix $\mathbf{B}$ of dimensions $p \times K$, and the linear predictor matrix $\mathbf{ZB}$ of dimensions $n \times K$.

3.1.4 Methods

3.1.4.1 Simulation study 1

simulation study 1 involves fitting the unpenalized version of the RRR model for survival data. The models are fitted using `R` software, specifically utilizing the *mstate* (redrank function) [31]. Regardless of the rank used in their simulation, all models are fitted with up to three distinct ranks (1, 2, or 3). The phrase *up to* means that the fitting process is sequential, starting with rank 1, moving on to rank 2, and only proceeding to rank 3 if rank 2 outperforms rank 1 in performance regarding the average error of linear predictor, the estimation of which is described in detail in the performance measures section of this chapter. This approach is selected to avoid unnecessary computational expense; fitting rank 3 is generally more time-consuming due to the increased number of parameters that should be estimated. Fitting models with different ranks allows for examining the impact of model rank on the accuracy of parameter estimation. Specifically, the results can be used to assess what the effect is of fitting a model when assuming

a different rank (smaller/larger) than what would be in line with the data-generating process. In order for the simulated datasets and the results to be reproducible a certain seed is set. It is important to highlight here that for similar studies it should be taken into account if, with the selected seed, the resulting $\mathbf{B}$ matrices exhibit the desired rank (the expected number of linearly independent columns). To assess the rank of the $\mathbf{B}$ matrix, singular value decomposition (SVD) is used [32]. SVD decomposes a matrix into three components: $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$, where $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values of the matrix ordered by magnitude. The number of nonzero singular values of a matrix is equal to the rank of that matrix. So, for an exact rank 1 matrix, only the first singular value should be significantly non-zero, whereas, for a rank 2 matrix, both the first and second singular values should be significantly non-zero and larger than the remaining values, which are very close to zero. In this study, to classify a matrix as 'close to rank 1', the ratio of the second-largest singular value to the largest one is calculated. If this ratio is below a small threshold (0.01), the matrix is considered approximately rank 1. This threshold is chosen as a practical criterion to flag matrices where the second singular value is very small compared to the first, so closer to the zero singular values [33].

3.1.4.2 Simulation study 2

In addition to fitting the unpenalized version of RRR Cox proportional hazards models, its penalized version using the LASSO penalty is fitted in simulation study 2. We consider the implementation based on `glmnet` [2], discussed in chapter 2. For this implementation, an initial $\mathbf{\Gamma}$ $(K \times R)$ matrix should be provided as input for the iterative procedure to begin. The elements of this $\mathbf{\Gamma}$ matrix are generated from a standard normal distribution, with a set seed. Similar to simulation study 1, all models are fitted again with up to three different ranks (1,2,3). However, this second simulation study requires the optimal value for the regularization parameter λ. For choosing that value the first five simulated datasets of each scenario are considered because considering all 100 MC trial datasets would have been too time-consuming. Each of these five datasets is initially fitted using ten different λ values, followed by up to three additional zooming steps between those values based on the initial results. The zooming process is detailed in Appendix Section 6.2.2.2, with an illustrative example provided in Figure A17. The λ that yields the smallest linear predictor error for each dataset is considered the optimal value for that dataset. This error is computed by calculating the squared difference between $\boldsymbol{\eta}$-true and $\boldsymbol{\eta}$-estimated, taking the row-wise average of the resulting $n \times K$ matrix, and computing the mean of the resulting n-dimensional vector. This metric is chosen for λ selection because the primary objective is to optimize predictive accuracy. The ten values that are initially used for each scenario are logarithmically spaced between 10^{-10} and the largest value of λ at which the first simulated dataset for that scenario successfully converges to the desired result without producing errors. This upper limit is determined manually by testing values differing in the second decimal place (e.g., 1.21, 1.22, 1.23) and selecting the largest value that can fit the desired model, without

producing errors (e.g., if 1.23 caused errors and 1.22 did not, then 1.22 is chosen). From each zooming step within each scenario, the median of the five optimal λ values is calculated. For this median λ, the average linear predictor error across the five datasets is computed at each zoom level. The median λ value of the zoom level that yields the smallest average linear predictor error (compared to the other zoom levels) is selected as the final optimal λ and used in the fitting procedure for all 100 simulated datasets in that scenario (95 remaining since the first 5 are already fitted). This is the last zoom plot in most scenarios but sometimes is not because the λ values tested in each zoom level are generated according to the prior zoom performed and to be log-spaced, as described in Figure A17. All the selected zoom plots, based on which the λ for each scenario is chosen are presented in Figures A18-A23 in the Appendix.

3.1.5 Performance measures

The metrics used for evaluating the performance of the fitted models in the different scenarios are described here and they are the same for both simulation studies 1 and 2.

Firstly, the values of absolute bias (absolute difference between the estimated and true coefficients) for each simulated dataset are calculated. These are averaged over all MC trials to calculate the average absolute bias per scenario.

To be more specific on how this measure is calculated, the absolute bias for simulation s is derived as:

$$\text{AbsoluteBias}_s = \frac{1}{pK} \sum_{j=1}^{pK} |\hat{\beta}_{sj} - \beta_{sj}| \tag{3.2}$$

And then, the average absolute bias for all simulations, for a given scenario, is calculated as:

$$\text{Average absolute bias} = \frac{1}{S} \sum_{i=1}^{S} \text{AbsoluteBias}_i \tag{3.3}$$

where S is the number of simulated datasets ($S = 200$ for simulation study 1 and $S = 100$ for simulation study 2), s is the specific simulation run for which the calculation is made ($s = 1, 2, ..., 200$ or $s = 1, 2, ..., 100$), $\hat{\beta}_{sj}$ is the $j-th$ element of the $pK \times 1$ vector with all estimated coefficients for simulated dataset s and β_{sj} is the $j-th$ element of the $pK \times 1$ vector with all true coefficients.

The next performance measure that is utilized is the Mean Squared Error of the regression coefficients(referred to as MSE). This is the average of the squared differences between the estimated and true coefficients. Specifically, for simulation s:

$$\text{MSE}_s = \frac{1}{pK} \sum_{j=1}^{pK} (\hat{\beta}_{sj} - \beta_{sj})^2 \tag{3.4}$$

For all simulations:

$$\text{Average MSE} = \frac{1}{S}\sum_{i=1}^{S}\text{MSE}_i, \tag{3.5}$$

where S, s, $\hat{\beta}_{sj}$ and β_{sj} are defined as before.

The average MSE and average absolute bias of the coefficient matrix $\mathbf{B}$, are not considered enough for calculating the performance of the models, since they fall short of capturing their predictive power. Absolute bias detects systemic patterns of differences between the estimation in the parameter space compared and the reality and because with penalization we already introduce bias it is considered insufficient to rely only on this metric. Regarding the MSE, it calculates the average squared deviation between the estimated and true coefficients. However, when covariates have a minimal effect on outcomes, even incorrect estimations can result in a low MSE because small changes in predictors with weak influence do not lead to substantial error. For example, heatmaps comparing estimated and true coefficient matrices might reveal notable discrepancies despite a low MSE. This can mask weaknesses in the model's ability to recover true parameters, as a low MSE might falsely suggest reliable estimation performance. Taking those facts into account, even with small absolute bias and MSE, the predictive performance of the model (capacity to produce precise predictions for specific subjects), which is frequently of most interest in real-world applications, is unclear.

To address this gap, we evaluate the expected predictive performance of each model by calculating the discrepancy between the true and estimated linear predictors ($\mathbf{ZB}$ and $\mathbf{Z\hat{B}}$). To clarify how this is calculated all the details are given here. Remember that all simulated datasets, under the same combination of parameters, share the same true $p \times K$ coefficient matrix $\mathbf{B}$. However, each simulated dataset has a different $n \times K$ covariate matrix $\mathbf{Z}_s$, under a specific combination of parameters. Hence, fitting each simulated dataset results in a different estimated $(p \times K)$ $\mathbf{\hat{B}_s}$ matrix. So, to calculate the linear predictor error between the true linear predictor, $\mathbf{Z_sB}$, and the estimated linear predictor of simulation s, $\mathbf{Z_s\hat{B}_s}$, firstly $\mathbf{D_s}$ is computed as:

$$\mathbf{D_s} = (\mathbf{Z_sB} - \mathbf{Z_s\hat{B}_s})^2, \quad \text{element-wise square and } \mathbf{D}_s \in \mathbb{R}^{n\times K}, \tag{3.6}$$

Then the mean of each row, $i = 1, .., n$, of D_s is taken to obtain a single linear predictor error per subject. This hence results in an $n \times 1$ vector:

$$\text{rowMean}_{\mathbf{D}_{s_i}} = \frac{1}{K}\sum_{k=1}^{K} D_{s_{ik}} \tag{3.7}$$

Finally, the average of this vector gives the overall mean error for the dataset s:

$$\text{linear predictor error}_s = \frac{1}{n}\sum_{i=1}^{n}\text{rowMean}_{D_{s_i}} \tag{3.8}$$

This is what is called linear predictor error (LPE). The same procedure is followed for all simulation runs, after which the mean over all S runs is taken to obtain the average LPE:

$$\text{Average linear predictor error} = \frac{1}{S}\sum_{s=1}^{S}(\text{linear predictor error})_s \tag{3.9}$$

The Monte Carlo (MC) error is calculated for the three discussed evaluation measures as follows:

Absolute Bias MC error: Let $\text{AbsoluteBias}_{sj} = |\hat{\beta}_{sj} - \beta_{sj}|$ for each element j in simulation s, as used in Equation (3.2). The MC error is:

$$\text{MC}_{\text{error,AbsBias}} = \frac{\sigma_{\text{AbsBias}}}{\sqrt{p \times K \times S}} \tag{3.10}$$

where σ_{AbsBias} is the standard deviation of all individual AbsoluteBias_{sj} values across all coefficients and simulations ($p \times K \times S$ vector).

MSE MC error: Let $\text{MSE}_{sj} = (\hat{\beta}_{sj} - \beta_{sj})^2$ for each element j in simulation s, as used in Equation (3.4). The MC error is:

$$\text{MC}_{\text{error,MSE}} = \frac{\sigma_{\text{MSE}}}{\sqrt{p \times K \times S}} \tag{3.11}$$

where σ_{MSE} is the standard deviation of all individual MSE_{sj} values across all coefficients and simulations ($p \times K \times S$ vector).

LPE MC error: Using the row means calculated in Equation (3.7), the MC error is:

$$\text{MC}_{\text{error,LPE}} = \frac{\sigma_{\text{LPE}}}{\sqrt{n \times S}} \tag{3.12}$$

where σ_{LPE} is the standard deviation of all $\text{rowMean}_{A_{s_i}}$ values across all subjects and simulations ($n \times S$ vector).

The final evaluation measure that we consider is correlation. For each scenario, we calculate and save the following correlations: (1) between the simulated $\mathbf{B}$ matrix and the estimated $\hat{\mathbf{B}}_\mathbf{s}$ matrices, and (2) between the true linear predictor ($\mathbf{Z_s B}$) and the linear predictor estimated using the model ($\mathbf{Z_s}\hat{\mathbf{B}}$). For computing those correlations the vectorized matrices are used. This results in two 200×1 vectors per combination of parameters (one for $\mathbf{B}$ matrices and one for linear predictor matrices). The mean and standard deviation of those vectors are then computed and reported.

3.2 Simulation study results

In this section, the results of simulation studies 1 and 2 are presented, and key observations are highlighted for all the examined scenarios.

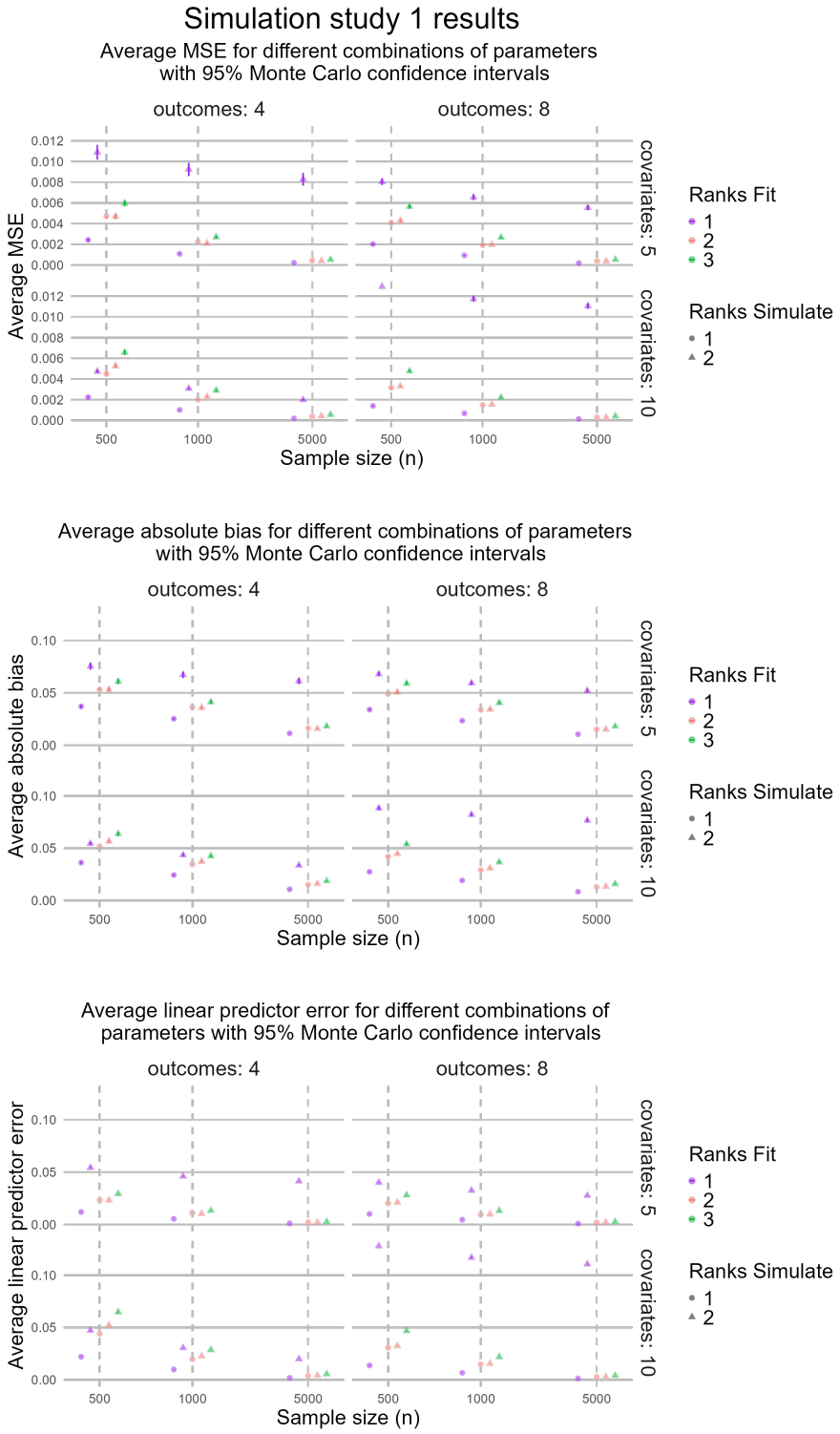

Figure 3: The average values for MSE (up), absolute bias (middle), and linear predictor error (bottom) for survival data simulated with rank 1 or 2 and fitted using the unpenalized RRR model for survival data with ranks 1, 2, or 3. The results are shown for different combinations of covariates (5 or 10), outcomes (4 or 8), and sample sizes (500, 1000, 5000). Monte Carlo confidence intervals are also included, represented as vertical lines; for points where the intervals are not visible, they are too small to be detected. For each combination of parameters (number of covariates, outcomes, and sample size) 200 datasets are simulated. Direct comparisons cannot be made between the three graphs since the 3 y-axes have different numbering

3.2.1 Simulation study 1

Figure 3 summarizes the average MSE (up), absolute bias (middle) and LPE (bottom), for all the different combinations of considered parameters. Monte Carlo (MC) 95% confidence intervals (Average MSE $\pm$ 1.96 $\cdot$ MSE MC error for the left plot, Average absolute bias $\pm$ 1.96 $\cdot$ absolute bias MC error for the middle plot and Average linear predictor error $\pm$ 1.96 $\cdot$ LPE MC error for the right plot) are included in this figure as vertical lines. In most scenarios the error bars are too small to be visible. The MC errors and Average values for MSE, absolute bias, and LPE are given in Tables A1–A9 in the Appendix, for all considered scenarios. Figure 3 demonstrates that if the true data-generating mechanism is rank 1, then the best performance (i.e., smaller MSE, absolute bias, and LPE) is achieved when also fitting a rank 1 model. The same holds for models that are generated under rank 2: the best performance is achieved when fitting a rank 2 model. This holds for all three considered evaluation metrics. This is what is expected and highlights that the model is working properly.

Another observation from Figure 3 is that as the sample size increases for each combination of predictors (p) and outcomes (K), the performance is improved with lower values of absolute bias, MSE and LPE achieved, as well as smaller MC errors. Also, when observing the models simulated under rank 1, which are represented by bullets, it can be noted that they are all relatively close to one another. On the other hand, when examining the data simulated using rank 2, although the performance remains very similar whether fitted with rank 2 or 3, the results are notably different when fitted with rank 1. In this case, the points (purple triangles) stand out clearly from the others and exhibit the largest MC errors.

Regarding the patterns followed across the different combinations of the number of outcomes and covariates, those are generally consistent. However, the plots reveal that when data is simulated with rank 2 but fitted with rank 1, performance differences emerge among the parameter combinations. Specifically, the combination of 4 outcomes and 10 covariates demonstrates the best performance. Contrary, the other three combinations have much worse performances, similar to one another, with the one having 4 outcomes and 5 covariates though being the most unstable with the highest MC errors. In general, given the above observations from Figure 3 it can be said that the trends observed for average MSE also apply to average absolute bias and average LPE. The improved performance of the ($p = 10$, $K = 4$) configuration and the different performance patterns across various parameter combinations demonstrate that model success is not just based on the predictor-to-outcome ratio when the model tries to be fitted with a smaller rank than the one it really is and thus, cannot fully capture the relationships in the data.

Finally, the Tables A10-A12 in Appendix subsection 6.2.1.2, show the values for the mean and standard deviation of correlations between $\mathbf{B}$ and between $\mathbf{ZB}$ matrices. For all the combinations of parameters both the mean $\mathbf{B}$ cor and mean $\mathbf{ZB}$ cor are very close to 1 (bigger than 0.9) suggesting both a strong accuracy in estimating the true coefficients $\mathbf{B}$, and a reliable prediction of the linear predictors $\mathbf{ZB}$. Also, the standard deviations are very small (smaller than 0.02), meaning that the model's performance is stable across simulations. The heatmaps of simulated

B matrices and the mean of estimated ones by models of different ranks are also provided in 6.2.1.3 subsection of the Appendix as a visual way to illustrate how close the true and estimated **B**. For all the scenarios considered, those heatmaps show that the estimated **B** matrices are very close to the true ones, especially when the rank of the fitted model is equal to the rank of the simulated data.

3.2.2 Simulation study 2

All scenarios are initially fitted with a λ value of 1e-10, which is nearly unpenalized. The performance metrics for these nearly unpenalized models are clearly worse compared to those with penalization highlighting that a model with a penalty should be used for them. The upper-left plot of A17 is an example of how the initial plots with different values of λ looked like for the smallest λ values.

Figure 4 corresponds to Figure 3. So, it again summarizes the average MSE (up), absolute bias (middle), and LPE (bottom), for all the different combinations of parameters that are used in simulation study 2. The left figures are for the 'case 1' and 'case 2' way of simulating the data and the right ones are for 'case 3'. Monte Carlo (MC) 95% confidence intervals are again included in all the plots of Figure 4 as vertical lines, but are too small to be clearly visible. The MC errors and average values for MSE, absolute bias, and LPE are given in Tables A13–A18 in the Appendix, for all considered scenarios. Figure 4 shows (like Figure 3) that if the true data-generating mechanism is rank 1, then the best performance (i.e., smallest MSE, absolute bias, and LPE) is achieved when also fitting a rank 1 model.

However, the same does not hold for the data which are generated under rank 2. Only for the scenario with data simulated using block structure ('case 2'), correlation of 0.7 and 1000 samples this is the case (for all performance measures) and that is why is the only scenario for which the model is fitted using additionally rank 3. The rank 3 fitted model performed slightly worse than the rank 2 model and better than the rank 1 model for this scenario for all three metrics. Regarding the rest of the scenarios, different patterns are observed for the three evaluation metrics, so they are described by considering each data-generating mechanism independently. Firstly, for the scenario with 'case 2', correlation 0.7, but 5000 samples (instead of 1000 mentioned earlier), the model fitted with rank 1 performed slightly better in terms of average LPE and MSE values, and slightly worse in terms of the resulted average absolute bias value. For scenarios with 'case 1' and correlation 0.7, for both sample sizes (1000 and 5000) examined, the models fitted using rank 1 performed slightly better regarding their average LPE value and slightly worse regarding their average MSE and absolute bias values. For 'case 3' and sample size 1000 the rank 1 model performed better for average MSE and LPE and worse for average absolute bias. Finally, for the same scenario, but with 5000 samples, only for average LPE, the rank 1 model, performed better and worse for the other two metrics.

The same observation as the one made for Figure 3, regarding the increase in sample size,

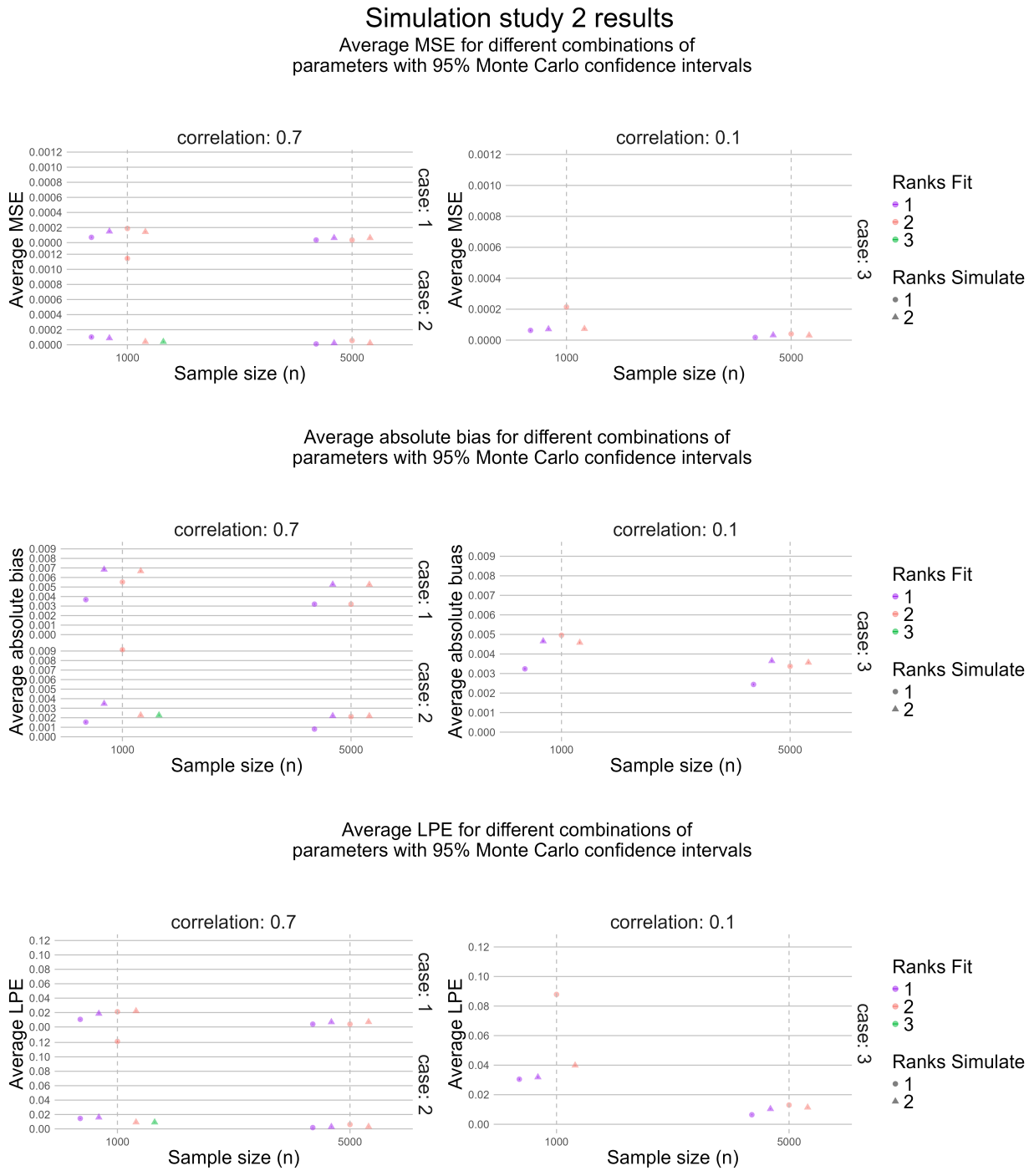

Figure 4: This figure presents the average values for MSE (up), absolute bias (middle), and linear predictor error (bottom) for survival data simulated with rank 1 or 2 and fitted using the LASSO penalized RRR model (`glmnet` version) for survival data with ranks 1, 2, or 3. The results are shown for different combinations of parameters (way for generating data-case 1, 2, or 3-, correlation-0.7 or 0.1 -and sample sizes-1000 or 5000-). Monte Carlo confidence intervals are also included, represented as vertical lines; for points where the intervals are not visible, they are too small to be detected. For each combination of parameters, 100 datasets are simulated with 300 predictors (p) and 8 outcomes (K). Direct comparisons cannot be made between the different graphs since the y-axes have different numbering

can also be made for Figure 4. As the sample size increases for each scenario, the performance is improved with lower values of average absolute bias, MSE, and LPE achieved, as well as smaller MC errors. Also, when observing the points of the models with 5000 samples (for all the scenarios), it can be derived that their clustering is closer, compared to the clustering when the sample size is 1000, for which there are sometimes outliers among the points. Consequently, the performance of most fitted models—whether the data are simulated under rank 1 or rank 2—is quite similar, especially when the sample size is 5000 (as indicated by the closeness of the bullets and triangles along the y-axis). Regarding the outliers mentioned, those are observed when the sample size is 1000 and 'case 2' (correlation 0.7) or 'case 3' (correlation 0.1) is used. These outliers occur for the models simulated using rank 1 and fitted using rank 2. Represented by orange bullets, these points exhibit significantly worse performance compared to the other models, standing out clearly with the highest error values across all metrics and plots. Nevertheless, for the 'case 3' scenario, the pattern is a bit different when it comes to the average absolute bias measure. While the model simulated under rank 1 and fitted under rank 2 still performs poorly compared to the fitted under rank 1 model, the other two models (represented by triangles in the plot) also show larger deviations from the fitted under rank 1 model and are closer in performance to the poorly performing model. As a result, for the average absolute bias measure, the model simulated and fitted under rank 1 clearly stands out as the best performer.

Finally, the tables A19 and A20 in Appendix subsection 6.2.2.3, show the values for the mean and standard deviation of correlations between **B** and between **ZB** matrices for the different scenarios considered. These tables reveal that for a sample size of 1000 (Table A19), the mean correlation values for the **B** matrices range between 0.00875 and 0.2731, with standard deviations varying from 0.0191 to 0.0664. The corresponding mean correlations for the **ZB** matrices demonstrate significantly higher consistency, with values between 0.2472 and 0.9949, and smaller standard deviations (0.0009 to 0.3158), reflecting the stability of the predictions in these scenarios. Notably, the lowest mean correlations and highest standard deviations are for the simulated under rank 1 but fitted with rank 2 models, which had the worst performance also in the previously described metrics.

In contrast, for a sample size of 5000 (Table A20), the mean correlations for the **B** matrices increase, ranging from 0.0883 to 0.4332, with similar standard deviations (0.0298 to 0.0684). Similarly, the **ZB** matrices maintain high mean correlation values, between 0.8858 and 0.9981, with even smaller standard deviations (0.0002 to 0.1310), emphasizing improved precision with a larger sample size.

The heatmaps of simulated **B** matrices and the mean of estimated ones by the different models are additionally presented in 6.2.2.4 subsection of the Appendix as a visual way to illustrate how close the true and estimated **B** matrices are. The heatmaps indicate that in most scenarios—except when **B** is generated with a block structure (i.e., 'case 2')—the model successfully approximates the true **B**, especially when the model is fitted with the same rank as the one used for simulating the data. However, in 'case 2', where simulated **B** has a block

structure, the estimated matrices fail to retain this structure, even though they still exhibit some similarities to the true ones.

The aforementioned results illustrate the model's effectiveness in approximating the true **B** matrices, while also revealing its limitations in capturing the structure of **B** when simulated with the very specific block structure used in this thesis, even if MSE is low. On the other hand, regardless of the scenario—including situations where **B** is generated with a block structure ('case 2')—the model consistently succeeds in approximating the true **ZB** matrices.

Chapter 4

Software implementation of penalized RRR for survival data using ADMM

The details regarding how Algorithm 3 (as detailed in chapter 3), has been implemented in `R`, are given here. This implementation enables direct comparison with the existing approach based on the `R glmnet` package [2]. Considering that both strategies ought to provide comparable outcomes, comparing both implementations is crucial to confirming ADMM as an appropriate method to apply penalization to the survRRR model. A strong basis for extending ADMM to more intricate penalties that can not be directly implementable with existing packages—such as the group LASSO penalty covered in chapter 3—is thus established by this validation.

4.1 Implementation details

The process outlined in Algorithm 3 is followed to apply the LASSO penalty to the survRRR model. The `optim` function from the `stats` package in `R` is employed to optimize the partial log-likelihood, as needed in stages 7 and 17 of the procedure. The objective function, which takes into account both the LASSO penalty term and the negative partial log-likelihood, is efficiently minimized using this method.

The log-likelihood function is computed for each outcome using the Cox proportional hazards model and specifically, the `coxph` function. Numerical differentiation is carried out using the `grad` function to determine the gradient of the objective function with respect to the vectorized $\mathbf{A}$ matrix ($\boldsymbol{\alpha}$), for step 7, or vectorized $\boldsymbol{\Gamma}$ matrix ($\boldsymbol{\gamma}$), for step 17. This function allows for an accurate approximation of the gradient. It estimates the derivative by perturbing the input vector and assessing the change in the log-likelihood function.

The `L-BFGS-B` approach in `optim` is used to carry out the optimization. To make sure the parameters stay within acceptable limits, lower and upper boundaries are set to -4 and 4. Tolerances for the projected gradient ($pgtol = 1e-7$) and a maximum number of iterations ($maxit = 1e4$) are set as the optimization's stopping criterion, guaranteeing convergence while avoiding over-computing.

For all computations, the regularization parameter of the algorithm, ρ, is fixed to 1, while the convergence thresholds $\epsilon_1 = 1e-5$ and ϵ_2, ϵ_3, ϵ_4, $\epsilon_5 = 1e-7$ are applied consistently throughout the algorithm. Finally, the loops in stages 7 and 17 of Algorithm 3 are programmed to iterate no more than 1000 times, guaranteeing termination in the event that residual convergence is not achieved.

4.2 Experiments

The estimated linear predictor matrix, $\boldsymbol{\eta} = \mathbf{ZB}$, and the estimated coefficient matrix, $\mathbf{B}$, are the two main estimands of interest of the experimental comparison between the two implementations. The examination focuses both at direct matrix elements comparisons as well as comparisons of performance measures, particularly the MSE and LPE calculated as explained in chapter 3. Since their similarity would indicate whether ADMM is appropriate for applying penalization to the RRR model for survival data, these components are chosen as the primary comparative metrics.

In order to balance algorithm testing and computational feasibility, the validation uses straightforward yet representative scenarios. Every scenario employs the same set of parameters: a sample size of 500 ($n = 500$), five predictors ($p = 5$), and four outcomes ($K = 4$). Matrix simulation follows the 'case 1' methodology described in chapter 3. Three crucial elements are examined for different values, with the aforementioned constant variables. The first parameter is the correlation between variables (cor), with values of 0 and 0.7. The λ value is examined also, at two different values, those being 0 and 0.0001. The value 0.0001 reflects a moderate penalty strength, chosen based on the experience gained from the simulation study since it was possible for this value to be used for fitting all the models, without giving errors. On the other hand, the zero-penalty scenario allows comparison in addition to the unpenalized RRR method for survival data from Fiocco et al. [9]. Finally, the values for rank (R) of the simulated $\mathbf{A}$ and $\boldsymbol{\Gamma}$ matrices are chosen to be 1 and 2.

Eight different datasets are produced by the experiment using the two-step procedure described in section 3.1.2 of chapter 3, with one dataset produced for each scenario ($S = 1$). All datasets are fitted with models using both rank-1 and rank-2. For scenarios with $\lambda = 0$, the comparison includes the unpenalized RRR for survival data, penalized RRR for survival data using ADMM, and penalized RRR for survival data using `glmnet`. In scenarios, where $\lambda = 0.0001$, the comparison excludes the unpenalized version of RRR model for survival data.

For the comparisons to be as fair as possible, the following values are set for the penalized survRRR with `glmnet` package: $eps = 1e-5$, $thresh = 1e-7$, $maxit = 1e4$. The heatmaps of

the estimated $\mathbf{B}$ and $\boldsymbol{\eta}$ matrices are presented in Figures 5,6,7,8 and the values of the errors per scenario for each model are presented in Table 1.

As can be observed from all the heatmaps, the estimated matrices are very similar, and the same holds for both the MSE and LPE values presented in the aforementioned table. The slight variations seen are probably caused by differences in the convergence criteria, thresholds, and underlying computing processes used by each method. It should be highlighted that no other comparisons, for determining the optimal rank should be made here, because not an optimal λ value is chosen. That was not considered necessary, since the goal is only the assess the appropriateness of LASSO penalized survRRR with ADMM by showing that it gives similar results with the other algorithms of interest.

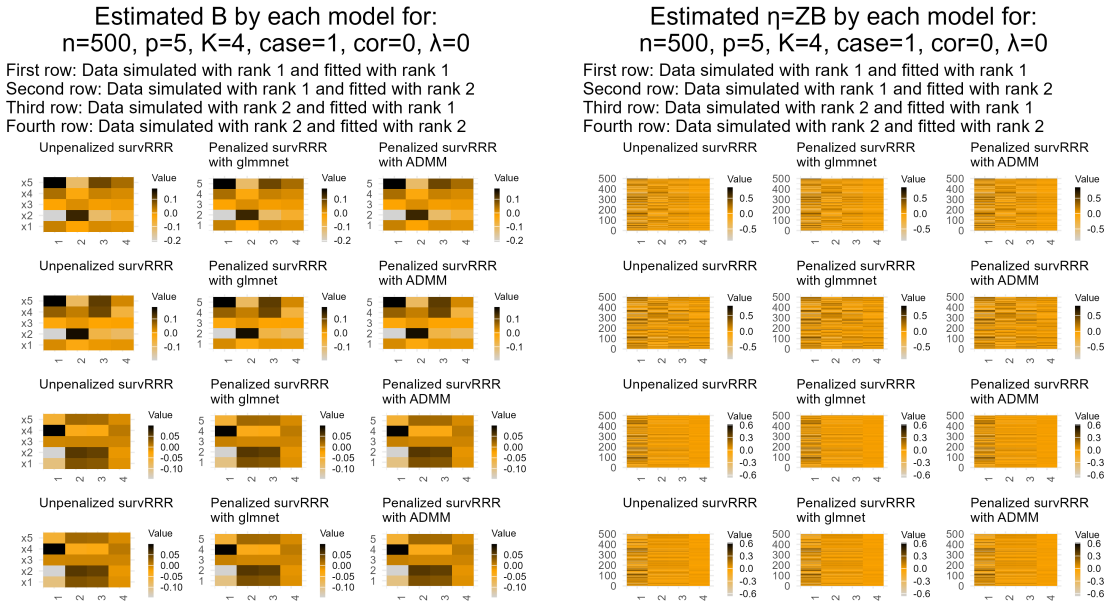

Figure 5: These figures present the heatmaps for the estimated $\mathbf{B}$ (left) and $\boldsymbol{\eta} = \mathbf{ZB}$ (right) matrices, using 3 different implementations of RRR model for survival data. The specific scenarios are mentioned in the title and subtitles.

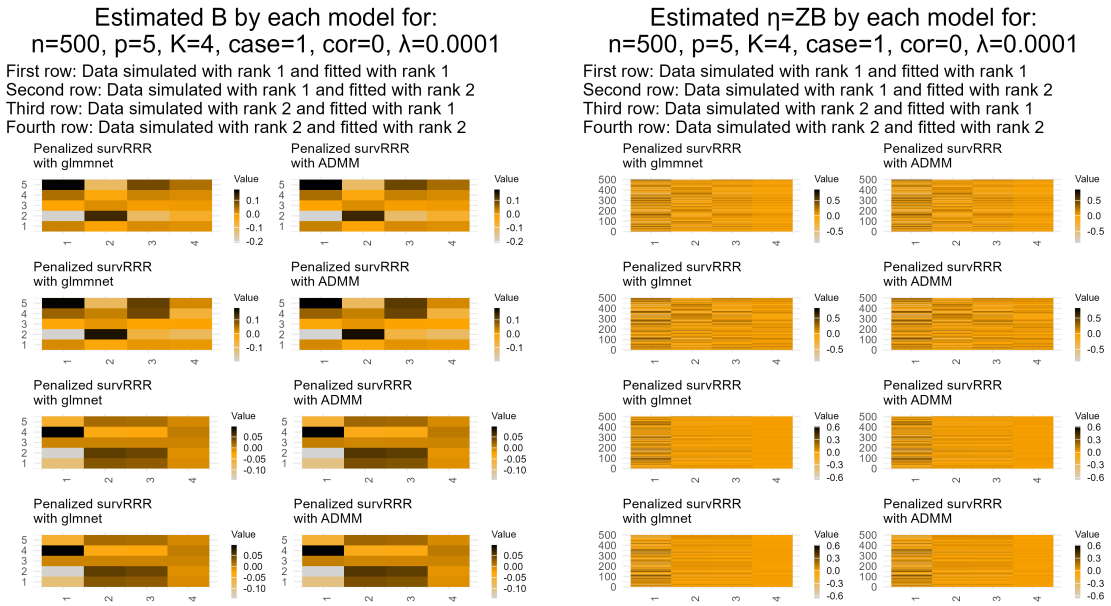

Figure 6: These figures present the heatmaps for the estimated $\mathbf{B}$ (left) and $\boldsymbol{\eta} = \mathbf{ZB}$ (right) matrices, using 3 different implementations of RRR model for survival data. The specific scenarios are mentioned in the title and subtitles.

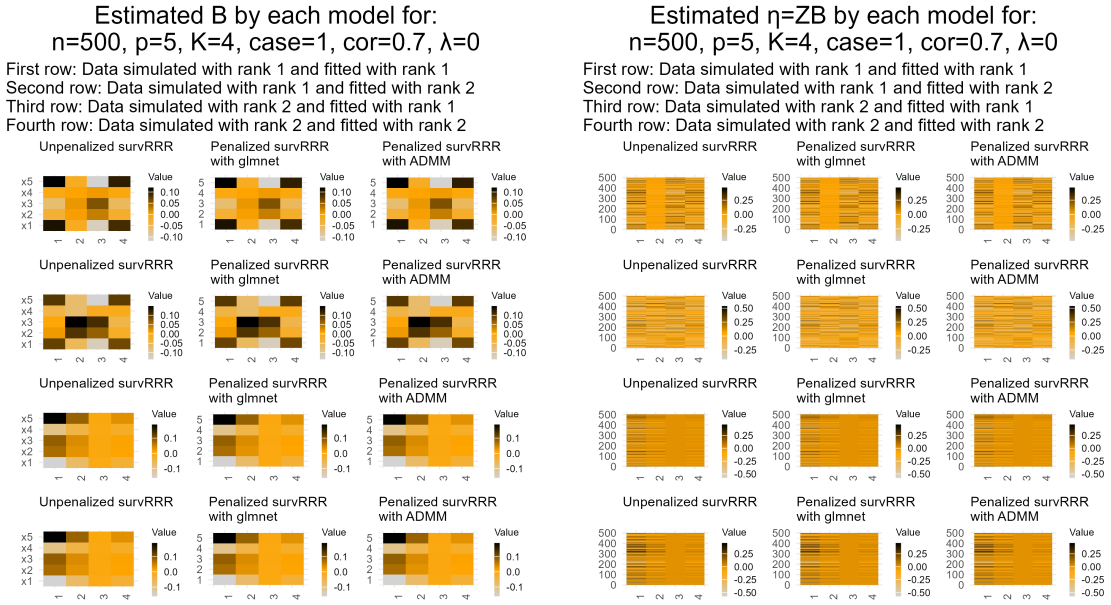

Figure 7: These figures present the heatmaps for the estimated $\mathbf{B}$ (left) and $\boldsymbol{\eta} = \mathbf{ZB}$ (right) matrices, using 2 different implementations of RRR model for survival data. The specific scenarios are mentioned in the title and subtitles.

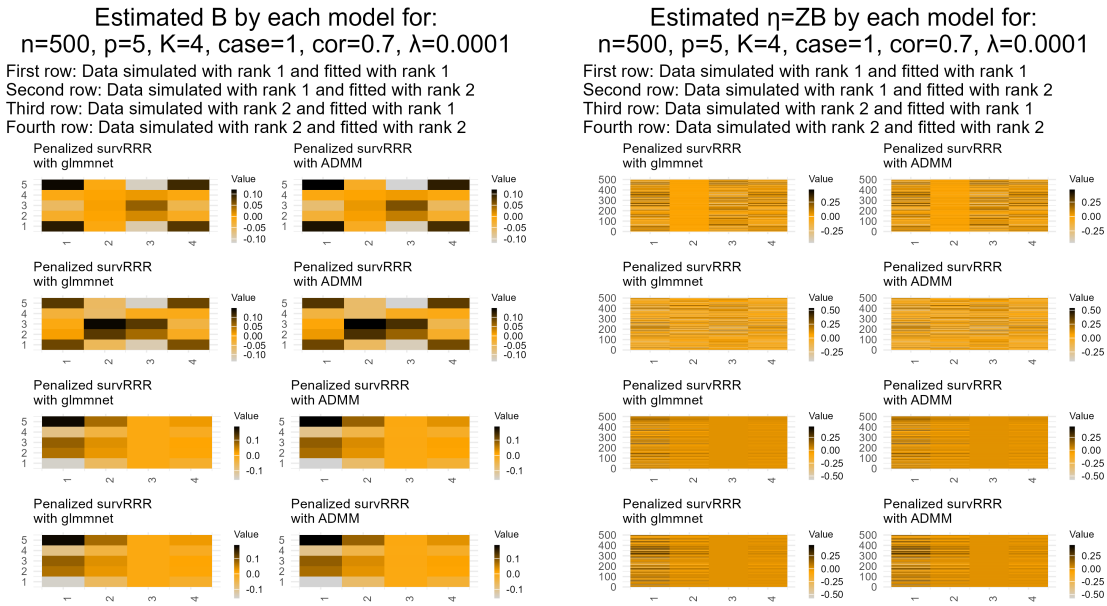

Figure 8: These figures present the heatmaps for the estimated $\mathbf{B}$ (left) and $\boldsymbol{\eta} = \mathbf{ZB}$ (right) matrices, using 2 different implementations of RRR model for survival data. The specific scenarios are mentioned in the title and subtitles.

	Cor.	Sim. rank	Fit rank	MSE			LPE		
				Unpen.	Pen.glmnet	Pen.ADMM	Unpen.	Pen.glmnet	Pen.ADMM
0	0	1	1	7.0120e-03	7.0062e-03	7.0098e-03	3.6617e-02	3.6595e-02	3.6614e-02
0	0	1	2	8.0141e-03	8.0134e-03	8.0135e-03	4.1745e-02	4.1739e-02	4.1740e-02
0	0	2	1	2.7596e-03	2.7663e-03	2.7669e-03	1.2877e-02	1.2824e-02	1.2828e-02
0	0	2	2	4.1247e-03	4.1192e-03	4.1192e-03	1.9288e-02	1.9272e-02	1.9272e-02
0	0.7	1	1	4.0348e-03	4.0301e-03	4.0520e-03	1.3383e-02	1.3381e-02	1.3392e-02
0	0.7	1	2	6.5172e-03	6.4784e-03	6.4846e-03	1.7342e-02	1.7311e-02	1.7327e-02
0	0.7	2	1	5.2216e-03	5.2630e-03	5.2568e-03	9.0512e-03	9.0434e-03	9.0443e-03
0	0.7	2	2	6.4572e-03	6.5364e-03	6.4996e-03	1.1868e-02	1.1875e-02	1.1872e-02
1e-04	0	1	1	-	6.6073e-03	7.0095e-03	-	3.4497e-02	3.6612e-02
1e-04	0	1	2	-	7.5476e-03	8.0118e-03	-	3.9303e-02	4.1729e-02
1e-04	0	2	1	-	2.5134e-03	2.7668e-03	-	1.1728e-02	1.2828e-02
1e-04	0	2	2	-	3.7571e-03	4.1092e-03	-	1.7583e-02	1.9239e-02
1e-04	0.7	1	1	-	3.3364e-03	4.0515e-03	-	1.1925e-02	1.3391e-02
1e-04	0.7	1	2	-	5.6563e-03	6.4250e-03	-	1.5500e-02	1.7308e-02
1e-04	0.7	2	1	-	4.5880e-03	5.2563e-03	-	7.9907e-03	9.0434e-03
1e-04	0.7	2	2	-	5.5919e-03	6.4982e-03	-	1.0452e-02	1.1869e-02

Table 1: Mean square error (MSE) and linear predictor error (LPE) values for different scenarios, that the unpenalized survRRR model, the penalized survRRR model with glmnet and the penalized survRRR with ADMM give.

Chapter 5

Discussion

5.1 Key Findings and Contributions

This thesis presents important contributions in the application of reduced rank regression (RRR) models to survival analysis, with a particular focus on incorporating penalization methods such as LASSO and group LASSO. Utilizing the optimization framework known as the Alternating Direction Method of Multipliers (ADMM), an algorithm that can handle intricate penalization structures (like LASSO and group LASSO) is presented in detail. Additionally, by following the detailed steps of the algorithm presented in this thesis, it is implemented in `R` for the LASSO penalty, with considerable potential to be extended to more complex penalties like the group LASSO. The proposed algorithm offers a framework for addressing key challenges in the analysis of multidimensional survival data, such as multicollinearity and grouping among predictors.

Moving on with the results from simulation study 1, they provide a baseline assessment of the performance of unpenalized reduced rank regression (RRR) models in the context of survival data. By focusing solely on unpenalized models with multiple outcomes, uncorrelated predictors and low-dimensional settings this study's results are as expected since the best-performing model of each scenarios is the one that is generated under the same rank as the one with which it is fitted.

As for simulation study 2, it reveals that penalization is significantly important since penalized RRR models consistently outperform their unpenalized counterparts ($\lambda = 1e - 10$) in settings with correlated predictors and a considerable high number of predictors. This underscores the critical importance of incorporating penalization in such scenarios. Additionally, we have found that for certain metrics, the models fitted using ranks 1 and 2 often perform similarly, with rank 1 sometimes outperforming rank 2, even when the data are generated using rank 2. This, means that the theoretical rank is not identified as the optimal one—across all scenarios and

performance metrics.

From both simulation studies is illustrated that in most of the times the real model can be recovered by both the unpenalized and penalized RRR models for survival data, but special attention should be given when choosing which model to use, and which value for λ and rank to choose for fitting the model. For scenarios like the ones of simulation 2 (high-dimensional with correlated predictors), choosing rank 1 for fitting the model seems like the safest and least computationally demanding choice, since in most scenarios those models have performed the best, and in circumstances for which they did not, their performance is very close to the models fitted with different ranks (2 or 3). However, for scenarios like the ones of simulation study 1 (low-dimensional and uncorrelated predictors) the choice of the rank needs to be considered more carefully. This is because the results illustrate that when data are simulated using rank 2 but fitted with rank 1, the performance is mostly notably worse than when fitted with rank 2. Nevertheless, the most safe option for any scenario is always to start by fitting models with both rank 1 and rank 2 and only if rank 2 outperforms rank 1, continue with higher ranks.

Also, from the results of simulation studies some important conclusions regarding the interpretability of **ZB** can be derived. The matrix **ZB**, representing the projection of predictors into the outcome space, is a more reliable and interpretable result. By directly capturing the combined effects of predictors on outcomes, **ZB** is more compatible with practical applications, such as prediction tasks, and provides a more useful summary of the model's output.

5.2 Limitations and Future Research

Even if this study makes significant contributions, there are a few limitations that should be discussed, which also provide promising routes for future research. First, while the simulation studies in this thesis are extensive, they are based on certain assumptions about the data-generating mechanisms. Expanding these assumptions, like testing the model on a broader range of datasets—including higher variability in the number of predictors, outcomes, ranks, and correlations among predictors, as well as data drawn from various distributions—, could provide even deeper insights about the applicability of survRRR model. Furthermore, situations in which there are more predictors than samples are not examined in this thesis. Such scenarios are likely to demand penalization, similar to the scenarios investigated in this thesis, with 300 predictors but more than 300 samples). Exploring them will help to clarify the performance of survRRR models in severe high-dimensional circumstances and provide a deeper understanding of the model's practical application.

Second, even if the penalty parameter λ is carefully determined within the study's computational restrictions, the values that are selected might not be the best ones, because they are approximated based only on the first five simulated datasets. This could explain why the theoretical rank is not identified as the optimal one across all scenarios and performance metrics.

Thus, more thorough hyperparameter tuning approaches, including cross-validation or automated search procedures, may improve the model's performance even more.

Third, another interesting direction for further study is the extension of the provided software implementation to include other penalization strategies, like the group LASSO. This might provide more flexibility, especially when dealing with grouped predictors or situations that need addressing both correlation and sparsity.

Finally, working with very high-dimensional datasets or a large number of outcomes may provide difficulties due to the ADMM algorithm's computational demands. Further optimization of its implementation—such as employing parallelization or distributed computing—could significantly improve its efficiency. With these improvements, it would be possible to use the framework for bigger datasets, such as those seen in longitudinal healthcare databases or population-level genomic investigations. Such developments would allow the suggested methodology to address a greater spectrum of real-world issues in addition to expanding its applicability.

Bibliography

[1] A. J. Izenman, "Reduced-rank regression for the multivariate linear model," *Journal of Multivariate Analysis*, vol. 5, no. 2, pp. 248–264, 1975.

[2] M. H. Sluiskes, H. Putter, M. Beekman, J. J. Goeman, and M. Rodríguez-Girondo, "Penalized reduced rank regression for multi-outcome survival data supports a common metabolic risk score for age-related diseases," *bioRxiv*, 2024.

[3] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 58, no. 1, pp. 267–288, 1996.

[4] D. Witten and R. Tibshirani, "Survival analysis with high-dimensional covariates," *Statistical Methods in Medical Research*, 2010.

[5] S. Salerno1 and Y. Li1, "High-dimensional survival analysis: Methods and applications," *ANNUAL REVIEW OF STATISTICS AND ITS APPLICATION*, 2023.

[6] G. L. Hickey, P. Philipson, A. Jorgensen, and R. Kolamunnage-Dona, "Joint modelling of time-to-event and multivariate longitudinal outcomes: recent developments and issues," *BMC Medical Research Methodology*, vol. 16, no. 1, p. 117, 2016.

[7] K. Mauff, D. Rizopoulos, S. Cro, and L. de Wreede, "Joint models with multiple longitudinal outcomes and a time-to-event outcome: a corrected two-stage approach," *Statistics and Computing*, vol. 30, no. 3, pp. 625–640, 2020.

[8] C. Luo, J. Liang, G. Li, F. Wang, C. Zhang, D. K. Dey, and K. Chen, "Leveraging mixed and incomplete outcomes via reduced-rank modeling," *Journal of Multivariate Analysis*, vol. 167, pp. 378–394, 2018.

[9] M. Fiocco, H. Putter, and J. C. van Houwelingen, "Reduced rank proportional hazards model for competing risks," *Journal Name*, 2005.

[10] M. Fiocco, H. Putter, C. van de Velde, and J. van Houwelingen, "Reduced rank proportional hazards model for competing risks: An application to a breast cancer trial," tech. rep.,

Department of Medical Statistics and Bioinformatics, Leiden University Medical Center, P.O. Box 9604, 2300 RC Leiden, The Netherlands, 2005.

[11] J. Qian, Y. Tanigawa, R. Li, R. Tibshirani, M. A. Rivas, and T. Hastie, "Large-scale multivariate sparse regression with applications to uk biobank," *Journal Name*, 2022.

[12] J. P. Klein and M. L. Moeschberger, *Survival Analysis: Techniques for Censored and Truncated Data*. New York: Springer, second ed., 2003.

[13] D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society, Series B*, vol. 34, pp. 187–220, 1972.

[14] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[15] D. M. McNeish, "Using lasso for predictor selection and to assuage overfitting: A method long overlooked in behavioral sciences," *Multivariate Behavioral Research*, vol. 50, no. 5, pp. 471–484, 2015. PMID: 26610247.

[16] V. C. Nguyen and C. T. Ng, "Variable selection under multicollinearity using modified log penalty," *Journal of Applied Statistics*, vol. 47, no. 2, pp. 201–230, 2020. PMID: 35706515.

[17] M. Yuan and Y. Lin, "Model Selection and Estimation in Regression with Grouped Variables," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 68, pp. 49–67, 12 2005.

[18] T. W. Anderson, "Estimating linear restrictions on regression coefficients for multivariate normal distributions," *Annals of Mathematical Statistics*, vol. 22, pp. 327–351, 1951.

[19] E. Bura and J. Yang, "Dimension reduction for multivariate response regression," *Statistica Sinica*, vol. 21, no. 1, pp. 13–33, 2011.

[20] L. Chen and J. Z. Huang, "Sparse reduced-rank regression for simultaneous dimension reduction and variable selection," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1533–1545, 2012.

[21] P. T. Reiss and R. T. Ogden, "Functional principal component regression and functional partial least squares," *Journal of the American Statistical Association*, vol. 102, no. 479, pp. 984–996, 2007.

[22] A. J. Izenman, "Reduced-rank regression for the multivariate linear model," *Journal of Multivariate Analysis*, vol. 5, no. 2, pp. 248–264, 1975.

[23] K. R. Gabriel and S. Zamir, "Lower rank approximation of matrices by least squares with any choice of weights," *Technometrics*, vol. 21, pp. 489–498, 1979.

[24] H. Wold and E. Lyttkens, "Nonlinear iterative partial least squares (nipals) estimation procedure," *Bulletin of the International Statistical Institute*, vol. 43, pp. 29–51, 1969.

[25] J. Friedman, R. Tibshirani, and T. Hastie, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.

[26] N. Simon, J. Friedman, R. Tibshirani, and T. Hastie, "Regularization paths for cox's proportional hazards model via coordinate descent," *Journal of Statistical Software*, vol. 39, no. 5, pp. 1–13, 2011.

[27] S. J. Wright, "Coordinate descent algorithms," 2015.

[28] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011.

[29] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[30] T. P. Morris, I. R. White, and M. J. Crowther, "Using simulation studies to evaluate statistical methods," *Statistics in Medicine*, vol. 38, no. 11, 2019.

[31] M. de Wreede, M. Fiocco, and H. C. van Houwelingen, *mstate: Data Preparation for Multistate Models*, 2011. R package version 0.3.1.

[32] ScienceDirect, "Singular value decomposition," n.d. Accessed: 2024-12-28.

[33] M. Hutchings, "Notes on singular value decomposition for math 54," n.d.

Chapter 6

Appendix

6.1 Evolution of ADMM

Two optimization techniques that serve as precursors to the ADMM are presented here. These provide some pertinent context and motivation.

6.1.1 Dual ascent

To begin with, the following equality-constrained optimization problem should be considered:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & h(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x} - \mathbf{b} = 0, \end{aligned} \tag{6.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function. Note that, convexity guarantees that any local minimum is also a global minimum, ensuring convergence to an optimal solution. The *standard Lagrangian function* is defined as:

$$L(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x}) + \boldsymbol{y}h(\boldsymbol{x}), \tag{6.2}$$

where $\boldsymbol{y}$ is called *Lagrange multiplier* or *dual variable*. This standard Lagrangian is used to define the following problem:

$$\text{maximize} \quad g(\boldsymbol{y}), \tag{6.3}$$

where $g(\boldsymbol{y}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{y})$ is the *dual function* and variable $\boldsymbol{y} \in \mathbb{R}^m$. The problem 6.3 is known as the *dual problem* and is used in order to solve the primal problem 6.1, since the optimal values

of both problems are the same. Thus, the primal optimal point $x^\star$ can be calculated from a dual optimal point $\boldsymbol{y}^\star$ as:

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{y}^\star), \tag{6.4}$$

given that there is only one value that minimizes $L(\boldsymbol{x}, \boldsymbol{y}^\star)$ (f strictly convex). It is assumed that g is differentiable and the gradient $\nabla g(\boldsymbol{y})$ is assessed as follows:

$$\nabla g(\boldsymbol{y}) = h(\boldsymbol{x}^+), \tag{6.5}$$

where $\boldsymbol{x}^+ = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{y})$. To solve the dual problem the following updates are iteratively taking place:

$$\boldsymbol{x}^{k+1} := \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{y}^k) \tag{6.6}$$

$$\boldsymbol{y}^{k+1} := \boldsymbol{y}^k + \alpha^k h(\boldsymbol{x}^{k+1}). \tag{6.7}$$

Here, $\alpha^k > 0$ is a step size, and k is the iteration counter. This algorithm is known as *dual ascent*, because of the increase, observed in each step, in the dual function, when α^k is chosen appropriately, i.e., $g(\boldsymbol{y}^{k+1}) > g(\boldsymbol{y}^k)$. Consequently, with appropriate α^k and assumptions held, $\boldsymbol{x}^k$ and $\boldsymbol{y}^k$ converge to an optimal point and an optimal dual point respectively.

6.1.1.1 Dual decomposition

The next term that should be outlined here, is the *dual decomposition*, which is a particular case of the dual ascent method. Its difference compared to the dual ascent algorithm lies in the property that the objective function, f, is 'separable'.

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}_i), \tag{6.8}$$

where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and the variables $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ are subvectors of $\boldsymbol{x}$. Partitioning the matrix $\mathbf{A}$ accordingly as:

$$\mathbf{A} = [\mathbf{A}_1 \cdots \mathbf{A}_N], \tag{6.9}$$

in order for $\mathbf{A}\boldsymbol{x} = \sum_{i=1}^{N} \mathbf{A}_i \boldsymbol{x}_i$ to hold, the Lagrangian can be written as

$$L(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} L_i(\boldsymbol{x}_i, \boldsymbol{y}) = \sum_{i=1}^{N} \left(f_i(\boldsymbol{x}_i) + \boldsymbol{y}^\top (\mathbf{A}_i \boldsymbol{x}_i - (1/N)\boldsymbol{y}^\top \mathbf{b} \right), \tag{6.10}$$

which is also separable in $\boldsymbol{x}$. This means that the first step 6.6 splits into N separable problems that can be solved in parallel. Explicitly, the steps now are:

$$\boldsymbol{x}_i^{k+1} := \arg\min_{\boldsymbol{x}_i} L_i(\boldsymbol{x}_i, \boldsymbol{y}^k) \tag{6.11}$$

$$\boldsymbol{y}^{k+1} := \boldsymbol{y}^k + \alpha^k (\mathbf{A}\boldsymbol{x}^{k+1} - \mathbf{b}). \tag{6.12}$$

The step 6.11 is executed independently, in parallel, for each $i = 1, \ldots, N$. In the step 6.12, the equality constraint residual contributions $\mathbf{A}_i \boldsymbol{x}^{k+1} - \mathbf{b}$ are collected to compute the residual $\mathbf{A}\boldsymbol{x}^{k+1} - \mathbf{b}$. Finally, once the (global) dual variable $\boldsymbol{y}^{k+1}$ is computed, it must be separated to the units that performs the N individual steps in 6.11.

6.1.2 Method of Multipliers

In summary, both dual ascent and dual decomposition are optimization methods that can be applied to complex, large-scale optimization problems. Their aim is to break the problems down into smaller, more manageable sub-problems in order to solve them instead of the larger one and then by combining their solutions to derive the solution to the original problem. They manage to do so by adding new variables and constraints. However, both those methods have strict conditions that need to hold for them to converge (strict convexity or finiteness of f) and for that reason, *Augmented Lagrangian* methods were developed to provide convergence without assumptions and therefore bring robustness to the dual ascent method.

The name of those methods comes from the *Augmented Lagrangian*, which adds an extra quadratic penalty component to the standard Lagrangian function and it is used for solving such problems. The augmented Lagrangian for the problem 6.1 is:

$$L_\rho(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x}) + \left(\boldsymbol{y}h(\boldsymbol{x}) + \frac{\rho}{2}h(\boldsymbol{x})^2\right). \tag{6.13}$$

Here, $\rho > 0$ represents the penalty parameter, that controls the weight of the additional quadratic terms. Bigger ρ values correspond to more heavily penalized constraint violations. So, applying the dual ascent method to the new problem the following steps are iteratively taking place:

$$\boldsymbol{x}^{k+1} := \arg\min_{\boldsymbol{x}} L_\rho(\boldsymbol{x}, \boldsymbol{y}^k) \tag{6.14}$$

$$\boldsymbol{y}^{k+1} := \boldsymbol{y}^k + \rho h(\boldsymbol{x}^{k+1}). \tag{6.15}$$

This is known as the *method of multipliers* for solving 6.1 and it converges under less strict conditions than the dual ascent, like when f takes infinite values or is not strictly convex. Nevertheless, for this method when f is separable, the augmented Lagrangian is not, so the x-minimization step 6.14 cannot be executed independently in parallel for each x_i and consequently the basic method of multipliers is not appropriate for decomposition.

This is where the ADMM becomes relevant. Combining the advantages of the dual ascent and the method of multipliers is particularly beneficial for solving large-scale optimization problems, especially those that can be divided into smaller sub-problems that can be solved independently and in parallel without the strict conditions that the dual ascent method relies on.

6.2 Tables and Figures

6.2.1 Simulation study 1

6.2.1.1 Data generation

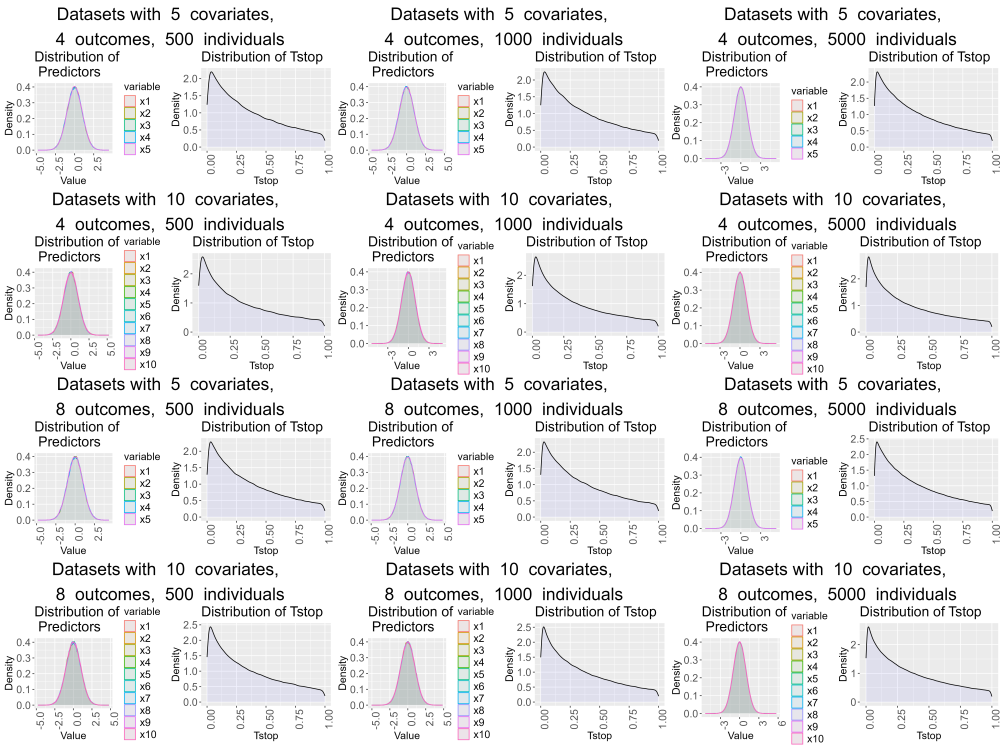

Figure A1: Each sub-figure represents the distributions of predictors(left) and Tstop(right) for simulation study 1. 200 datasets are simulated per scenario using 'case 1' way of simulating and rank 1. Each subfigure's title includes the number of predictors(p), outcomes(k), and sample size(n) for the datasets.

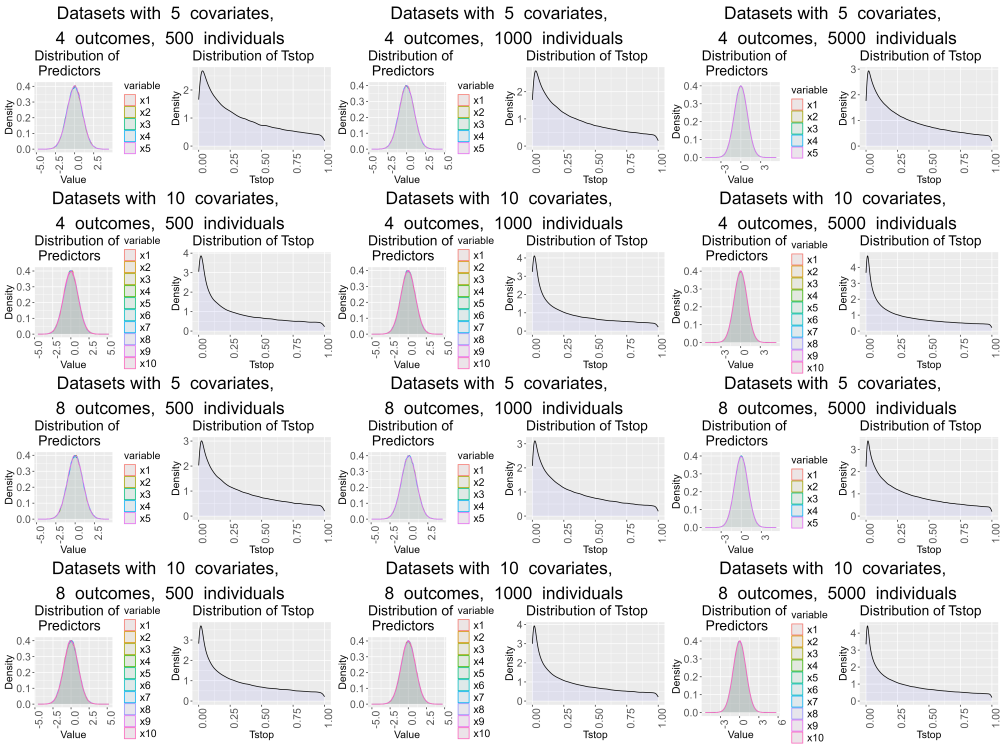

Figure A2: Each sub-figure represents the distributions of predictors(left) and Tstop(right) for simulation study 1. 200 datasets are simulated per scenario using 'case 1' way of simulating and rank 2. Each subfigure's title includes the number of predictors(p), outcomes(k), and sample size(n) for the datasets.

6.2.1.2 Performance values

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_MSE	MC_error_MSE
500	5	4	1	1	2.4282e-03	6.8207e-05
500	5	4	1	2	4.7430e-03	1.2228e-04
500	5	4	2	1	1.0894e-02	3.6128e-04
500	5	4	2	2	4.6933e-03	1.2313e-04
500	5	4	2	3	5.9701e-03	1.4230e-04
500	5	8	1	1	2.0197e-03	3.9821e-05
500	5	8	1	2	4.0875e-03	7.6524e-05
500	5	8	2	1	8.0441e-03	1.5239e-04
500	5	8	2	2	4.2829e-03	8.1016e-05
500	5	8	2	3	5.6507e-03	1.0038e-04
500	10	4	1	1	2.2428e-03	4.4491e-05
500	10	4	1	2	4.4985e-03	8.1045e-05
500	10	4	2	1	4.7346e-03	7.6789e-05
500	10	4	2	2	5.2637e-03	9.1478e-05
500	10	4	2	3	6.5675e-03	1.1018e-04
500	10	8	1	1	1.4054e-03	2.1827e-05
500	10	8	1	2	3.1555e-03	4.5864e-05
500	10	8	2	1	1.2932e-02	1.4693e-04
500	10	8	2	2	3.3008e-03	4.2159e-05
500	10	8	2	3	4.7711e-03	6.0067e-05

Table A1: Simulation study 1, average MSE and Monte Carlo error for sample space 500 and different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_MSE	MC_error_MSE
1000	5	4	1	1	2.9170e-05	1.0905e-03
1000	5	4	1	2	6.0545e-05	2.2488e-03
1000	5	4	2	1	3.2382e-04	9.2400e-03
1000	5	4	2	2	5.1807e-05	2.0954e-03
1000	5	4	2	3	6.3163e-05	2.7094e-03
1000	5	8	1	1	1.7306e-05	9.3708e-04
1000	5	8	1	2	3.4911e-05	1.9234e-03
1000	5	8	2	1	1.3423e-04	6.5466e-03
1000	5	8	2	2	3.5837e-05	1.9731e-03
1000	5	8	2	3	4.5047e-05	2.6696e-03
1000	10	4	1	1	2.0020e-05	1.0120e-03
1000	10	4	1	2	3.5773e-05	1.9966e-03
1000	10	4	2	1	4.9986e-05	3.0838e-03
1000	10	4	2	2	3.9740e-05	2.2860e-03
1000	10	4	2	3	4.8142e-05	2.9056e-03
1000	10	8	1	1	1.0375e-05	6.8188e-04
1000	10	8	1	2	2.1180e-05	1.5042e-03
1000	10	8	2	1	1.3806e-04	1.1741e-02
1000	10	8	2	2	1.9663e-05	1.5627e-03
1000	10	8	2	3	2.7862e-05	2.2164e-03

Table A2: Simulation study 1, average MSE and Monte Carlo error for sample space 1000 and different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_MSE	MC_error_MSE
5000	5	4	1	1	5.7626e-06	2.2283e-04
5000	5	4	1	2	1.1045e-05	4.4265e-04
5000	5	4	2	1	3.0865e-04	8.2881e-03
5000	5	4	2	2	1.0548e-05	4.1000e-04
5000	5	4	2	3	1.2669e-05	5.3324e-04
5000	5	8	1	1	3.5505e-06	1.9061e-04
5000	5	8	1	2	6.7261e-06	3.8128e-04
5000	5	8	2	1	1.1874e-04	5.5395e-03
5000	5	8	2	2	7.2517e-06	3.9279e-04
5000	5	8	2	3	9.1065e-06	5.3369e-04
5000	10	4	1	1	3.3869e-06	1.9419e-04
5000	10	4	1	2	6.9353e-06	3.8683e-04
5000	10	4	2	1	3.3627e-05	1.9945e-03
5000	10	4	2	2	7.1131e-06	4.2561e-04
5000	10	4	2	3	8.8454e-06	5.5978e-04
5000	10	8	1	1	2.0319e-06	1.3557e-04
5000	10	8	1	2	4.0103e-06	2.9357e-04
5000	10	8	2	1	1.3444e-04	1.1054e-02
5000	10	8	2	2	3.4354e-06	2.9121e-04
5000	10	8	2	3	4.9901e-06	4.1495e-04

Table A3: Simulation study 1, average MSE and Monte Carlo error for sample space 5000 rounded to 4 decimals for different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_absolute_bias	MC_error_absolute_bias
500	5	4	1	1	7.7792e-04	3.7088e-02
500	5	4	1	2	1.0874e-03	5.3005e-02
500	5	4	2	1	1.6477e-03	7.5419e-02
500	5	4	2	2	1.0787e-03	5.3091e-02
500	5	4	2	3	1.2159e-03	6.0849e-02
500	5	8	1	1	5.0203e-04	3.4009e-02
500	5	8	1	2	7.1398e-04	4.9195e-02
500	5	8	2	1	1.0027e-03	6.8053e-02
500	5	8	2	2	7.2909e-04	5.0649e-02
500	5	8	2	3	8.3652e-04	5.9064e-02
500	10	4	1	1	5.2692e-04	3.6310e-02
500	10	4	1	2	7.4513e-04	5.1723e-02
500	10	4	2	1	7.6885e-04	5.4529e-02
500	10	4	2	2	7.9666e-04	5.6683e-02
500	10	4	2	3	8.8360e-04	6.3688e-02
500	10	8	1	1	4.3060e-04	2.7444e-02
500	10	8	1	2	6.1939e-04	4.0669e-02
500	10	8	2	1	1.1432e-03	6.7064e-02
500	10	8	2	2	6.5215e-04	3.9625e-02
500	10	8	2	3	7.3402e-04	4.5648e-02

Table A4: Simulation study 1, average absolute bias and Monte Carlo error for sample space 500 rounded to 4 decimals for different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_absolute_bias	MC_error_absolute_bias
1000	5	4	1	1	5.2132e-04	2.5304e-02
1000	5	4	1	2	7.4888e-04	3.6407e-02
1000	5	4	2	1	1.5158e-03	6.7271e-02
1000	5	4	2	2	7.2122e-04	3.5768e-02
1000	5	4	2	3	8.2004e-04	4.1239e-02
1000	5	8	1	1	3.4224e-04	2.3488e-02
1000	5	8	1	2	4.9023e-04	3.3889e-02
1000	5	8	2	1	9.0398e-04	5.9397e-02
1000	5	8	2	2	4.9596e-04	3.4319e-02
1000	5	8	2	3	5.7653e-04	4.0632e-02
1000	10	4	1	1	3.5517e-04	2.4464e-02
1000	10	4	1	2	4.9837e-04	3.4673e-02
1000	10	4	2	1	6.2034e-04	4.3605e-02
1000	10	4	2	2	5.3044e-04	3.7474e-02
1000	10	4	2	3	5.9602e-04	4.2563e-02
1000	10	8	1	1	3.0515e-04	2.8972e-02
1000	10	8	1	2	4.3926e-04	4.1390e-02
1000	10	8	2	1	7.5488e-04	5.7152e-02
1000	10	8	2	2	4.4622e-04	4.0148e-02
1000	10	8	2	3	5.1527e-04	4.4452e-02

Table A5: Simulation study 1, average absolute bias and Monte Carlo error for sample space 1000 rounded to 4 decimals for different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_absolute_bias	MC_error_absolute_bias
5000	5	4	1	1	2.3602e-04	1.1382e-02
5000	5	4	1	2	3.3266e-04	1.6301e-02
5000	5	4	2	1	1.4316e-03	6.1369e-02
5000	5	4	2	2	3.2001e-04	1.5816e-02
5000	5	4	2	3	3.6495e-04	1.8222e-02
5000	5	8	1	1	1.5435e-04	1.0543e-02
5000	5	8	1	2	2.1832e-04	1.5148e-02
5000	5	8	2	1	8.3062e-04	5.2016e-02
5000	5	8	2	2	2.2159e-04	1.5311e-02
5000	5	8	2	3	2.5647e-04	1.8173e-02
5000	10	4	1	1	1.5576e-04	1.0819e-02
5000	10	4	1	2	2.1980e-04	1.5217e-02
5000	10	4	2	1	4.9519e-04	3.3749e-02
5000	10	4	2	2	2.3061e-04	1.6233e-02
5000	10	4	2	3	2.6435e-04	1.8819e-02
5000	10	8	1	1	1.0616e-04	1.0386e-02
5000	10	8	1	2	1.5169e-04	1.4456e-02
5000	10	8	2	1	3.0372e-04	2.7863e-02
5000	10	8	2	2	1.5438e-04	1.4364e-02
5000	10	8	2	3	1.7987e-04	1.6697e-02

Table A6: Simulation study 1, average absolute bias and Monte Carlo error for sample space 5000 rounded to 4 decimals for different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_LPE	MC_error_LPE
500	5	4	1	1	1.1992e-02	5.5965e-05
500	5	4	1	2	2.3308e-02	9.0005e-05
500	5	4	2	1	5.4342e-02	2.0915e-04
500	5	4	2	2	2.3245e-02	9.1874e-05
500	5	4	2	3	2.9485e-02	1.0576e-04
500	5	8	1	1	1.0070e-02	4.3350e-05
500	5	8	1	2	2.0308e-02	7.0868e-05
500	5	8	2	1	4.0161e-02	1.4021e-04
500	5	8	2	2	2.1349e-02	7.6295e-05
500	5	8	2	3	2.8094e-02	8.9564e-05
500	10	4	1	1	2.2156e-02	1.0052e-04
500	10	4	1	2	4.4077e-02	1.5709e-04
500	10	4	2	1	4.7184e-02	1.7186e-04
500	10	4	2	2	5.2116e-02	1.9579e-04
500	10	4	2	3	6.4809e-02	2.2392e-04
500	10	8	1	1	1.3805e-02	5.6915e-05
500	10	8	1	2	3.0731e-02	1.0266e-04
500	10	8	2	1	1.2778e-01	4.7593e-04
500	10	8	2	2	3.2511e-02	1.1465e-04
500	10	8	2	3	4.6923e-02	1.4305e-04

Table A7: Simulation study 1, average linear predictor error (LPE) and Monte Carlo error for sample space 500 rounded to 4 decimals for different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_LPE	MC_error_LPE
1000	5	4	1	1	5.4348e-03	1.7399e-05
1000	5	4	1	2	1.1181e-02	3.0053e-05
1000	5	4	2	1	4.6171e-02	1.3144e-04
1000	5	4	2	2	1.0433e-02	2.7659e-05
1000	5	4	2	3	1.3469e-02	3.3238e-05
1000	5	8	1	1	4.6605e-03	1.3802e-05
1000	5	8	1	2	9.5708e-03	2.3167e-05
1000	5	8	2	1	3.2581e-02	8.6856e-05
1000	5	8	2	2	9.7921e-03	2.4426e-05
1000	5	8	2	3	1.3253e-02	2.9265e-05
1000	10	4	1	1	9.9993e-03	3.0501e-05
1000	10	4	1	2	1.9679e-02	4.7908e-05
1000	10	4	2	1	3.0643e-02	7.4769e-05
1000	10	4	2	2	2.2630e-02	5.8326e-05
1000	10	4	2	3	2.8709e-02	6.6000e-05
1000	10	8	1	1	6.8010e-03	1.9087e-05
1000	10	8	1	2	1.4941e-02	3.4862e-05
1000	10	8	2	1	1.1675e-01	3.2365e-04
1000	10	8	2	2	1.5541e-02	3.8050e-05
1000	10	8	2	3	2.1953e-02	4.6526e-05

Table A8: Simulation study 1, average linear predictor error (LPE) and Monte Carlo error for sample space 1000 rounded to 4 decimals for different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Average_LPE	MC_error_LPE
5000	5	4	1	1	1.1122e-03	1.5652e-06
5000	5	4	1	2	2.2105e-03	2.5989e-06
5000	5	4	2	1	4.1441e-02	5.6392e-05
5000	5	4	2	2	2.0459e-03	2.4065e-06
5000	5	4	2	3	2.6610e-03	2.8856e-06
5000	5	8	1	1	9.5506e-04	1.2702e-06
5000	5	8	1	2	1.9092e-03	2.0615e-06
5000	5	8	2	1	2.7708e-02	3.6955e-05
5000	5	8	2	2	1.9675e-03	2.2501e-06
5000	5	8	2	3	2.6720e-03	2.6883e-06
5000	10	4	1	1	1.9398e-03	2.5235e-06
5000	10	4	1	2	3.8656e-03	4.2136e-06
5000	10	4	2	1	1.9934e-02	2.4460e-05
5000	10	4	2	2	4.2517e-03	4.8697e-06
5000	10	4	2	3	5.5920e-03	5.6702e-06
5000	10	8	1	1	1.3541e-03	1.7135e-06
5000	10	8	1	2	2.9332e-03	3.0433e-06
5000	10	8	2	1	1.1056e-01	1.4364e-04
5000	10	8	2	2	2.9102e-03	3.1263e-06
5000	10	8	2	3	4.1468e-03	3.8734e-06

Table A9: Simulation study 1, average linear predictor error (LPE) and Monte Carlo error for sample space 5000 rounded to 4 decimals for different combinations of parameters. All scenarios consist of 200 simulated datasets (S), simulated using the 'case 1' way of simulation and zero correlation.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Mean B Corr	Sd B Corr	Mean ZB Corr	Sd ZB Corr
500	5	4	1	1	9.8614e-01	8.8712e-03	9.9427e-01	3.3726e-03
500	5	4	1	2	9.6989e-01	1.2529e-02	9.8811e-01	4.7855e-03
500	5	4	2	1	9.2728e-01	1.0079e-02	9.8786e-01	1.9573e-03
500	5	4	2	2	9.7501e-01	1.0446e-02	9.9539e-01	1.8779e-03
500	5	4	2	3	9.6731e-01	1.1228e-02	9.9404e-01	1.9352e-03
500	5	8	1	1	9.8605e-01	6.1991e-03	9.9560e-01	1.8500e-03
500	5	8	1	2	9.6999e-01	8.3059e-03	9.9074e-01	2.4600e-03
500	5	8	2	1	9.7128e-01	4.3379e-03	9.9366e-01	1.0463e-03
500	5	8	2	2	9.8644e-01	4.4343e-03	9.9689e-01	9.6285e-04
500	5	8	2	3	9.8185e-01	4.9201e-03	9.9586e-01	1.0700e-03
500	10	4	1	1	9.8319e-01	7.5400e-03	9.9542e-01	1.9840e-03
500	10	4	1	2	9.6315e-01	1.0814e-02	9.9031e-01	2.7201e-03
500	10	4	2	1	9.3985e-01	1.6876e-02	9.9682e-01	9.6845e-04
500	10	4	2	2	9.4120e-01	2.0869e-02	9.9673e-01	1.1511e-03
500	10	4	2	3	9.2819e-01	2.0427e-02	9.9597e-01	1.1503e-03
500	10	8	1	1	9.9068e-01	3.8244e-03	9.9574e-01	1.7559e-03
500	10	8	1	2	9.7782e-01	5.3863e-03	9.9013e-01	2.4809e-03
500	10	8	2	1	9.1607e-01	7.4720e-03	9.8864e-01	1.4783e-03
500	10	8	2	2	9.8196e-01	5.9033e-03	9.9741e-01	8.5396e-04
500	10	8	2	3	9.7386e-01	6.4424e-03	9.9626e-01	9.1412e-04

Table A10: Simulation study 1, the mean and standard deviation of (200×1) vector with correlation between true $\mathbf{B}$ matrix and estimated $\hat{\mathbf{B}}_\mathbf{s}$ matrices after vectorization, for 200 simulations (S) using 'case 1' and zero correlation, 5000 subjects and different combinations of parameters. Same metrics for vectorized linear predictor matrices (true: $\mathbf{Z_s B}$ and estimated: $\mathbf{Z_s}\hat{\mathbf{B}}_\mathbf{s}$). All are rounded to 4 decimals. The s symbolizes the simulation run ($s = 1, 2, .., 200$).

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Mean B Corr	Sd B Corr	Mean ZB Corr	Sd ZB Corr
1000	5	4	1	1	9.9412e-01	3.2362e-03	9.9744e-01	1.3751e-03
1000	5	4	1	2	9.8562e-01	5.4326e-03	9.9427e-01	2.1239e-03
1000	5	4	2	1	9.3611e-01	4.7323e-03	9.8954e-01	1.0358e-03
1000	5	4	2	2	9.8824e-01	4.6921e-03	9.9789e-01	8.4137e-04
1000	5	4	2	3	9.8430e-01	5.3586e-03	9.9721e-01	9.5110e-04
1000	5	8	1	1	9.9339e-01	2.8734e-03	9.9791e-01	8.6687e-04
1000	5	8	1	2	9.8538e-01	3.7317e-03	9.9550e-01	1.1648e-03
1000	5	8	2	1	9.7604e-01	2.0281e-03	9.9475e-01	5.4901e-04
1000	5	8	2	2	9.9366e-01	2.1795e-03	9.9855e-01	4.8637e-04
1000	5	8	2	3	9.9119e-01	2.3830e-03	9.9800e-01	5.3633e-04
1000	10	4	1	1	9.9192e-01	3.2609e-03	9.9783e-01	8.6552e-04
1000	10	4	1	2	9.8238e-01	4.3062e-03	9.9544e-01	1.0924e-03
1000	10	4	2	1	9.5794e-01	8.3055e-03	9.9786e-01	4.3741e-04
1000	10	4	2	2	9.7237e-01	1.1287e-02	9.9852e-01	5.7696e-04
1000	10	4	2	3	9.6454e-01	1.0940e-02	9.9810e-01	5.6812e-04
1000	10	8	1	1	9.9532e-01	1.8359e-03	9.9786e-01	8.2674e-04
1000	10	8	1	2	9.8897e-01	2.5148e-03	9.9510e-01	1.1188e-03
1000	10	8	2	1	9.2324e-01	4.0530e-03	9.8977e-01	7.9115e-04
1000	10	8	2	2	9.9105e-01	2.9222e-03	9.9874e-01	4.0729e-04
1000	10	8	2	3	9.8704e-01	3.0564e-03	9.9820e-01	4.2057e-04

Table A11: Simulation study 1, the mean and standard deviation of (200×1) vector with correlation between true $\mathbf{B}$ matrix and estimated $\hat{\mathbf{B}}_\mathbf{s}$ matrices after vectorization, for 200 simulations (S) using 'case 1' and zero correlation, 1000 subjects and different combinations of parameters. Same metrics for vectorized linear predictor matrices (true: $\mathbf{Z_sB}$ and estimated: $\mathbf{Z_s}\hat{\mathbf{B}}_\mathbf{s}$). All are rounded to 4 decimals. The s symbolizes the simulation run $(s = 1, 2, .., 200)$.

Sample_size	Covariates	Outcomes	Ranks_simulate	Ranks_fit	Mean B Corr	Sd B Corr	Mean ZB Corr	Sd ZB Corr
5000	5	4	1	1	9.9871e-01	7.3801e-04	9.9947e-01	2.8197e-04
5000	5	4	1	2	9.9704e-01	1.0983e-03	9.9884e-01	4.1401e-04
5000	5	4	2	1	9.4183e-01	1.2953e-03	9.9053e-01	3.5019e-04
5000	5	4	2	2	9.9764e-01	9.7656e-04	9.9957e-01	1.7077e-04
5000	5	4	2	3	9.9680e-01	1.1404e-03	9.9943e-01	1.9575e-04
5000	5	8	1	1	9.9867e-01	5.6904e-04	9.9957e-01	1.7282e-04
5000	5	8	1	2	9.9705e-01	7.3607e-04	9.9910e-01	2.1993e-04
5000	5	8	2	1	9.7928e-01	5.5489e-04	9.9549e-01	1.7604e-04
5000	5	8	2	2	9.9870e-01	4.7285e-04	9.9970e-01	1.0368e-04
5000	5	8	2	3	9.9817e-01	5.0690e-04	9.9959e-01	1.1012e-04
5000	10	4	1	1	9.9840e-01	6.1543e-04	9.9956e-01	1.6496e-04
5000	10	4	1	2	9.9647e-01	8.8201e-04	9.9908e-01	2.3553e-04
5000	10	4	2	1	9.7107e-01	2.1947e-03	9.9855e-01	1.1394e-04
5000	10	4	2	2	9.9471e-01	1.8069e-03	9.9972e-01	9.4791e-05
5000	10	4	2	3	9.9275e-01	2.0010e-03	9.9962e-01	1.0198e-04
5000	10	8	1	1	9.9908e-01	3.4477e-04	9.9958e-01	1.5349e-04
5000	10	8	1	2	9.9782e-01	4.5056e-04	9.9903e-01	1.9721e-04
5000	10	8	2	1	9.2762e-01	1.4182e-03	9.9039e-01	3.2721e-04
5000	10	8	2	2	9.9833e-01	4.9856e-04	9.9976e-01	6.9829e-05
5000	10	8	2	3	9.9752e-01	5.4668e-04	9.9965e-01	7.6244e-05

Table A12: Simulation study 1, the mean and standard deviation of (200×1) vector with correlation between true $\mathbf{B}$ matrix and estimated $\hat{\mathbf{B}}_\mathbf{s}$ matrices after vectorization, for 200 simulations (S) using 'case 1' and zero correlation, 5000 subjects and different combinations of parameters. Same metrics for vectorized linear predictor matrices (true: $\mathbf{Z_sB}$ and estimated: $\mathbf{Z_s}\hat{\mathbf{B}}_\mathbf{s}$). All are rounded to 4 decimals. The s symbolizes the simulation run $(s = 1, 2, .., 200)$.

6.2.1.3 Heatmaps of true B and estimated $\hat{B}$

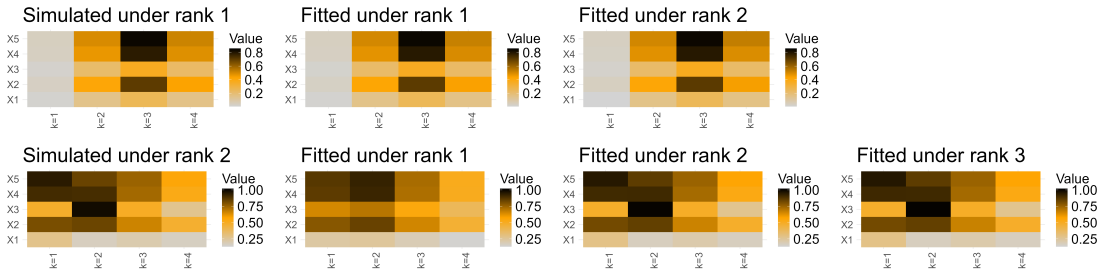

Figure A3: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 500 samples (n), 5 predictors (p), and 4 outcomes (K).

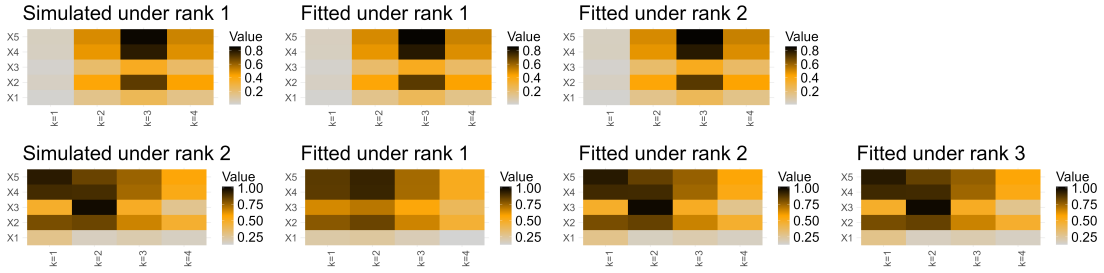

Figure A4: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 1000 samples, 5 predictors (p), and 4 outcomes (K).

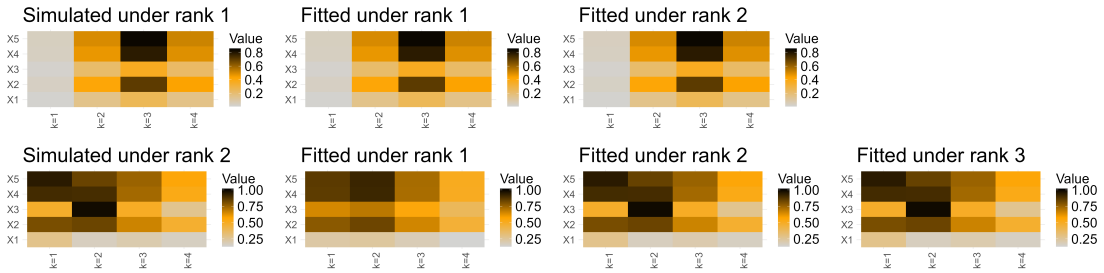

Figure A5: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 5000 samples (n), 5 predictors (p), and 4 outcomes (K).

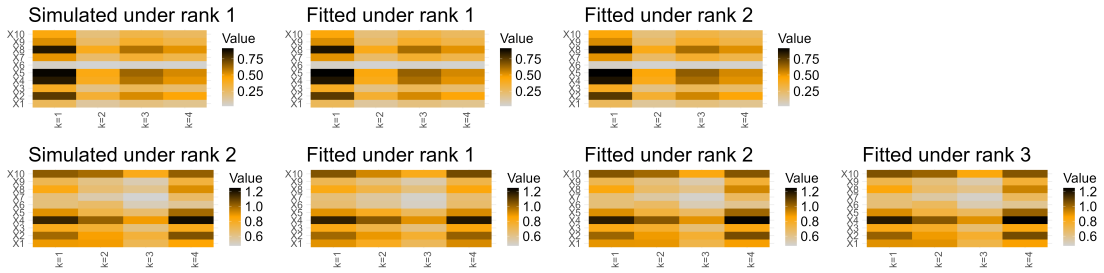

Figure A6: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 500 samples (n), 10 predictors (p), and 4 outcomes (K).

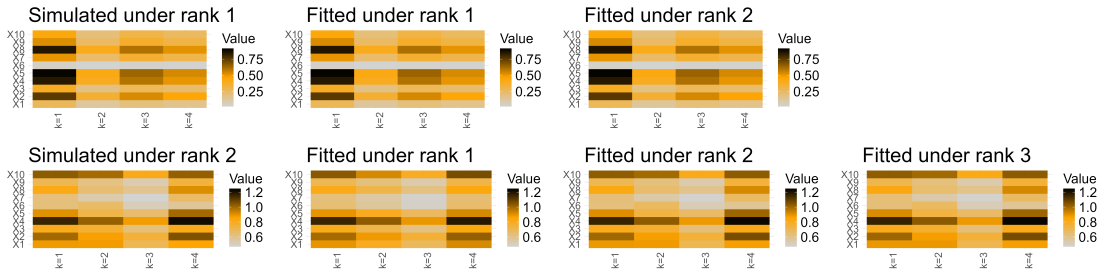

Figure A7: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 1000 samples (n), 10 predictors (p), and 4 outcomes (K).

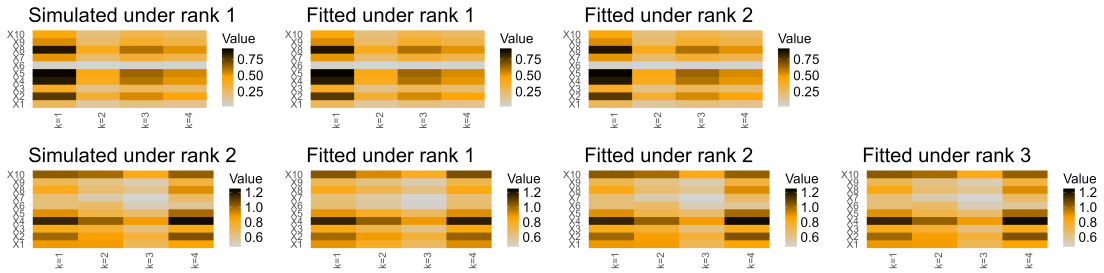

Figure A8: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 5000 samples (n), 10 predictors (p), and 4 outcomes (K).

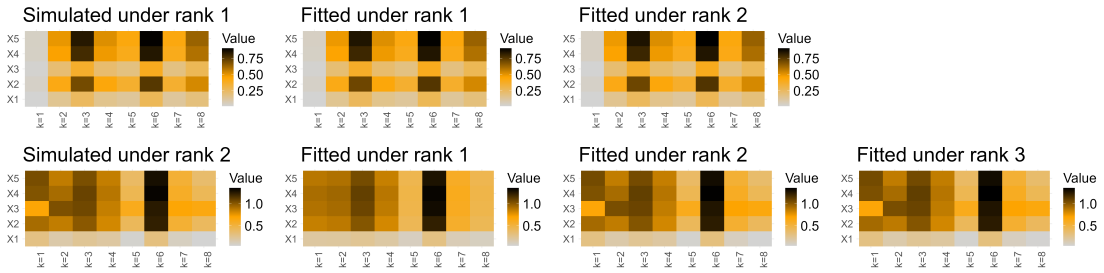

Figure A9: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 500 samples (n), 5 predictors (p), and 8 outcomes (K).

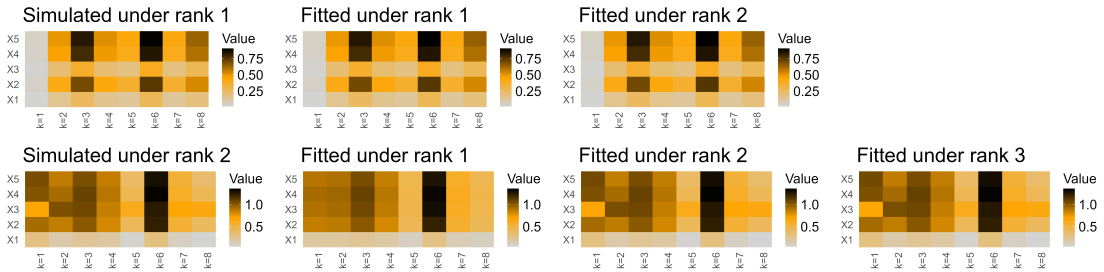

Figure A10: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 1000 samples (n), 5 predictors (p), and 8 outcomes (K).

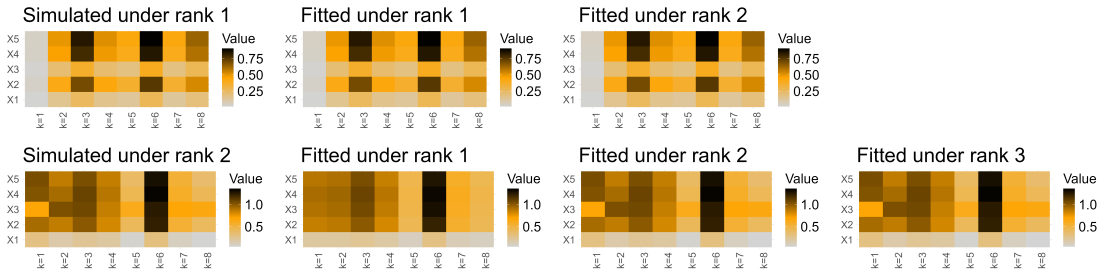

Figure A11: The leftmost heatmap of the first row represents the simulated under rank 1, $\mathbf{B}$ matrix, and the one of second line the simulated under rank 2, $\mathbf{B}$ matrix. The matrices next to them represent the mean of all the estimated $\mathbf{B}$ matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 5000 samples (n), 5 predictors (p), and 8 outcomes (K).

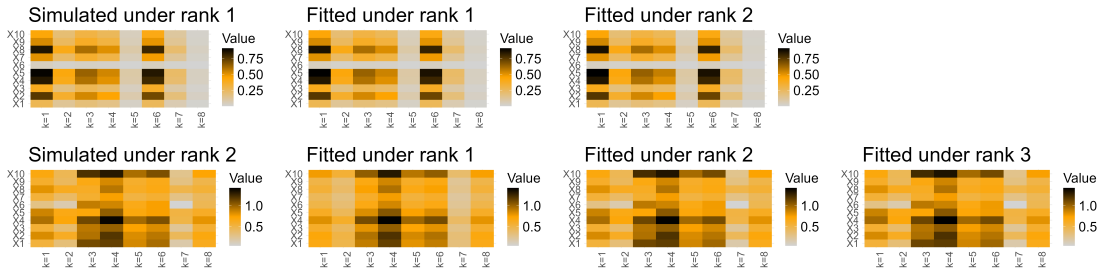

Figure A12: The leftmost heatmap of the first row represents the simulated under rank 1, $\mathbf{B}$ matrix, and the one of second line the simulated under rank 2, $\mathbf{B}$ matrix. The matrices next to them represent the mean of all the estimated $\mathbf{B}$ matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 500 samples (n), 10 predictors (p), and 8 outcomes (K).

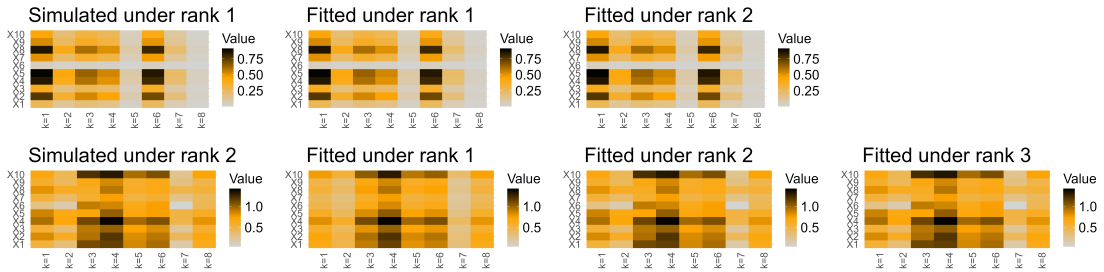

Figure A13: The leftmost heatmap of the first row represents the simulated under rank 1, $\mathbf{B}$ matrix, and the one of second line the simulated under rank 2, $\mathbf{B}$ matrix. The matrices next to them represent the mean of all the estimated $\mathbf{B}$ matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 1000 samples (n), 10 predictors (p,) and 8 outcomes (K).

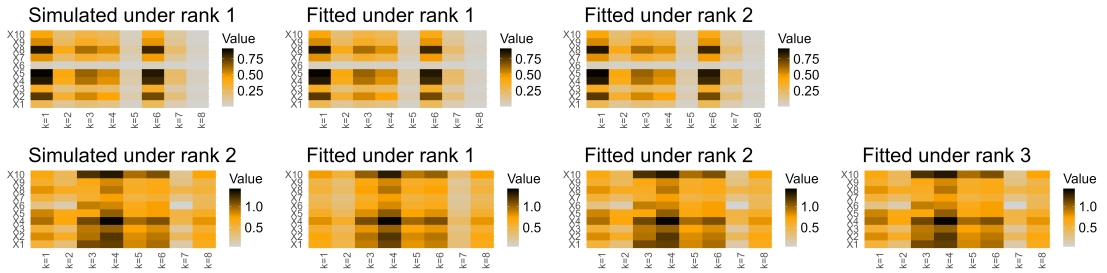

Figure A14: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 200 simulated datasets (S) with 5000 samples (n), 10 predictors (p), and 8 outcomes (K).

6.2.2 Simulation study 2

6.2.2.1 Data generation

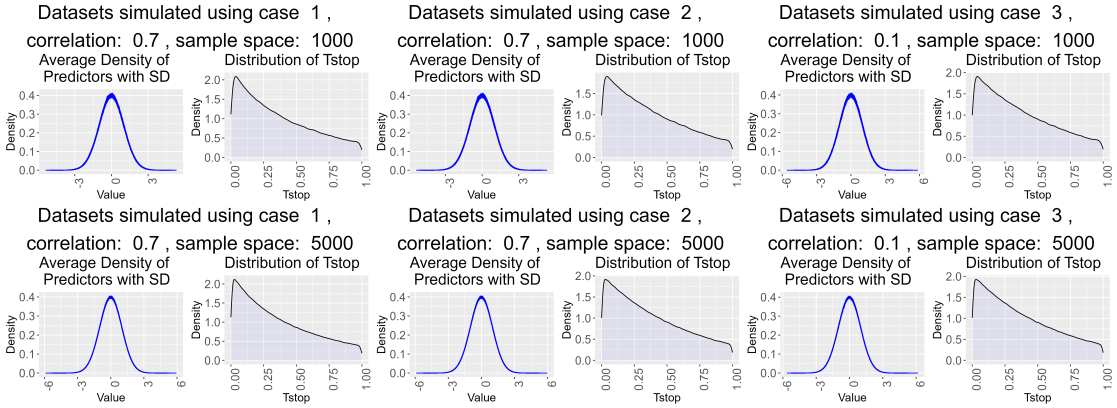

Figure A15: Each sub-figure represents the average distribution of predictors with standard deviation, which is minor(left) and Tstop(right) for simulation study 2. 100 datasets are simulated per scenario using 300 predictors (p), 8 outcomes (K) and rank (R) 1. Each subfigure's title includes how each dataset is simulated (case), correlation, and sample size (n) for the datasets.

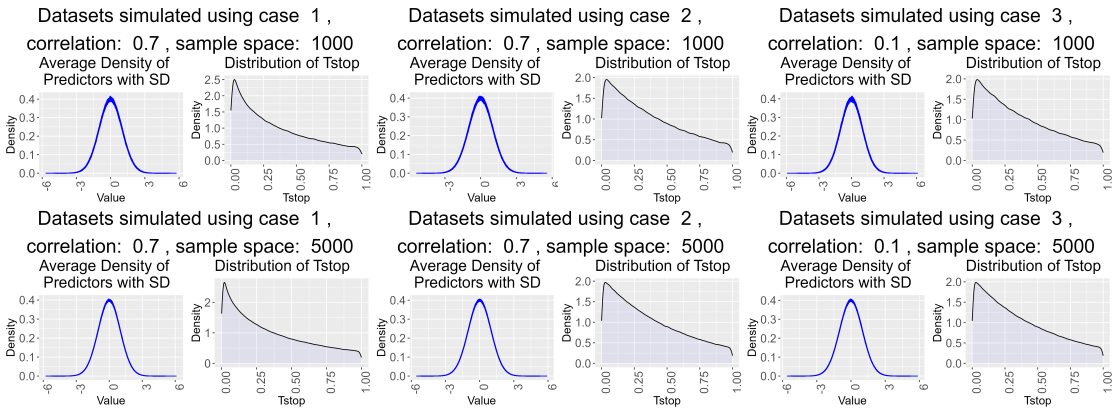

Figure A16: Each sub-figure represents the average distribution of predictors with standard deviation, which is minor(left) and Tstop(right) for simulation study 2. 100 datasets are simulated per scenario using 300 predictors (p), 8 outcomes (K) and rank (R) 2. Each subfigure's title includes how each dataset is simulated (case), correlation, and sample size (n) for the datasets.

6.2.2.2 Plots used to choose λ to fit models

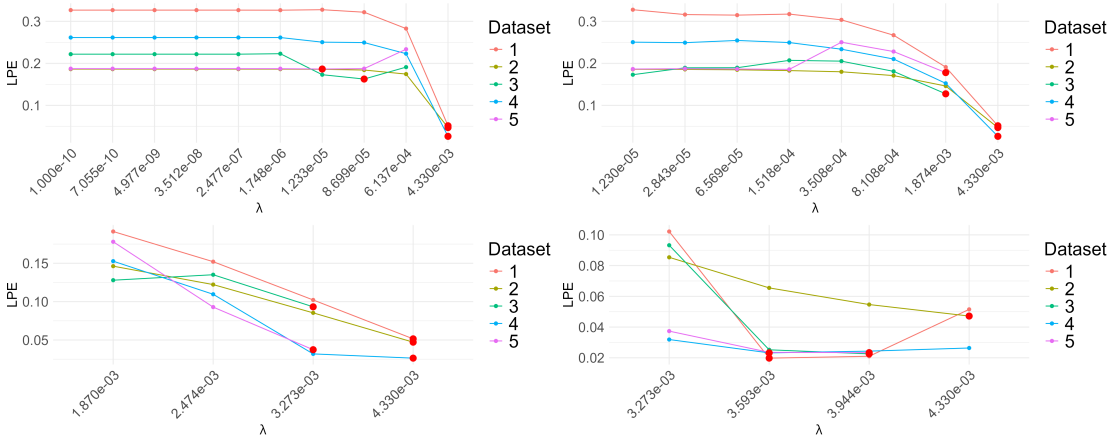

Figure A17: The figure aims to illustrate how the zooms are performed using the initial plot of 10 log-spaced λ values to choose the best possible value for it, for each of the different scenarios considered in simulation 2. Three rules are taken into account. **First**, if after performing two additional zooms in a plot, the resulting chosen λ can not fit all the 5 datasets, one additional zoom is performed. **Second**, the same number of zooms should be performed for datasets simulated within the same scenario and rank to have fair results. For example, if in a scenario, datasets simulated using rank 1 require based on the prior rule an additional zoom, the additional zoom should also be performed for the model fitted with ranks 2 and 3, even if it is not required solely based on the derived plots and the first rule. **Third**, if the optimal median λ (after all zooms) fails to fit any of the 5 datasets, the second-best median λ (calculated by excluding the optimal) is chosen instead.
In all the plots, each line corresponds to one of the five different simulated datasets, used to fit the models. The x-axis represents the values of λ for which the models are fitted, and the y-axis the LPE. The red dots represent the (λ, lowest LPE for specific dataset) points. For each scenario, ten values of λ are initially fitted as in the upper-left plot. Because in this plot the smallest (1.233e-05) and the biggest (4.330e-03) best λ values are separated by 3 positions in the sequence of log-spaced λ values, the new vector with log-spaced values of λ to fit is chosen to have $3 \times 2 = 6$ values between the 2 best λ values. Then this second vector of 8 ($6 + 2$) values in total is fitted and results in the upper-right plot. Because in this plot there are 2 best λ values (1.874e-04 and 4.333e-03) that are apart by 1 step, the new vector of values to be fitted is chosen to have $1 \times 2 = 2$ values between the best λ values and resulted in the bottom-left plot. Since, here the third zoom plot (bottom-left) has 4.33e-03 as the median best value for λ, which can not fit all five datasets because of error, another final zoom is performed again using the same logic described before (first rule). Finally, after performing all the zooms the median best value out of all the 5 best values of λ is chosen for each of the zoom plots (e.g. in the illustrated example, the median best value resulted from both the upper-right and the bottom-left plot is 4.33e-03, but since it should be excluded-third rule- the two median best values are 1.874e-03 and 3.273e-03 respectively. The bottom-right plot's median best value is 3.944e-03). Then the average LPE across all 5 datasets is calculated for each of the zoom plots for their respective median best value. So, in this example, 3 values (one for each zoom plot) of average LPE are obtained. The λ that achieves the smallest LPE (not always the last) is the one chosen to fit the rest 95 datasets.

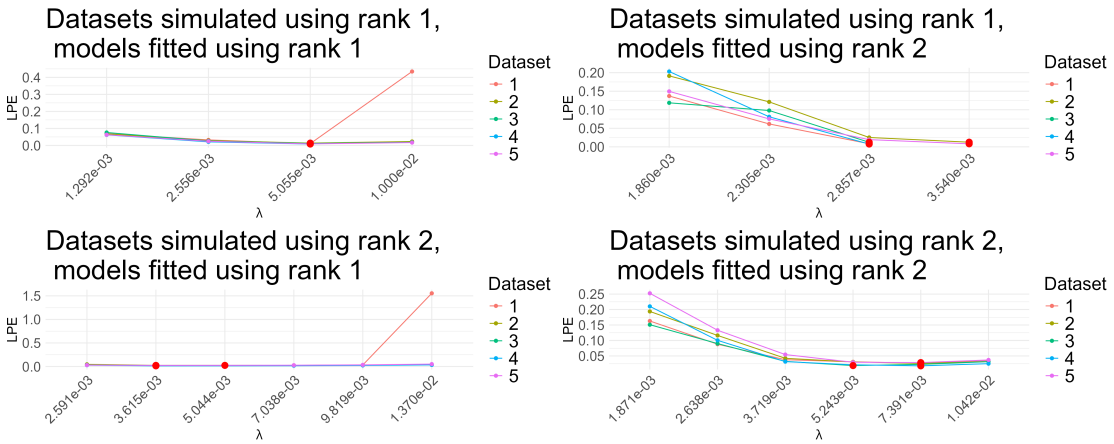

Figure A18: The figure represents the plots based on which the different λ values for fitting the different models are chosen. Each line represents one of the five different simulated datasets, which are used to fit the models, the x-axis the different values of λ for which the models of different ranks (1 or 2) are fitted, and the y-axis the LPE value. The red dots represent the λ for which each of the five datasets achieved the lowest LPE. The median of those 5 λ values is chosen as the best λ to fit all the 100 datasets. The datasets used for this figure are simulated using the 'case 1' way of simulation, correlation equal to 0.7, 1000 samples (n), 300 predictors (p), and 8 outcomes (K).

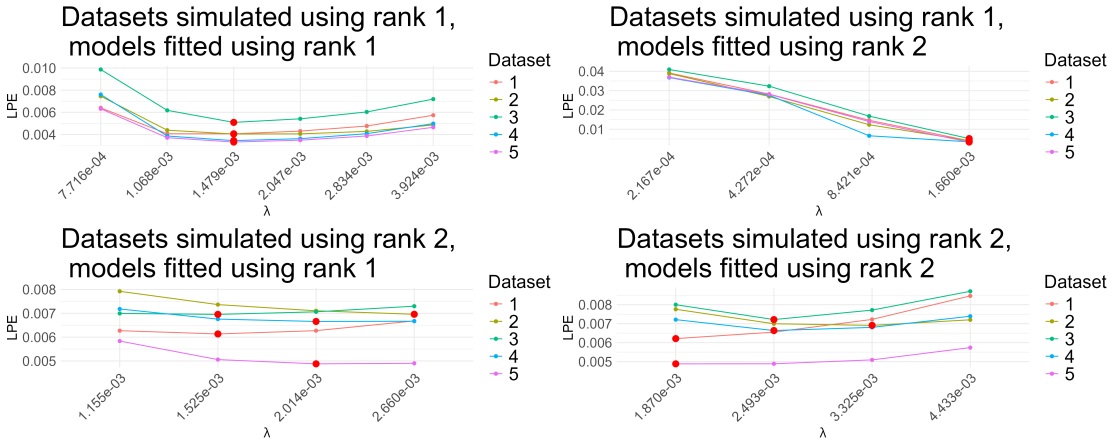

Figure A19: The figure represents the plots based on which the different λ values for fitting the different models are chosen. Each line represents one of the five different simulated datasets, which are used to fit the models, the x-axis the different values of λ for which the models of different ranks (1 or 2) are fitted, and the y-axis the LPE value. The red dots represent the λ for which each of the five datasets achieved the lowest LPE. The median of those 5 λ values is chosen as the best λ to fit all the 100 datasets. The datasets used for this figure are simulated using the 'case 1' way of simulation, correlation equal to 0.7, 5000 samples (n), 300 predictors (p), and 8 outcomes (K).

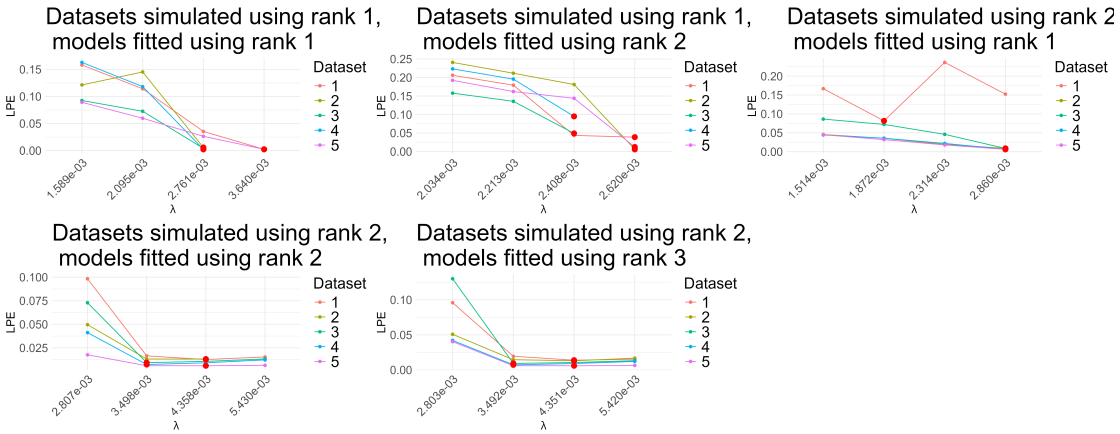

Figure A20: The figure represents the plots based on which the different λ values for fitting the different models are chosen. Each line represents one of the five different simulated datasets, which are used to fit the models, the x-axis the different values of λ for which the models of different ranks (1, 2 or 3) are fitted, and the y-axis the LPE value. The red dots represent the λ for which each of the five datasets achieved the lowest LPE. The median of those 5 λ values is chosen as the best λ to fit all the 100 datasets. The datasets used for this figure are simulated using the 'case 2' way of simulation, correlation equal to 0.7, 1000 samples (n), 300 predictors (p), and 8 outcomes (K).

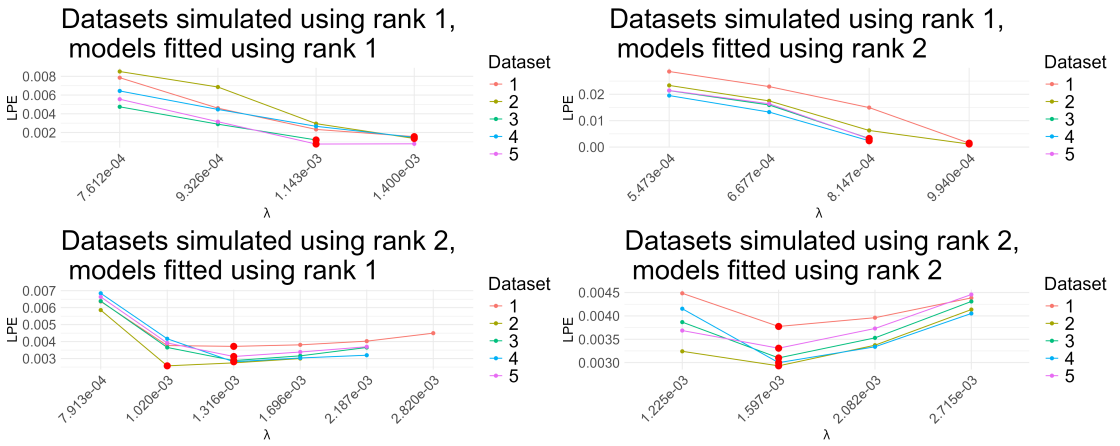

Figure A21: The figure represents the plots based on which the different λ values for fitting the different models are chosen. Each line represents one of the five different simulated datasets, which are used to fit the models, the x-axis the different values of λ for which the models of different ranks (1 or 2) are fitted, and the y-axis the LPE value. The red dots represent the λ for which each of the five datasets achieved the lowest LPE. The median of those 5 λ values is chosen as the best λ to fit all the 100 datasets. The datasets used for this figure are simulated using the 'case 2' way of simulation, correlation equal to 0.7, 5000 samples (n), 300 predictors (p), and 8 outcomes (K).

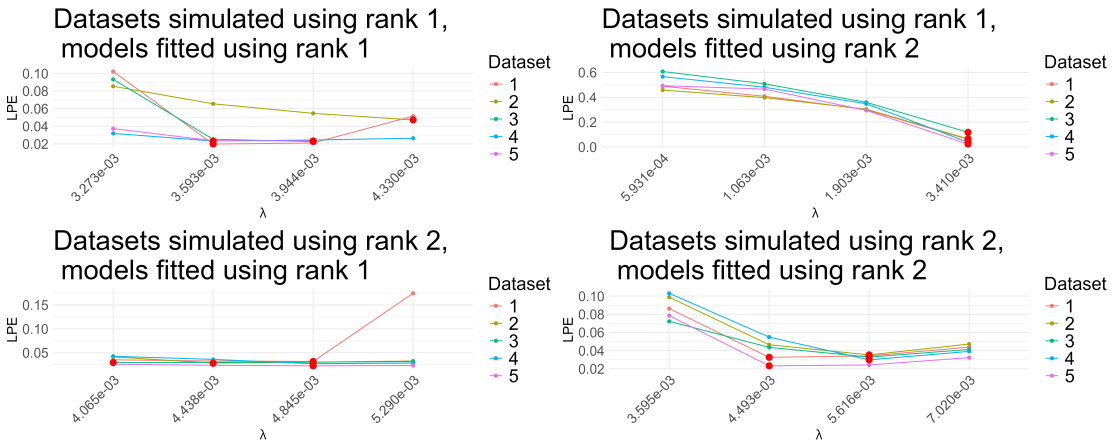

Figure A22: The figure represents the plots based on which the different λ values for fitting the different models are chosen. Each line represents one of the five different simulated datasets, which are used to fit the models, the x-axis the different values of λ for which the models of different ranks (1 or 2) are fitted, and the y-axis the LPE value. The red dots represent the λ for which each of the five datasets achieved the lowest LPE. The median of those 5 λ values is chosen as the best λ to fit all the 100 datasets. The datasets used for this figure are simulated using the 'case 3' way of simulation, correlation equal to 0.1, 1000 samples (n), 300 predictors (p), and 8 outcomes (K).

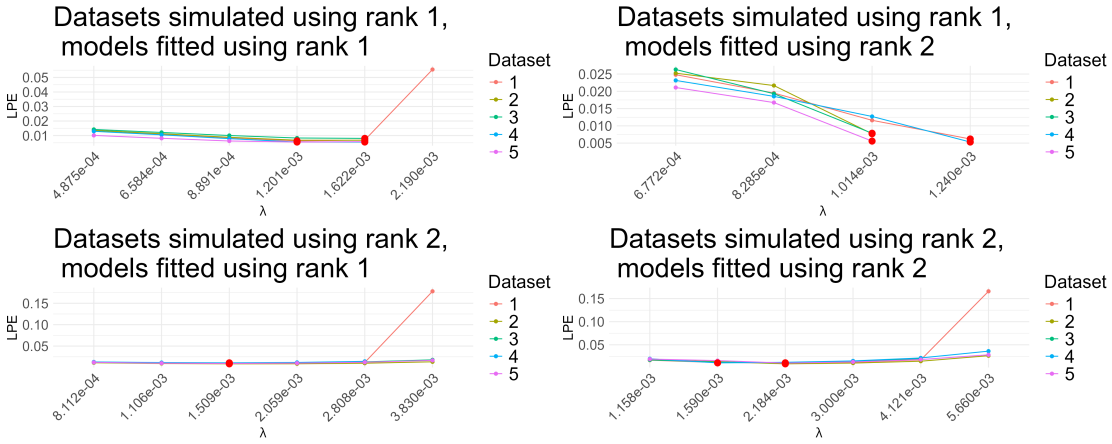

Figure A23: The figure represents the plots based on which the different λ values for fitting the different models are chosen. Each line represents one of the five different simulated datasets, which are used to fit the models, the x-axis the different values of λ for which the models of different ranks (1 or 2) are fitted, and the y-axis the LPE value. The red dots represent the λ for which each of the five datasets achieved the lowest LPE. The median of those 5 λ values is chosen as the best λ to fit all the 100 datasets. The datasets used for this figure are simulated using the 'case 3' way of simulation, correlation equal to 0.1, 5000 samples (n), 300 predictors (p), and 8 outcomes (K).

6.2.2.3 Performance values

Sample_size	Case	Correlation	Ranks_simulate	Ranks_fit	Average_MSE	MC_error_MSE
1000	1	0.7	1	1	7.0263e-05	8.6514e-07
1000	1	0.7	1	2	1.8779e-04	2.5990e-06
1000	1	0.7	2	1	1.4880e-04	1.4943e-06
1000	1	0.7	2	2	1.4282e-04	1.5055e-06
1000	2	0.7	1	1	1.0244e-04	3.3500e-06
1000	2	0.7	1	2	1.1447e-03	1.3587e-05
1000	2	0.7	2	1	8.7892e-05	1.0516e-06
1000	2	0.7	2	2	3.9765e-05	6.5703e-07
1000	2	0.7	2	3	3.9589e-05	6.6124e-07
1000	3	0.1	1	1	6.3042e-05	1.1773e-06
1000	3	0.1	1	2	2.1436e-04	1.1773e-06
1000	3	0.1	2	1	7.1350e-05	7.6988e-07
1000	3	0.1	2	2	7.3306e-05	9.1146e-07

Table A13: Simulation study 2, average MSE and Monte Carlo error rounded to 4 decimals for different combinations of parameters. All scenarios consist of 100 simulated datasets (S) with 300 predictors (p), 8 outcomes (K), and sample space (n):1000.

Sample_size	Case	Correlation	Ranks_simulate	Ranks_fit	Average_MSE	MC_error_MSE
5000	1	0.7	1	1	3.4487e-05	2.8406e-07
5000	1	0.7	1	2	3.4483e-05	2.8973e-07
5000	1	0.7	2	1	6.2616e-05	4.0678e-07
5000	1	0.7	2	2	6.2241e-05	4.0418e-07
5000	2	0.7	1	1	1.1319e-05	2.3543e-07
5000	2	0.7	1	2	5.6025e-05	7.4143e-07
5000	2	0.7	2	1	2.2053e-05	2.1293e-07
5000	2	0.7	2	2	2.2067e-05	2.2013e-07
5000	3	0.1	1	1	1.7696e-05	1.2122e-07
5000	3	0.1	1	2	4.0629e-05	3.6286e-07
5000	3	0.1	2	1	3.2087e-05	1.9999e-07
5000	3	0.1	2	2	3.0883e-05	1.9682e-07

Table A14: Simulation study 2, average MSE and Monte Carlo error rounded to 4 decimals for different combinations of parameters. All scenarios consist of 100 simulated datasets (S) with 300 predictors (p), 8 outcomes (K), and sample space (n):5000.

Sample_size	Case	Correlation	Ranks_simulate	Ranks_fit	Average_absolute_bias	MC_error_absolute_bias
1000	1	0.7	1	1	3.6697e-03	1.7108e-05
1000	1	0.7	1	2	5.5203e-03	2.8549e-05
1000	1	0.7	2	1	6.8344e-03	2.4899e-05
1000	1	0.7	2	2	6.6668e-03	2.4387e-05
1000	2	0.7	1	1	1.5381e-03	2.0763e-05
1000	2	0.7	1	2	9.1373e-03	6.9408e-05
1000	2	0.7	2	1	3.4750e-03	1.9130e-05
1000	2	0.7	2	2	2.2583e-03	1.2864e-05
1000	2	0.7	2	3	2.2574e-03	1.2900e-05
1000	3	0.1	1	1	3.2373e-03	1.6141e-05
1000	3	0.1	1	2	4.9585e-03	1.6141e-05
1000	3	0.1	2	1	4.6537e-03	1.7289e-05
1000	3	0.1	2	2	4.5803e-03	1.7255e-05

Table A15: Simulation study 2, average absolute bias and Monte Carlo error rounded to 4 decimals for different combinations of parameters. All scenarios consist of 100 simulated datasets (S) with 300 predictors (p), 8 outcomes (K), and sample space (n):1000.

Sample_size	Case	Correlation	Ranks_simulate	Ranks_fit	Average_absolute_bias	MC_error_absolute_bias
5000	1	0.7	1	1	3.1997e-03	1.1987e-05
5000	1	0.7	1	2	3.1959e-03	1.2170e-05
5000	1	0.7	2	1	5.2623e-03	1.6152e-05
5000	1	0.7	2	2	5.2488e-03	1.6103e-05
5000	2	0.7	1	1	8.0751e-04	6.8667e-06
5000	2	0.7	1	2	2.1234e-03	1.5278e-05
5000	2	0.7	2	1	2.1868e-03	9.5847e-06
5000	2	0.7	2	2	2.1748e-03	9.5871e-06
5000	3	0.1	1	1	2.4400e-03	8.5617e-06
5000	3	0.1	1	2	3.3713e-03	1.3070e-05
5000	3	0.1	2	1	3.6437e-03	1.1542e-05
5000	3	0.1	2	2	3.5617e-03	1.1294e-05

Table A16: Simulation study 2, average absolute bias and Monte Carlo error rounded to 4 decimals for different combinations of parameters. All scenarios consist of 100 simulated datasets (S) with 300 predictors (p), 8 outcomes (K), and sample space (n):5000.

Sample_size	Case	Correlation	Ranks_simulate	Ranks_fit	Average_LPE	MC_error_LPE
1000	1	0.7	1	1	1.0529e-02	4.0540e-05
1000	1	0.7	1	2	2.1129e-02	1.0170e-04
1000	1	0.7	2	1	1.8491e-02	6.9689e-05
1000	1	0.7	2	2	2.2088e-02	8.7416e-05
1000	2	0.7	1	1	1.4502e-02	1.2590e-04
1000	2	0.7	1	2	1.2147e-01	4.7636e-04
1000	2	0.7	2	1	1.5966e-02	9.6373e-05
1000	2	0.7	2	2	9.2052e-03	3.4256e-05
1000	2	0.7	2	3	9.1311e-03	3.4333e-05
1000	3	0.1	1	1	3.0492e-02	1.6665e-04
1000	3	0.1	1	2	8.7866e-02	1.6665e-04
1000	3	0.1	2	1	3.1855e-02	1.1521e-04
1000	3	0.1	2	2	3.9839e-02	1.4048e-04

Table A17: Simulation study 2, average linear predictor error (LPE) and Monte Carlo error rounded to 4 decimals for different combinations of parameters. All scenarios consist of 100 simulated datasets (S) with 300 predictors (p), 8 outcomes (K), and sample space (n):1000.

Sample_size	Case	Correlation	Ranks_simulate	Ranks_fit	Average_LPE	MC_error_LPE
5000	1	0.7	1	1	3.8929e-03	6.9797e-06
5000	1	0.7	1	2	3.9106e-03	7.1273e-06
5000	1	0.7	2	1	6.7630e-03	1.2252e-05
5000	1	0.7	2	2	6.9935e-03	1.2721e-05
5000	2	0.7	1	1	1.8175e-03	3.6577e-06
5000	2	0.7	1	2	6.0608e-03	1.4065e-05
5000	2	0.7	2	1	2.8372e-03	4.4499e-06
5000	2	0.7	2	2	3.0349e-03	4.8096e-06
5000	3	0.1	1	1	6.4319e-03	1.1612e-05
5000	3	0.1	1	2	1.3017e-02	2.8116e-05
5000	3	0.1	2	1	1.0371e-02	1.8505e-05
5000	3	0.1	2	2	1.1397e-02	1.9803e-05

Table A18: Simulation study 2, average linear predictor error (LPE) and Monte Carlo error rounded to 4 decimals for different combinations of parameters. All scenarios consist of 100 simulated datasets (S) with 300 predictors (p), 8 outcomes (k), and sample space (n):5000.

Sample_size	Case	Correlation	Ranks_simulate	Ranks_fit	Mean B Corr	Sd B Corr	Mean ZB Corr	Sd ZB Corr
1000	1	0.7	1	1	1.5142e-01	3.8612e-02	9.8921e-01	2.0677e-03
1000	1	0.7	1	2	1.1595e-01	4.5940e-02	9.7778e-01	1.2002e-02
1000	1	0.7	2	1	1.9302e-01	3.7471e-02	9.9449e-01	8.8299e-04
1000	1	0.7	2	2	1.8966e-01	3.7586e-02	9.9418e-01	9.9801e-04
1000	2	0.7	1	1	5.7517e-02	5.5461e-02	8.2347e-01	2.0871e-01
1000	2	0.7	1	2	8.7584e-03	1.9087e-02	2.4724e-01	2.4249e-01
1000	2	0.7	2	1	5.0919e-02	2.9211e-02	9.4259e-01	9.3649e-02
1000	2	0.7	2	2	6.1037e-02	2.7191e-02	9.7252e-01	6.6467e-03
1000	2	0.7	2	3	6.3154e-02	2.6613e-02	9.7265e-01	6.8306e-03
1000	3	0.1	1	1	1.3985e-01	6.6428e-02	7.1618e-01	2.2426e-01
1000	3	0.1	1	2	6.1559e-02	6.7902e-02	3.6873e-01	3.1576e-01
1000	3	0.1	2	1	2.7306e-01	3.3923e-02	9.0936e-01	1.3581e-02
1000	3	0.1	2	2	2.6284e-01	3.8284e-02	8.8547e-01	2.9830e-02

Table A19: Simulation study 2, the mean and standard deviation of (100×1) vector with correlation between true $\mathbf{B}$ matrix and estimated $\hat{\mathbf{B}}_\mathbf{s}$ matrices after vectorization, for 100 simulations (S) of 300 predictors (p), 8 outcomes (k), 1000 samples (n) and different combinations of parameters. Same metrics for vectorized linear predictor matrices (true: $\mathbf{Z_sB}$ and estimated: $\mathbf{Z_s}\hat{\mathbf{B}}_\mathbf{s}$). All are rounded to 4 decimals. The s symbolizes the simulation run $(s = 1, 2, .., 100)$.

Sample_size	Case	Correlation	Ranks_simulate	Ranks_fit	Mean B Corr	Sd B Corr	Mean ZB Corr	Sd ZB Corr
5000	1	0.7	1	1	2.3879e-01	3.9772e-02	9.9606e-01	4.7745e-04
5000	1	0.7	1	2	2.3851e-01	4.0255e-02	9.9599e-01	4.7722e-04
5000	1	0.7	2	1	3.1705e-01	3.4757e-02	9.9808e-01	1.7544e-04
5000	1	0.7	2	2	3.1716e-01	3.4736e-02	9.9806e-01	1.7719e-04
5000	2	0.7	1	1	1.2466e-01	5.7674e-02	9.7302e-01	1.4076e-02
5000	2	0.7	1	2	8.8321e-02	4.3567e-02	9.0670e-01	9.8559e-02
5000	2	0.7	2	1	9.3094e-02	2.9776e-02	9.9144e-01	1.1436e-03
5000	2	0.7	2	2	9.3391e-02	2.9892e-02	9.9108e-01	1.3395e-03
5000	3	0.1	1	1	3.1219e-01	3.9079e-02	9.4373e-01	5.6697e-03
5000	3	0.1	1	2	2.4494e-01	6.8364e-02	8.8578e-01	1.3100e-01
5000	3	0.1	2	1	4.3215e-01	3.3756e-02	9.7180e-01	2.6753e-03
5000	3	0.1	2	2	4.3317e-01	3.4676e-02	9.6955e-01	3.2002e-03

Table A20: Simulation study 2, the mean and standard deviation of (100×1) vector with correlation between true $\mathbf{B}$ matrix and estimated $\hat{\mathbf{B}}_\mathbf{s}$ matrices after vectorization, for 100 simulations (S) of 300 predictors (p), 8 outcomes (k), 5000 samples (n) and different combinations of parameters. Same metrics for vectorized linear predictor matrices (true: $\mathbf{Z_sB}$ and estimated: $\mathbf{Z_s}\hat{\mathbf{B}}_\mathbf{s}$). All are rounded to 4 decimals. The s symbolizes the simulation run $(s = 1, 2, .., 100)$.

6.2.2.4 Heatmaps of true B and estimated $\hat{\mathbf{B}}$

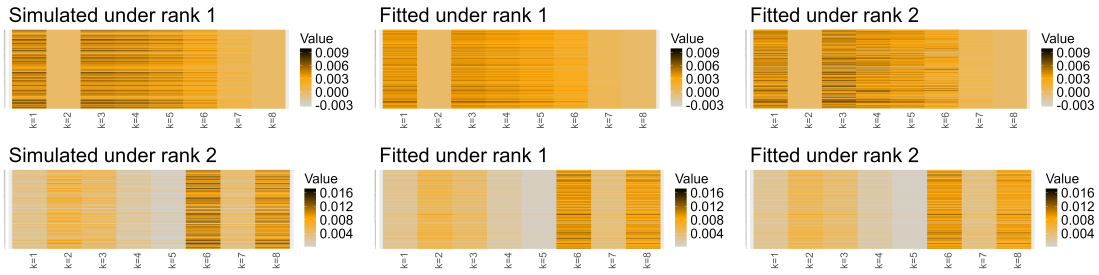

Figure A24: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1 or 2, mentioned in each subfigure's title. This is for 100 simulated datasets (S) with 1000 samples (n), 300 predictors (p), and 8 outcomes (K), simulated using the 'case 1' and correlation equal to 0.7.

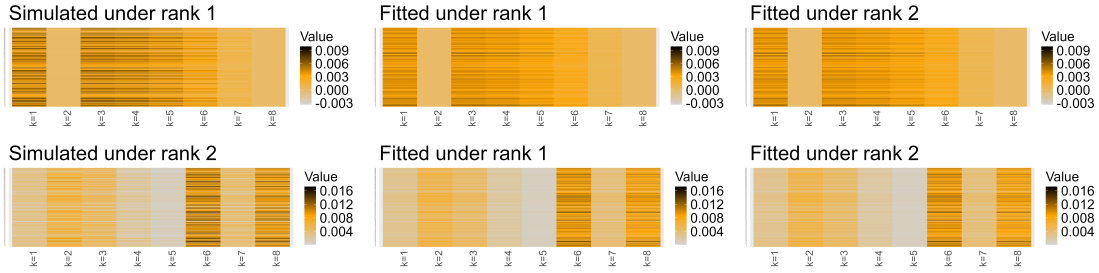

Figure A25: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1 or 2, mentioned in each sub-figure. This is for 100 simulated datasets (S) with 5000 samples (n), 300 predictors (p), and 8 outcomes (K), simulated using the 'case 1' and correlation equal to 0.7.

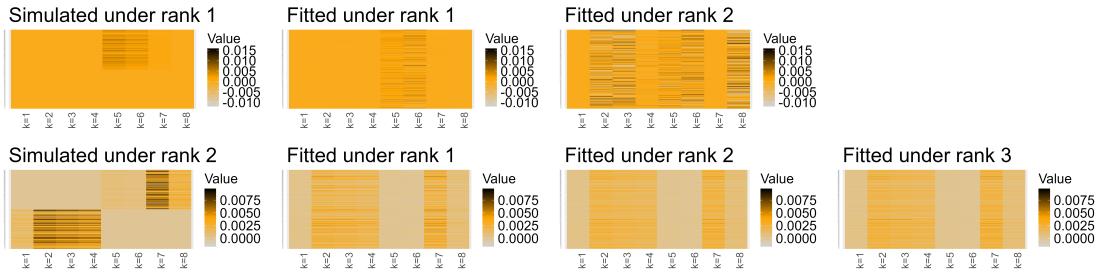

Figure A26: The leftmost heatmap of the first row represents the simulated under rank 1, $\mathbf{B}$ matrix, and the one of second line the simulated under rank 2, $\mathbf{B}$ matrix. The matrices next to them represent the mean of all the estimated $\mathbf{B}$ matrices by the models fitted using rank 1, 2, or 3, mentioned in each sub-figure. This is for 100 simulated datasets (S) with 1000 samples (n), 300 predictors (p), and 8 outcomes (K), simulated using 'case 2' and correlation equal to 0.7.

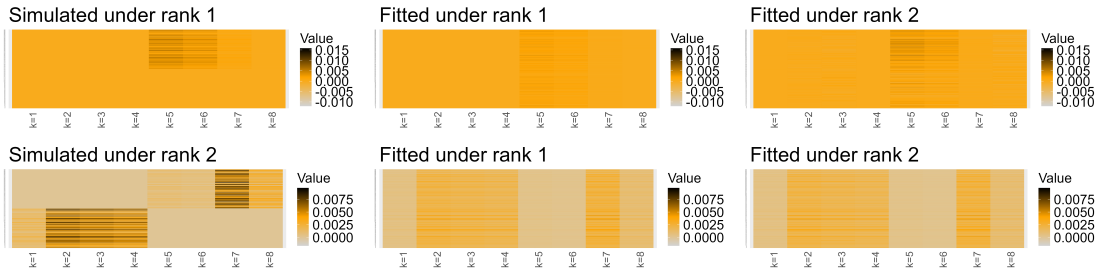

Figure A27: The leftmost heatmap of the first row represents the simulated under rank 1, $\mathbf{B}$ matrix, and the one of second line the simulated under rank 2, $\mathbf{B}$ matrix. The matrices next to them represent the mean of all the estimated $\mathbf{B}$ matrices by the models fitted using rank 1 or 2, mentioned in each sub-figure. This is for 100 simulated datasets (S) with 5000 samples (n), 300 predictors (p), and 8 outcomes (K), simulated using 'case 2' and correlation equal to 0.7.

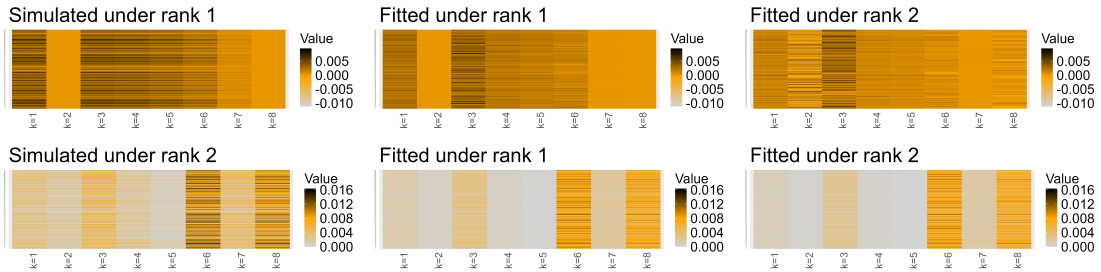

Figure A28: The leftmost heatmap of the first row represents the simulated under rank 1, $\mathbf{B}$ matrix, and the one of second line the simulated under rank 2, $\mathbf{B}$ matrix. The matrices next to them represent the mean of all the estimated $\mathbf{B}$ matrices by the models fitted using rank 1 or 2, mentioned in each sub-figure. This is for 100 simulated datasets (S) with 1000 samples (n), 300 predictors (p), and 8 outcomes (K), simulated using 'case 3' and correlation equal to 0.1.

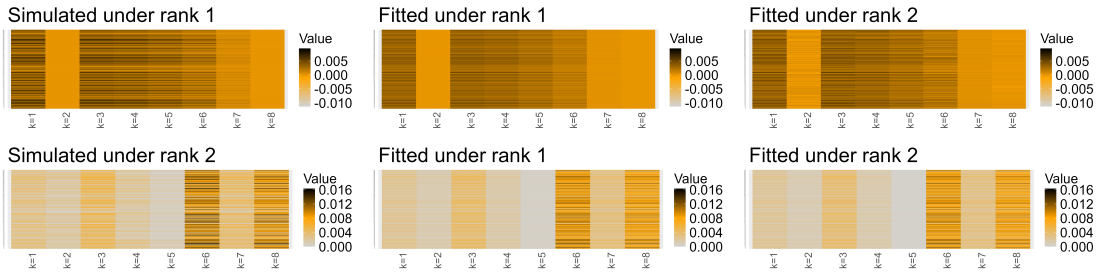

Figure A29: The leftmost heatmap of the first row represents the simulated under rank 1, **B** matrix, and the one of second line the simulated under rank 2, **B** matrix. The matrices next to them represent the mean of all the estimated **B** matrices by the models fitted using rank 1 or 2, mentioned in each sub-figure. This is for 100 simulated datasets (S) with 5000 samples (n), 300 predictors (p), 8 outcomes (K), simulated using 'case 3' and correlation equal to 0.1.

6.3 Code

Link GitHub Repository