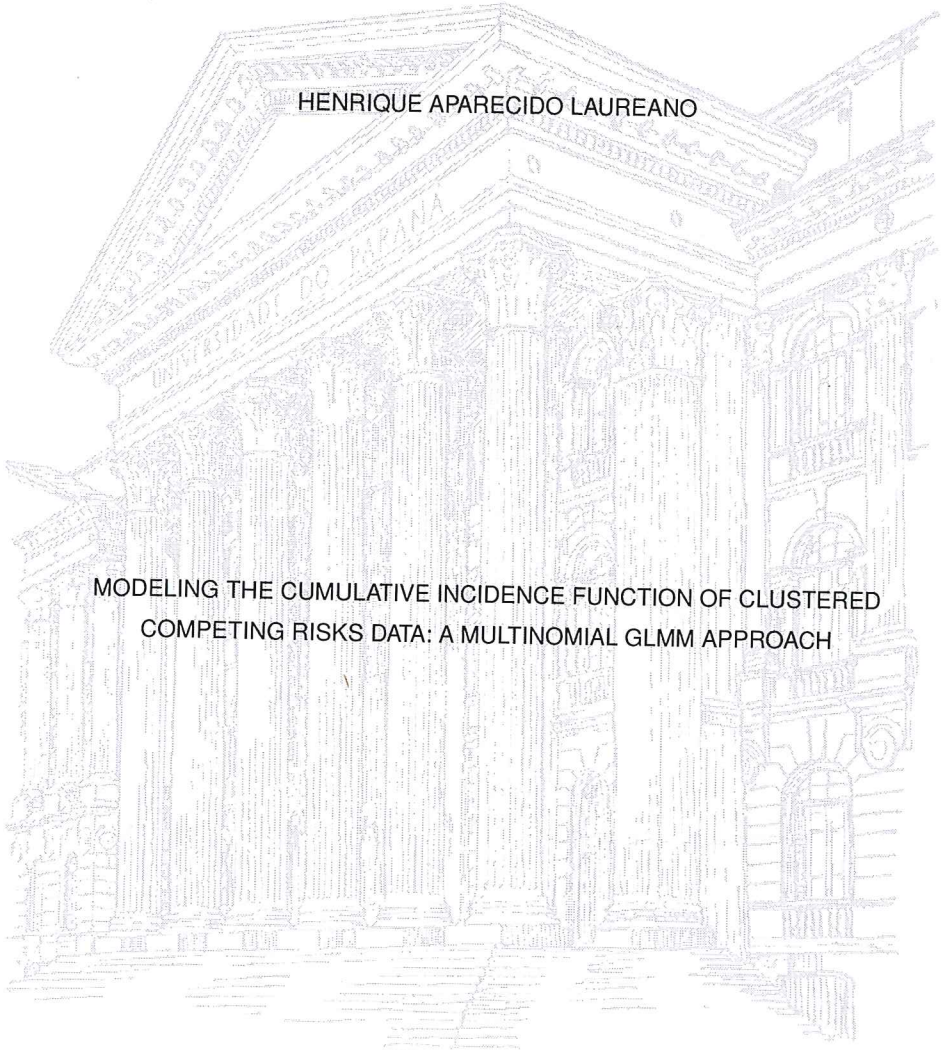
FEDERAL UNIVERSITY OF PARANÁ

HENRIQUE APARECIDO LAUREANO



MODELING THE CUMULATIVE INCIDENCE FUNCTION OF CLUSTERED COMPETING RISKS DATA: A MULTINOMIAL GLMM APPROACH

CURITIBA

2021

HENRIQUE APARECIDO LAUREANO

MODELING THE CUMULATIVE INCIDENCE FUNCTION OF CLUSTERED COMPETING RISKS DATA: A MULTINOMIAL GLMM APPROACH

Thesis presented to the Graduate Program of Numerical Methods in Engineering, Concentration Area in Mathematical Programming: Statistical Methods ~~Applied in Engineering~~, Federal University of Paraná, as part of the requirements to the obtention of the Master's Degree in Sciences.

Supervisor: Prof. PhD Wagner Hugo Bonat

Co-supervisor: Prof. PhD Paulo Justiniano Ribeiro Jr

CURITIBA

2021

2. Write the model in the Template Model Builder (TMB) software, developed by Kristensen et al. (2016) and possibly the most efficient likelihood-based way of doing such task.

3. Take advantage of TMB's functionalities with special attention to the computation of gradients and Hessians via a *state-of-art* automatic differentiation (AD) implementation; and a joint likelihood marginalization via an efficient Laplace approximation routine.

4. Study the model identifiability through the proposition of different complexity level models in terms of parametric space and latent effect structures.

5. Make exact likelihood-based inference to the cluster and cause-specific CIF of clustered competing risks data.

1.2 JUSTIFICATION

In the biomedical statistical modeling literature, the study of disease occurrence in related individuals receives the name of family studies. Key points of interest are the within-family dependence and determining the role of different risk factors. The within-family dependence may reflect both disease heritability and the impact of shared environmental effects. The role of different risk factors arrives in the class of multivariate models, which options are limited in the statistical literature. Thus, the number of statistical models for competing risks data that accommodate the within-cluster/family dependence is even more limited. Some modeling options are briefly commented in Cederkvist et al. (2019), with his pairwise composite approach being proposed as a new and better option to model the cause-specific cumulative incidence function (CIF), describing age at disease onset, of clustered competing risks data on the probability scale. We propose to model the cause-specific CIF and accommodate the within-family dependence in the same fashion (via a latent structure that allows the absolute risk and the failure time distribution to vary between families) but with an easier framework, based on a multinomial generalized linear mixed model approach.

1.3 LIMITATION

não!

This work restraint to the proposition and model identifiability study of a multinomial model for the cause-specific cumulative incidence function (CIF) of competing risks data, with a latent effect structure to accommodate within-family dependence with regard to both risk and timing. Given its considerable model complexity, hypothesis tests; residual analysis; and good-of-fit measures are not contemplated.

Inclue que vc se restringe a family studies onde os agrupamentos de clusters é grande porém elementos dentro de cada cluster é pequeno.

1.4 THESIS ORGANIZATION

This master thesis contains 6 chapters including this introduction. Chapter 2 presents a systematic review of the main aspects involved in the formulation, optimzation, and implementation of a generalized linear mixed model (GLMM). Given the modeling framework overview, Chapter 3 presents our multinomial GLMM (multiGLMM) to model the cause-specific cumulative incidence function (CIF) of clustered competing risks data. In Chapter 4 we describe the simulation procedure to generate synthetic data and present some model particularities. In Chapter 5 the obtained results are presented, and in Chapter 6 we discuss the contributions of this thesis and present some suggestions for future work.

Vc está estudando coisas relacionadas ao método de estimação não necessariamente sobre o modelo.
Mesmo que vc chegasse a conclusão que vc não consegue estimar nada o que usando o método de máx verossimilhança vc não consegui estimar. Isso não significa que o modelo não é estimável.
Vc precisa deixar bem a sua motivação para explicar que vc está avaliando propriedades do esquema de estimação e não do modelo

2 GENERALIZED LINEAR MIXED MODELS: FORMULATION, OPTIMIZATION, AND IMPLEMENTATION

This chapter presents a systematic review of the main theoretical aspects involved in the formulation, estimation, and implementation of a generalized linear mixed model (GLMM). We start in Section 2.1 with the model formulation framework, concluding with the so-called joint or full likelihood function. Section 2.2 address the marginalization of that joint likelihood, performed here in terms of a Laplace approximation technique. Section 2.3 discusses available alternatives for the marginal likelihood parameters optimization. Section 2.4 present the automatic differentiation (AD) procedure, the most efficent routine for the computation of derivatives, and a key point for us. Last but not least, in Section 2.5 we present the computational tool used to peform all the discussed methodology, a very exciting R (R Core Team, 2021) package called TMB: Template Model Builder, developed by Kristensen et al. (2016).

2.1 FORMULATION: OBTAINING A JOINT LIKELIHOOD FUNCTION

We model an n-vector of exponential family random variables $\boldsymbol{Y}$, in terms of its conditional expected value $\boldsymbol{\mu} \equiv \mathbb{E}(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{u})$, via a linear combination called of linear predictor and generally expressed by

$$g(\boldsymbol{\mu}) = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u}, \quad \boldsymbol{u} \sim \text{Multivariate Normal}(\boldsymbol{0}, \boldsymbol{\Sigma}). \tag{2.1}$$

In other words, a GLMM (MCCULLOCH; SEARLE, 2001) is a generalized linear model (GLM) in which the linear predictor depends on some Gaussian latent effects, $\boldsymbol{u}$, times a latent effects design-matrix $\boldsymbol{Z}$. Since we do not observe the latent component, an exemplification of the idea embedded in matrix $\boldsymbol{Z}$ is welcome. Suppose e.g., three individuals (or clusters) and that each one has two measures. This configures a repeated measures context, the most common latent structure in family studies. Also, it is reasonable to admit that each individual has its particular latent effect value. Consequently, we have

$$\boldsymbol{Z}\boldsymbol{u} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_1 \\ u_2 \\ u_2 \\ u_3 \\ u_3 \end{bmatrix},$$

where $\boldsymbol{u}^\top = [u_1\ u_2\ u_3]$ and $\boldsymbol{Z}$ has the role of projecting the values of $\boldsymbol{u}$ to match the number of measures.

In a mixed model the mean structure is approached as a combination of probability distributions. It is a combination since we have to assume probabilistic structures for the observed and non-observed/latent data. To each observed variable y_{ij} we have a probability distribution of the exponential family, denoted by $f(y_{ij} \mid \boldsymbol{u}_i, \boldsymbol{\theta})$. To the latent effect we have, generally, a (multivariate) Gaussian distribution, denoted by $f(\boldsymbol{u}_i \mid \boldsymbol{\Sigma})$. To each individual or unity under study i, and to each measure j, we have the product of these probability densities, a likelihood contribution.

Our goal is to estimate the parameter vector $\boldsymbol{\theta} = [\boldsymbol{\beta}\ \boldsymbol{\Sigma}]^\top$ of a mean structure, as in Equation 2.1. Besides the role of emphasizing the fact that $\boldsymbol{\mu}$ is a function of $\boldsymbol{\theta}$, and that we want to estimate $\boldsymbol{\theta}$, the likelihood function ties the probability densities i.e., the likelihood is the product of the product of probability densities, to each subject i. Since Y_i are mutually independent, the likelihood for $\boldsymbol{\theta}$ can be written as

$$L(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{u}) = \prod_{i=1}^{I} \prod_{j=1}^{n_i} f(y_{ij} \mid \boldsymbol{u}_i, \boldsymbol{\beta}, \boldsymbol{\Sigma})\, f(\boldsymbol{u}_i \mid \boldsymbol{\Sigma}). \tag{2.2}$$

From standard probability theory is easy to see that in the right-hand side (r.h.s.) we have a joint density, consequently, Equation 2.2 represents what is called a full or a joint likelihood function. What makes problematic working with this joint likelihood is that we do not have all the necessary information to just maximize it and get the desired parameter estimates. The latent effect $\boldsymbol{u}$ is *latent* i.e., we do not observe it. To handle this we have basically two available paths.

2.2 MARGINALIZATION: LAPLACE APPROXIMATION AND ALTERNATIVES

To deal with a joint likelihood function as in Equation 2.2 we have a choice to make. Be or not to be Bayesian. Each choice has its own difficulties, advantages, and characteristics.

The Bayesian path assumes that all $\boldsymbol{\theta}$ components are random variables. With all parameters being treated as random variables, and since we do not observe them, what the Bayesian framework does is try to compute the mode of each "parameter" marginal distribution, generally, via a sampling algorithm called MCMC: Markov chain Monte Carlo (GELFAND; SMITH, 1990; DIACONIS, 2009).

The advantage of being Bayesian is that we can reach an MCMC algorithm to basically any statistical model, the disadvantage is that this approach is very time consuming and we have to propose prior distributions to each "parameter". These prior proposals are not always easy to make, and the resulting marginal distributions can be very depending of it. A Bayesian approach can be applied in basically any context, without guarantees that will work - obtain convergence to all parameters is not

a straightforward task. However, in complex scenarios they can be the only available method to "maximize" the likelihood function. This is not the case here.

We have a joint density where one of the random variables is not observed, but we are not interested in it, only in the variance parameters inherent in it. Again, from standard probability theory, if we have a joint density we can just integrate out the undesired variable resulting in

$$\begin{aligned} L(\theta \mid y) &= \prod_{i=1}^{I} \int_{\mathcal{R}^{u_i}} \left[\prod_{j=1}^{n_i} f(y_{ij} \mid u_i, \beta, \Sigma)\, f(u_i \mid \Sigma) \right] \mathrm{d}u_i \\ &= \prod_{i=1}^{I} \int_{\mathcal{R}^{u_i}} f(y_i, u_i \mid \theta)\, \mathrm{d}u_i, \end{aligned} \tag{2.3}$$

a marginal density that keeps the parameters Σ of the integrated variable.

When the response distribution of a mixed model is Gaussian, is analytically tractable to integrate u out of the joint density. Consequently, it is possible to evaluate the marginal likelihood exactly. This is the case of the linear mixed models (LMMs) and one of the main differences to the GLMMs. When the response distribution is not Gaussian, generally, it is not anymore analytically tractable to integrate out the latent effect. So what do we do? Well, we have basically two options again.

We can avoid the integrals in Equation 2.3, replacing it by integrals that are more analytically tractable. This can be performed via an algorithm called Expectation-Maximization (EM), proposed by Dempster, Laird & Rubin (1977). This approach is considered a little bit naive and generally is not recommended if you have a better option. The other option consists of performing a numerical integration i.e., approximating the integral. The most common way of doing that in the statistical modeling literature is via an importance sampling version of the Gaussian quadrature rule, denoted adaptive Gaussian quadrature (AGQ) (PINHEIRO; CHAO, 2006). In general, adaptive Gaussian quadratures are not so simple to use (computationally expensive; we have to choose how many integration points will be used; and we also have to choose an importance distribution to approximate the integrand).

To us, the better option consists in take advantage of the exponential family structure together with the fact that we are dealing with Gaussian latent effects. These ideas converge to an adaptive Gaussian quadrature with one integration point, also called as *Laplace approximation* (MOLENBERGHS; VERBEKE, 2005; SHUN; MCCULLAGH, 1995; TIERNEY; KADANE, 1986; WOOD, 2015).

With an integral that is analytically intractable, we may approximate it to obtain a tractable closed-form expression allowing then the numerical maximization of the resulting marginal likelihood function (BONAT; RIBEIRO, 2016). The Laplace

approximation has been designed to approximate integrals in the form

$$\int_{\mathcal{R}^{u_i}} \exp\{Q(u_i)\}\mathrm{d}u_i \approx (2\pi)^{n_u/2}\, |Q''(\hat{u}_i)|^{-1/2}\, \exp\{Q(\hat{u}_i)\}, \tag{2.4}$$

where $Q(u_i)$ is a known, unimodal bounded function, and $\hat{u}_i$ is the value for which $Q(u_i)$ is maximized. As Wood (2015) shows, a Laplace approximation consists of a second order Taylor expansion of $\log f(y_i, u_i \mid \theta)$, about $\hat{u}_i$, that gives

$$\log f(y_i, u_i \mid \theta) \approx \log f(y_i, \hat{u}_i \mid \theta) - \frac{1}{2}(u_i - \hat{u}_i)^\top H\, (u_i - \hat{u}_i),$$

where $H = -\nabla^2_u \log f(y_i, \hat{u}_i \mid \theta)$. Hence, we can approximate the joint by

$$f(y_i, u_i \mid \theta) \approx f(y_i, \hat{u}_i \mid \theta)\, \exp\left\{ -\frac{1}{2}(u_i - \hat{u}_i)^\top H\, (u_i - \hat{u}_i) \right\}. \tag{2.5}$$

From here we start to take advantage of the points mentioned above.

First, the fact that we are dealing with Gaussian distributed latent effects. In Equation 2.5 we have the core of a Gaussian density, that complete is

$$\int_{\mathcal{R}^{u_i}} \frac{1}{(2\pi)^{n_u/2}\, |H^{-1}|^{1/2}}\, \exp\left\{ -\frac{1}{2}(u_i - \hat{u}_i)^\top H\, (u_i - \hat{u}_i) \right\} \mathrm{d}u_i = 1$$

i.e., integrates to 1. Integrating Equation 2.5 follows that

$$\begin{aligned} \int_{\mathcal{R}^{u_i}} f(y_i, u_i \mid \theta)\, \mathrm{d}u_i &\approx f(y_i, \hat{u}_i \mid \theta) \int_{\mathcal{R}^{u_i}} \exp\left\{ -\frac{1}{2}(u_i - \hat{u}_i)^\top H\, (u_i - \hat{u}_i) \right\} \mathrm{d}u_i \\ &= (2\pi)^{n_u/2}\, |H|^{-1/2}\, f(y_i, \hat{u}_i \mid \theta) \end{aligned}$$

i.e., we get Equation 2.4, a first order Laplace approximation to the integral. Careful accounting of the approximation error shows it to generally be $\mathcal{O}(n^{-1})$, where n is the sample size, and assuming a fixed length for u_i (WOOD, 2015).

The second advantage of a Laplace approximation approach in a GLMM is the exponential family structure. In a usual GLMM the response follows a one-parameter exponential family distribution that can be written as

$$f(y_i \mid u_i, \theta) = \exp\left\{ y_i^\top (x_i\beta + z_i u_i) - \mathbf{1}_i^\top b(x_i\beta + z_i u_i) + \mathbf{1}_i^\top c(y_i) \right\},$$

where $b(\cdot)$ and $c(\cdot)$ are known functions.

This general and easy to compute expression, together with a (multivariate) Gaussian distribution, highlights the convenience of the Laplace method. The $Q(u_i)$ function to be maximized can be expressed as

$$\begin{aligned} Q(u_i) = {} & y_i^\top (x_i\beta + z_i u_i) - \mathbf{1}_i^\top b(x_i\beta + z_i u_i) + \mathbf{1}_i^\top c(y_i) \\ & - \frac{n_u}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2} u_i^\top \Sigma^{-1}\, u_i. \end{aligned} \tag{2.6}$$

The approximation in Equation 2.4 requires the maximum $\hat{u}_i$ of the function $Q(u_i)$. As we assume a Gaussian distribution with a known mean for the latent effects, we have the perfect initial guess for a gradient-based maximization method, as the Newton-Raphson (NR) algorithm.

The NR method consists of an iterative scheme as follows:

$$u_i^{(k+1)} = u_i^{(k)} - Q''(u_i^{(k)})^{-1}\, Q'(u_i^{(k)}), \quad k = 0, 1, \ldots$$

until convergence, which gives $\hat{u}_i$. At this stage, all parameters θ are considered known. Bonat & Ribeiro (2016) presents the generic expressions for the derivatives required by the NR method, given by the following:

$$Q'(u_i^{(k)}) = \{y_i - b'(x_i\beta + z_i u_i^{(k)})\}^\top - {u_i^{(k)}}^\top \Sigma^{-1},$$
$$Q''(u_i^{(k)}) = -\text{diag}\{b''(x_i\beta + z_i u_i^{(k)})\} - \Sigma^{-1}.$$

We have the initial guesses at $k = 0$.

Finally, the marginal log-likelihood function returned by the Laplace approximation, to each individual or unit under study i, is as follows:

$$\begin{aligned} l(\theta \mid y_i) = \log L(\theta \mid y_i) = {} & \frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left|\text{diag}\{b''(x_i\beta + z_i\hat{u}_i)\} + \Sigma^{-1}\right| \\ & + y_i^\top (x_i\beta + z_i\hat{u}_i) - \mathbf{1}_i^\top b(x_i\beta + z_i\hat{u}_i) + \mathbf{1}_i^\top c(y_i) \\ & - \frac{n_u}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}\hat{u}_i^\top \Sigma^{-1}\, \hat{u}_i, \end{aligned}$$

that can now be numerically maximized over the model parameters $\theta = [\beta\ \Sigma]^\top$.

2.3 OPTIMIZATION: MARGINAL LIKELIHOOD FUNCTION

At this point it is already clear that we have two optimizations to be performed, an "inside" and an "outside" optimization. The inside one is made into the Laplace approximation layer via a Newton-Raphson algorithm, a Newton's method. The outside optimization is made with the Laplace approximation outputs i.e., the maximization of Equation 2.3's marginal log-likelihood over its parameters θ. This task is usually performed via a quasi-Newton method, we focus on two of the most traditional ones: the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm and the PORT routines.

The inside optimization is the numerical maximization of the joint log-likelihood with respect to (w.r.t.) its latent effects. This is kind of a simple task since all model parameters are considered as fixed, and we "know" that the latent effects are distributed with zero mean i.e., we have the perfect initial guess. In this context, the use of a Newton's method is straightforward. When we talk about the outside optimization

it is a completely different scenario, it is not straightforward to find a good initial guess or reach convergence. Thus, more robust methods are a good choice.

In optimization, Newton methods are algorithms for finding local maxima and minima of functions i.e., the search for the zeroes of the gradient of that function. Newton methods are characterized by the use of a symmetric matrix of function's second derivatives, the Hessian matrix. Quasi-Newton methods are based on Newton's method and are seen as an alternative to it. They can be used if the Hessian is unavailable or if is too expensive to compute it at every iteration.

As shown in Nocedal & Wright (2006), major advantages of quasi-Newton methods over Newton's method are that the Hessian matrix does not need to be computed, it is approximated; and it also does not need to be inverted. Newton's method requires the Hessian to be inverted, typically by solving a system of linear equations - often quite costly. In contrast, quasi-Newton methods usually generate an estimate of it directly. As in Newton's method, they use a second-order approximation to find the minimum of a function $f(x)$. The Taylor series of $f(x)$ around an iterate is

$$f(x_k + \Delta x) \approx f(x_k) + \nabla f(x_k)^\top \Delta x + \frac{1}{2}\Delta x^\top B\, \Delta x,$$

where $\nabla f(\cdot)$ is the gradient, and B an approximation to the Hessian matrix. The gradient of this approximation w.r.t. Δx is

$$\nabla f(x_k + \Delta x) \approx \nabla f(x_k) + B\, \Delta x,$$

setting this gradient to zero provides the Newton step:

$$\Delta x = -B^{-1}\nabla f(x_k).$$

The Hessian approximation B is chosen to satisfy

$$\nabla f(x_k + \Delta x) = \nabla f(x_k) + B\, \Delta x,$$

which is called the *secant* equation i.e., the Taylor series of the gradient itself. Solving for B and applying the Newton's step with the updated value is equivalent to the *secant* method. Quasi-Newton methods are a generalization of the secant method to find the root of the first derivative for multidimensional problems. The various quasi-Newton methods differ in their choice of the solution to the secant equation.

In a general quasi-Newton method, the unknown x_k is updated applying the Newton's step calculated using the current approximate Hessian matrix B_k in the following fashion:

- $\Delta x_k = -\alpha_k B_k^{-1}\nabla f(x_k)$, with α chosen to satisfy the so called Wolfe conditions (NOCEDAL; WRIGHT, 2006, p. 34);

- $x_{k+1} = x_k + \Delta x_k$;
- The gradient computed at the new point $\nabla f(x_{k+1})$, and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ is used to update the approximate Hessian B_{k+1}, or directly its inverse $H_{k+1} = B_{k+1}^{-1}$.

The most popular quasi-Newton method is the BFGS algorithm, named for its discoverers Broyden, Fletcher, Goldfarb, and Shanno. It has the following update formula

$$B_{k+1} = B_k + \frac{y_k y_k^\top}{y_k^\top \Delta x_k} - \frac{B_k \Delta x_k (B_k \Delta x_k)^\top}{\Delta x_k^\top B_k \Delta x_k},$$

$$H_{k+1} = B_{k+1}^{-1} = \left(I - \frac{\Delta x_k y_k^\top}{y_k^\top \Delta x_k}\right) H_k \left(I - \frac{y_k \Delta x_k^\top}{y_k^\top \Delta x_k}\right) + \frac{\Delta x_k \Delta x_k^\top}{y_k^\top \Delta x_k}.$$

Another quasi-Newton method popular in the statistical modeling literature, is the one based on the PORT routines {http://www.netlib.org/port/}. It is a Fortran mathematical subroutine library designed to be *portable* over different types of computers, developed by David Gay in the Bell Labs (GAY, 1990). It a quasi-Newton adaptive nonlinear least-squares algorithm (DENNIS; GAY; WELSCH, 1981) with the following update formula

$$\begin{aligned} B_{k+1} = B_k & \\ & + \frac{(y_k - B_k \Delta x_k) \Delta x_k^\top B_k + B_k \Delta x_k (y_k - B_k \Delta x_k)^\top}{\Delta x_k^\top B_k \Delta x_k} \\ & - \frac{\Delta x_k^\top (y_k - B_k \Delta x_k) B_k \Delta x_k \Delta x_k^\top B_k}{(\Delta x_k^\top B_k \Delta x_k)^\top \Delta x_k^\top B_k \Delta x_k}. \end{aligned}$$

As Nocedal & Wright (2006) points out, each quasi-Newton method iteration can be performed at a cost of $\mathcal{O}(n^2)$ arithmetic operations (plus the cost of function and gradient evaluations); there are no $\mathcal{O}(n^3)$ operations such as linear system solves or matrix-matrix operations. In the BFGS algorithm is known that the rate of convergence is superlinear, which is a valid assumption to any quasi-Newton method and is fast enough for most practical purposes. Even though Newton's method converges more rapidly, quadratically, its cost per iteration usually is higher because of its need for second derivatives and solution of a linear system.

In this thesis, the used BFGS implementation is the one in the `R` (R Core Team, 2021) function `base::optim()`, and the PORT routine used is the one implemented in the `R` function `base::nlminb()`.

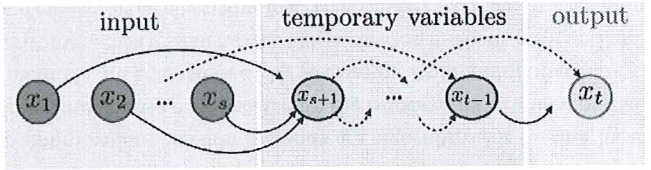
2.4 AD: AUTOMATIC DIFFERENTIATION

The computation of gradients, $\nabla f(x)$, are a fundamental and crucial task but also the main computational bottleneck to any Newton and quasi-Newton method. We choose to use the most efficient manner of computing gradients, and one of the best scientific computing techniques but still not so famous in the statistical modeling literature, the *automatic differentiation* (AD) procedure. AD has two modes, the so-called forward and reverse mode. We will talk a bit about both but we will use only the reverse mode. The reason can be illustraded by a simple example, given later.

Automatic differentiation, also called algorithmic differentiation or computational differentiation, is a set of techniques to numerically and recursively evaluate the derivative of a function specified by a computer program. AD techniques are based on the observation that any function, no matter how complicated, is evaluated by performing a sequence of simple elementary operations involving just one or two arguments at a time. Derivatives of arbitrary order can be computed automatically, automatized and accurately to working precision. Most of the information in this section was taken of Peyré (2020), but Wood (2015, p. 120) and Nocedal & Wright (2006, p. 204) are also very good references.

The most common differentiation approaches are finite differences (FD) and symbolic calculus. Considering a function $f : \mathbb{R}^p \to \mathbb{R}$ and the goal of deriving a method to evaluate $\nabla f : \mathbb{R}^p \to \mathbb{R}^p$, the approximation of this vector field via FD would require $p + 1$ evaluations of f. The same task via reverse mode AD has in most cases a cost proportional to a single evaluation of f. AD is similar to symbolic calculus in the sense that it provides an exact gradient computation, up to machine precision. However, symbolic calculus does not takes into account the underlying algorithm which compute the function, while AD factorizes the computation of the derivative according to an efficient algorithm. The use of AD is inherent to the use of a computational graph, as exemplified in Figure 2.

FIGURE 2 – A COMPUTATIONAL GRAPH



SOURCE: Peyré (2020, p. 31).

Assuming that f is implemented in an algorithm, the goal is to compute the

derivatives

$$\frac{\partial f(x)}{\partial x_k} \in \mathbb{R}^{n_t \times n_k},$$

for a numerical algorithm (succession of functions) of the form

$$\forall\, k = s+1, \ldots, t, \quad x_k = f_k(x_1, \ldots, x_{k-1}),$$

where f_k is a function which only depends on the previous variables. The computational graph as in Figure 2, has the role of represent the linking of the variables involved in f_k to x_k. The evaluation of $f(x)$ corresponds to a forward traversal of this graph.

Now, how we evaluate f through the graph? Via one of the AD modes.

2.4.1 Forward Mode

The forward mode correspond to the usual way of computing differentials. The method initialize with the derivative of the input nodes

$$\frac{\partial x_1}{\partial x_1} = \mathrm{Id}_{n_1 \times n_1}, \quad \frac{\partial x_2}{\partial x_1} = 0_{n_2 \times n_1}, \quad \frac{\partial x_s}{\partial x_1} = 0_{n_s \times n_1},$$

and then iteratively make use of the following recursion formula

$$\forall\, k = s+1, \ldots, t,$$
$$\frac{\partial x_k}{\partial x_1} = \sum_{l \,\in\, \text{father}(k)} \frac{\partial x_k}{\partial x_l} \times \frac{\partial x_l}{\partial x_1} = \sum_{l \,\in\, \text{father}(k)} \frac{\partial}{\partial x_l} f_k(x_1, \ldots, x_{k-1}) \times \frac{\partial x_l}{\partial x_1}.$$

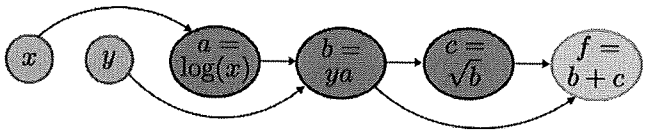
The notation "father(k)" denotes the nodes $l < k$ of the graph that are connected to k. We make use of Peyré (2020, p. 32)'s simple example.

Example. Consider the function

$$f(x,y) = y\log(x) + \sqrt{y\log(x)}$$

with the corresponding computational graph being displayed in Figure 3.

FIGURE 3 – EXAMPLE OF A SIMPLE COMPUTATIONAL GRAPH



SOURCE: Peyré (2020, p. 33).

The forward mode iterations to compute the derivative w.r.t. x following the computational graph, are given by

$$\frac{\partial x}{\partial x} = 1, \quad \frac{\partial y}{\partial x} = 0$$
$$\frac{\partial a}{\partial x} = \frac{\partial a}{\partial x}\frac{\partial x}{\partial x} = \frac{1}{x}\frac{\partial x}{\partial x} \qquad \{x \mapsto a = \log(x)\}$$
$$\frac{\partial b}{\partial x} = \frac{\partial b}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial b}{\partial y}\frac{\partial y}{\partial x} = y\frac{\partial a}{\partial x} + 0 \qquad \{(y,a) \mapsto b = ya\}$$
$$\frac{\partial c}{\partial x} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial x} = \frac{1}{2\sqrt{b}}\frac{\partial b}{\partial x} \qquad \{b \mapsto c = \sqrt{b}\}$$
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial x} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial x} = 1\frac{\partial b}{\partial x} + 1\frac{\partial c}{\partial x} \qquad \{(b,c) \mapsto f = b + c\}$$

To compute the derivative w.r.t. y we run another forward process

$$\frac{\partial x}{\partial y} = 0, \quad \frac{\partial y}{\partial y} = 1$$
$$\frac{\partial a}{\partial y} = \frac{\partial a}{\partial x}\frac{\partial x}{\partial y} = 0 \qquad \{x \mapsto a = \log(x)\}$$
$$\frac{\partial b}{\partial y} = \frac{\partial b}{\partial a}\frac{\partial a}{\partial y} + \frac{\partial b}{\partial y}\frac{\partial y}{\partial y} = 0 + a\frac{\partial y}{\partial y} \qquad \{(y,a) \mapsto b = ya\}$$
$$\frac{\partial c}{\partial y} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial y} = \frac{1}{2\sqrt{b}}\frac{\partial b}{\partial y} \qquad \{b \mapsto c = \sqrt{b}\}$$
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial y} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial y} = 1\frac{\partial b}{\partial y} + 1\frac{\partial c}{\partial y} \qquad \{(b,c) \mapsto f = b + c\}$$

2.4.2 Reverse Mode

Instead of evaluating the differentials for all the input nodes, which is problematic for a large number of nodes, the reverse mode evaluates the differentials of the output node w.r.t. all the inner nodes.

The method is based on a backward adjoint chain rule and initialize with the derivative of the final node

$$\frac{\partial x_t}{\partial x_t} = \mathrm{Id}_{n_t \times n_t},$$

and then from the last to the first node, iteratively make use of the following recursion formula

$$\forall\, k = t-1, t-2, \ldots, 1,$$
$$\frac{\partial x_t}{\partial x_k} = \sum_{m \,\in\, \text{son}(k)} \frac{\partial x_t}{\partial x_m} \times \frac{\partial x_m}{\partial x_k} = \sum_{m \,\in\, \text{son}(k)} \frac{\partial x_t}{\partial x_m} \times \frac{\partial}{\partial x_k} f_m(x_1, \ldots, x_m).$$

The notation "son(k)" denotes the nodes $m < k$ of the graph that are connected to k. To be clear, the same simple example.

Example. Consider again the function

$$f(x,y) = y\log(x) + \sqrt{y\log(x)}.$$

The iterations of the reverse mode are given by

$$\frac{\partial f}{\partial f} = 1$$

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial c} = \frac{\partial f}{\partial f}1 \qquad \{c \mapsto f = b + c\}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial b} + \frac{\partial f}{\partial f}\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c}\frac{1}{2\sqrt{b}} + \frac{\partial f}{\partial f}1 \qquad \{b \mapsto c = \sqrt{b},\ b \mapsto f = b + c\}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} = \frac{\partial f}{\partial b}y \qquad \{a \mapsto b = ya\}$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial y} = \frac{\partial f}{\partial b}a \qquad \{y \mapsto b = ya\}$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x} = \frac{\partial f}{\partial a}\frac{1}{x} \qquad \{x \mapsto a = \log(x)\}$$

This is the advantage of reverse mode over the forward mode. A single traversal over the computational graph allows to compute both derivatives w.r.t. x and y, while the forward mode necessities two processes.

An drawback of the reverse mode is the need to store the entire computational graph, which is needed for the reverse sweep. In principle, storage of this graph is not too difficult to implement. However, the main benefit of AD is higher accuracy, and in many applications the cost is not critical.

2.5 TMB: TEMPLATE MODEL BUILDER

Note that the goal of AD is not to define an efficient computational graph, it is up to the user to provide it. However, computing an efficient graph associated to a mathematical formula is a complicated combinatorial problem. Thus, since our goal is to be able to fit our desired statistical models, a computational tool able to efficiently define and implement this computational graph is make necessary. To solve this and many other tasks we have the Template Model Builder (TMB), developed by Kristensen et al. (2016).

TMB ⟨http://tmb-project.org⟩ is an `R` (R Core Team, 2021) package for fitting statistical latent variable models to data, inpired by AD Model Builder (ADMB) (FOURNIER et al., 2012). ADMB is a statistical application for fitting nonlinear statistical models and solve optimization problems, that implements AD using `C++` classes and a native template language. Unlike most `R` packages, in TMB the model is formulated in `C++`. This characteristic provides great flexibility but requires some familiarity with the `C/C++` programming language.

With TMB a user should be able to quickly implement complex latent effect models through simple `C++` templates. As an illustrative example let us consider an simple mixed logistic regression i.e., a binomial GLMM with a logistic link function. The latent structure is in the context of repeated measures, the same subject is observed three times. Trying to keep it simple, no covariates. A hierarchical model's description is given by

$$\begin{aligned} y_{ij} \mid u_i &\sim \text{Binomial}(n, p_{ij}) \\ u_i &\sim \text{Normal}(0,1) \\ g(p_{ij}) = \text{logit}(p_{ij}) &= \log\frac{p_{ij}}{1-p_{ij}} = \beta + u_i \\ p_{ij} &= \frac{\exp\{\beta + u_i\}}{1+\exp\{\beta + u_i\}}, \quad i \text{ the subject, } j \text{ the subject observation } (1,2,3). \end{aligned}$$

The TMB implementation of this model is provided in Figure 4.

To keep the coherence would be more adequate to fit here a multinomial model. However, I want to show you that in TMB we can also simulate data from the model but not all `r`-distributions are implemented, as is the case of the multinomial. For this reason, a binomial model was the choice. An even easier implementation of a logistic model is available in TMB, called `dbinom_robust()`, where we pass directly the logit$(\cdot)$ but we do not have an `rbinom_robust()` implementation available. Just for the sake of completeness, in Figure 5 we have the `R` code showing how to manipulate the `C++` model template in terms of object definitions and parameters estimation and extraction for the logistic mixed model.

In this chapter we describe step-by-step all the processes involved in the formulation and parameter estimation of a GLMM. With TMB all this is put in practice in an efficient and robust fashion. The user needs to provide the negative joint log-likelihood function writing in a `C++` template as exemplified in Figure 4, using specialized macros that pass the parameters, latent effects and data from `R`, as exemplified in Figure 5.

When the model presents latent effects, during the model template compilation the latent effects are integrated out via an efficient Laplace approximation routine with the inner optimization made by a Newton algorithm, and the negative marginal log-likelihood gradient is computed, via AD. The negative marginal log-likelihood is returned into an `R` object that can then be optimized using the user's favorite quasi-Newton routine, available in `R`. All these procedures are briefly exemplified for a logistic mixed model in Figure 4 and Figure 5.

To accomplish all that, TMB combines some state-of-art software

- `CppAD`, a `C++` AD package ⟨https://coin-or.github.io/CppAD/⟩;
- `Eigen` (GUENNEBAUD; JACOB et al., 2010), a `C++` templated matrix-vector library;

- CHOLMOD, C sparse matrix routines available from R, used to obtain an efficient implementation of the Laplace approximation with exact derivatives ⟨https://developer.nvidia.com/cholmod⟩;
- Parallelism through BLAS ⟨http://www.netlib.org/blas/⟩, a Fortran tuned set of Basic Linear Algebra Subprograms;

FIGURE 4 – R CODE FOR THE TMB IMPLEMENTATION OF A LOGISTIC MIXED MODEL

```
dll <- 'model'
filename <- paste0(dll, '.cpp')
writeLines({'// A LOGISTIC MIXED MODEL (RANDOM INTERCEPT)
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  // SPECIFY THE MODEL INPUTS AS DATA_
  DATA_VECTOR(y);
  DATA_SPARSE_MATRIX(Z);
  DATA_SCALAR(n);

  // SPECIFY THE MODEL PARAMETERS AND LATENT EFFECTS AS PARAMETER_
  PARAMETER(beta);
  PARAMETER(logsd);            Type sd = exp(logsd);
  PARAMETER_VECTOR(u); vector<Type> Zu = Z*u;

  // IMPLEMENT THE MODEL
  vector<Type>  risk = exp(beta+Zu);
  vector<Type> level = 1+risk;
  vector<Type>  prob = risk/level;

  // nll: NEGATIVE LOG-LIKELIHOOD
  parallel_accumulator<Type> nll(this); // DO THE MODEL IN PARALLEL
  nll -= dnorm(u, Type(0), sd, true).sum();
  nll -= dbinom(y, n, prob, true).sum();

  // TMB ALLOWS THE USER TO WRITE THE SIMULATION CODE AS AN INTEGRATED
  // PART OF THE C++ MODEL TEMPLATE
  SIMULATE {
    y = rbinom(Type(100), prob);
    REPORT(y);
  }
  // WE MODEL THE LOG STANDARD DEVIATION (logsd) BUT TMB ALLOWS US TO
  // MAKE DIRECT INFERENCE TO PARAMETER TRANSFORMATIONS, IN THIS CASE
  // THE STANDARD DEVIATION (sd)
  ADREPORT(sd);

  return nll;
}'}, con=filename)

library(TMB) ## install.packages('TMB')
TMB::compile(filename)
dyn.load(TMB::dynlib(dll)) ## loading the C++ model to R
```

SOURCE: The author (2021).

- Matrix (BATES; MAECHLER, 2019), a rich hierarchy sparse and dense matrix classes and methods using LAPACK ⟨http://www.netlib.org/lapack/⟩ and SuiteSparse ⟨https://sparse.tamu.edu/⟩ libraries.

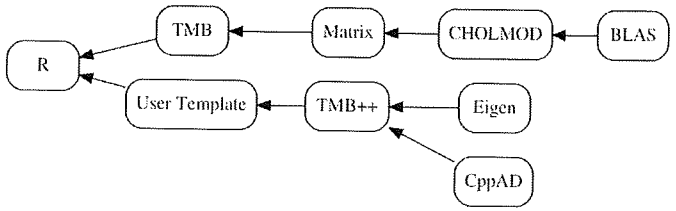
FIGURE 5 – R CODE FOR THE MODEL FITTING OF A LOGISTIC MIXED MODEL WRITTEN IN TMB

```
library(Matrix) ## install.packages('Matrix')
beta <- 2
sd   <- 1
cs   <- 3  ## cluster size
nc   <- 50 ## number of cluster
n    <- nc * cs
Z    <- Matrix::bdiag(replicate(nc, rep(1, cs), simplify=FALSE))
u    <- rnorm(nc, mean=0, sd=sd)
u0   <- numeric(nc) ## empty vector (initial guess)
risk <- exp(beta + Z %*% u)
prob <- risk/(1 + risk)
y    <- numeric(n)
## base::rbinom() is not vectorized, do a raw loop is an option
for (i in seq(y)) y[i] <- rbinom(n=1, size=n, prob=prob[i])

## building objective functions with derivatives based on the compiled
## C++ template
obj <- TMB::MakeADFun(data      =list(y=y, Z=Z, n=n),
                      parameters=list(beta=beta, logsd=log(sd), u=u0),
                      DLL       =dll,
                      random    ='u')
set.seed(1)     ## optional
obj$simulate() ## generating a simulation
## obj$simulate(complete=TRUE)
(opt <- nlminb(obj$par, obj$fn, obj$gr)) ## parameters estimation
 sdr <- TMB::sdreport(obj)               ## standard deviations
summary(sdr, select='fixed')             ## extracting model parameters
summary(sdr, select='report')            ## ... reported variables
cbind(u, summary(sdr, select='random'))  ## ... random effects
```

SOURCE: The author (2021).

An overview of the pachage design is shown in Figure 6.

FIGURE 6 – TMB PACKAGE DESIGN



SOURCE: Kristensen et al. (2016).

Reinforcing, some key characteristics are

- TMB employs AD to calculate first and second order derivatives of the log-likelihood function or of any objective function written in `C++`;
- The objective function, and its derivatives, can be called from `R`. Hence, parameter estimation via `base::optim()` or `base::nlminb()` is easy to be performed;
- Standard deviations of any parameter, or derived parameter, can be obtained via the *delta method* (Ver HOEF, 2012) implemented in `TMB::sdreport()`.

Here we focus on GLMMs, but basically any statistical model with a latent structure (or not), linear (or not), can be fitted with TMB. In times of *big data* and with the TMB's authors having a professional preference for state-space and spatial models, TMB has also automatic sparseness detection and some other nice built tools. Pre and post-processing of data should be done in `R`.

A TMB Users' mailing list exists, and it is extremely helpful for taking doubts and questions ⟨https://groups.google.com/g/tmb-users⟩. Also, a very didactic and comprehensive documentation with several examples is available online ⟨https://kaskr.github.io/adcomp/_book/Tutorial.html⟩.

3 multiGLMM: A MULTINOMIAL GLMM FOR CLUSTERED COMPETING RISKS DATA

The clustered competing risks setting is a specific survival data structure. Although, we are using a general statistical modeling framework, a generalized linear mixed model (GLMM). Consequently, the data structure characteristics have to be properly accommodated into the modeling construction.

To model competing risks data we need a multivariate model and we have to choose in which scale to work on. We may work on the hazard scale and deal with the cause-specific hazard function or on the probability scale and deal with the cause-specific cumulative incidence function (CIF). By choosing the correct link function, we are able to construct an appropriate multivariate GLMM to work on the probability scale.

Our goal is to be able to deal with complex family studies, where there is generally a strong interest in describing age at disease onset in the scenarios of within-cluster dependence. The distribution of age at disease onset is directly described by the cause-specific CIF. To build a multivariate GLMM for this type of data we need to accommodate the cause-specific CIFs and the censorings. Assuming the conditional distribution for our model response as multinomial we already deal with both left-truncation and right-censoring, avoiding the specification of a censoring distribution. The cause-specific CIFs can be modeled via the link function of our, then, multinomial GLMM (multiGLMM). The multinomial distribution also guarantees that the CIFs of all causes are modeled.

Our choice for a general framework tries to make the inference of this complex model, easier. Besides, taking advantage of all the computational procedures mentioned in the previous chapter. This chapter presents our multiGLMM for clustered competing risks data and is divided into two sections. In Section 3.1 we discuss in detail the cluster-specific cumulative incidence function (CIF) and in Section 3.2 we present the complete modeling framework.

3.1 CLUSTER-SPECIFIC CUMULATIVE INCIDENCE FUNCTION (CIF)

Consider that the observed follow-up time of an individual is given by $T = \min(T^*, C)$, where T^* denote the failure time and C denote the censoring time. Given the possible covariates x, for a cause-specific of failure k, the cumulative incidence

function (CIF) is defined as

$$F_k(t \mid \boldsymbol{x}) = \mathbb{P}[T \leq t,\, K = k \mid \boldsymbol{x}] = \int_0^t f_k(z \mid \boldsymbol{x})\, \mathrm{d}z$$
$$= \int_0^t \lambda_k(z \mid \boldsymbol{x})\, S(z \mid \boldsymbol{x})\, \mathrm{d}z, \quad t > 0, \quad k = 1, \ldots, K,$$

where $f_k(t \mid \boldsymbol{x})$ is the (sub)density for the time to a type k failure. This is the general definition of a CIF, and to define it we need to define the functions that compose the subdensity. The first is the cause-specific hazard function or process

$$\lambda_k(t \mid \boldsymbol{x}) = \lim_{h \to 0} \frac{1}{h}\, \mathbb{P}[t \leq T < t + h,\, K = k \mid T \geq t,\, \boldsymbol{x}], \quad t > 0, \quad k = 1, \ldots, K.$$

In words, the cause-specific hazard function $\lambda_k(t \mid \boldsymbol{x})$, represents the instantaneous rate for failures of type k at time t given $\boldsymbol{x}$ and all other failure types (competing causes). If we sum up all cause-specific hazard functions we get the overall hazard function,

$$\lambda(t \mid \boldsymbol{x}) = \sum_{k=1}^{K} \lambda_k(t \mid \boldsymbol{x}).$$

From the overall hazard function we arrive in the overall survival function,

$$S(t \mid \boldsymbol{x}) = \mathbb{P}[T > t \mid \boldsymbol{x}] = \exp\left\{ - \int_0^t \lambda(z \mid \boldsymbol{x})\, \mathrm{d}z \right\},$$

the second function that compose the subdensity $f_k(t \mid \boldsymbol{x})$. A comprehensive reference for all these definitions is the book of Kalbfleisch & Prentice (2002).

Until this point, we were talking about a general CIF's definition. We need now a precise framework telling us how to take into consideration our clustered/family structure. We use the same CIF specification of Cederkvist et al. (2019) i.e., the approach that motivated this thesis.

For two competing causes of failure, the cause-specific CIFs are specified in the following manner

$$F_k(t \mid \boldsymbol{x}, u_1, u_2, \eta_k) = \underbrace{\pi_k(\boldsymbol{x}, u_1, u_2)}_{\substack{\text{cluster-specific} \\ \text{risk level}}} \times \underbrace{\Phi[w_k g(t) - \boldsymbol{x}\gamma_k - \eta_k]}_{\substack{\text{cluster-specific} \\ \text{failure time trajectory}}}, \quad t > 0, \quad k = 1, 2, \quad (3.1)$$

i.e., as the product of a cluster-specific risk level with a cluster-specific failure time trajectory, resulting in a cluster-specific CIF. What makes these components cluster-specific are $\boldsymbol{u} = \{u_1, u_2\}$ and $\boldsymbol{\eta} = \{\eta_1, \eta_2\}$, Gaussian distributed latent effects with zero mean and potentially correlated i.e.,

$$\begin{bmatrix} u_1 \\ u_2 \\ \eta_1 \\ \eta_2 \end{bmatrix} \sim \text{Multivariate Normal} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2_{u_1} & \operatorname{cov}(u_1, u_2) & \operatorname{cov}(u_1, \eta_1) & \operatorname{cov}(u_1, \eta_2) \\ & \sigma^2_{u_2} & \operatorname{cov}(u_2, \eta_1) & \operatorname{cov}(u_2, \eta_2) \\ & & \sigma^2_{\eta_1} & \operatorname{cov}(\eta_1, \eta_2) \\ & & & \sigma^2_{\eta_2} \end{bmatrix} \right).$$

The cluster-specific survival function is given by $S(t \mid \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\eta}) = 1 - F_1(t \mid \boldsymbol{x}, \boldsymbol{u}, \eta_1) - F_2(t \mid \boldsymbol{x}, \boldsymbol{u}, \eta_2)$. Since we use the same CIF specification of Cederkvist et al. (2019), the following details are essentially the same encountered in the paper.

Focusing first on the second component of Equation 3.1, the cluster-specific failure time trajectory

$$\Phi[w_k g(t) - \boldsymbol{x}\gamma_k - \eta_k], \quad t > 0, \quad k = 1, 2,$$

where $\Phi(\cdot)$ is the cumulative distribution function of a standard Gaussian distribution. Instead of $w_k g(t)$, in Cederkvist et al. (2019) is specified $\alpha_k(g(t))$ with $\alpha_k(\cdot)$ being a monotonically increasing function known up to a finite-dimensional parameter vector, w_k. Examples are monotonically increasing B-splines or piecewise linear functions. However, to simplify the model structure we consider just the finite-dimensional parameter vector. The bottom line is that the authors do the same approach in their applications. With regard to the function $g(t)$, it plays a crucial role since the CIF separation in Equation 3.1 is only possible with it. It is used a time t transformation given by

$$g(t) = \operatorname{arctanh}\left(\frac{t - \delta/2}{\delta/2} \right), \quad t \in (0, \delta), \quad g(t) \in (-\infty, \infty),$$

where δ depends on the data and cannot exceed the maximum observed follow-up time τ i.e., $\delta \leq \tau$. With this Fisher-based transformation the value of the cluster-specific failure time trajectory is equal 1, at time δ. Consequently, $F_k(\delta \mid \boldsymbol{x}, \boldsymbol{u}, \eta_k) = \pi_k(\boldsymbol{x} \mid \boldsymbol{u})$ and we can interpret $\pi_1(\boldsymbol{x} \mid \boldsymbol{u})$ and $\pi_2(\boldsymbol{x} \mid \boldsymbol{u})$ as the cause-specific cluster-specific risk levels, at time δ.

The cluster-specific risk levels are modeled by a multinomial logistic regression model with latent effects i.e.,

$$\pi_k(\boldsymbol{x}, \boldsymbol{u}) = \frac{\exp\{\boldsymbol{x}\beta_k + u_k\}}{1 + \exp\{\boldsymbol{x}\beta_1 + u_1\} + \exp\{\boldsymbol{x}\beta_2 + u_2\}}, \quad k = 1, 2, \quad (3.2)$$

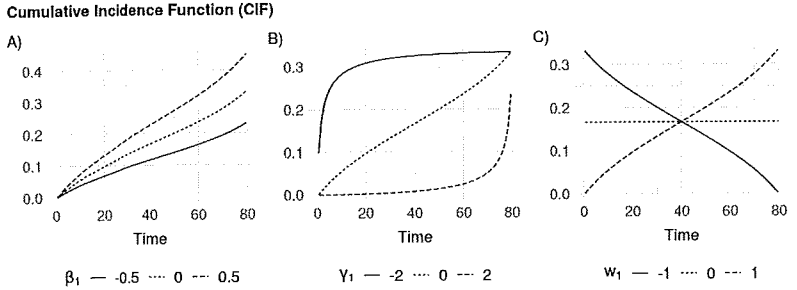
where the β_k's are the coefficients responsible for quantifying the impact of the covariates in the cause-specific risk levels. For individuals from the same cluster/family, at the same time point, the β_ks have the well-known odds ratio interpretation.

A direct understanding of all coefficients/parameters of Equation 3.1 can be reached via the illustrations in Figure 7. To really understand what is going on, we simplify the model. We still consider just two competing causes but without covariates and we plot just the cluster-specific CIF of one failure cause. In Figure 7 A) we see that the β's are also related with the curve's maximum value i.e., bigger the β, highest the CIF will be.

The γ_k's are the coefficients responsible for quantifying the impact of the covariates in the cause-specific failure time trajectories i.e., the shape of the cumulative

incidence. In Figure 7 B) we see that the γ's are also related with an idea of midpoint and consequently, growth speed. The fact that γ_k enters negatively in the cluster-specific failure time trajectory makes that a negative value causes an advance towards the curve, whereas a positive value causes a delay. Last but not least, the w's in Figure 7 C). With negative values, we have a decreasing curve and with positive values an increasing curve i.e., we are interested only on the positive side.

FIGURE 7 – ILLUSTRATION OF COEFFICIENT BEHAVIORS FOR A GIVEN CUMULATIVE INCIDENCE FUNCTION (CIF) (PROPOSED BY Cederkvist et al. (2019)), IN A MODEL WITH TWO COMPETING CAUSES OF FAILURE, WITHOUT COVARIATES, AND WITH THE FOLLOWING CONFIGURATION: $\beta_2 = 0$, $u = 0$ AND $\eta = 0$; IN EACH SCENARIO ALL OTHER COEFFICIENTS ARE SET TO ZERO, WITH THE EXCEPTION OF $w_1 = 1$



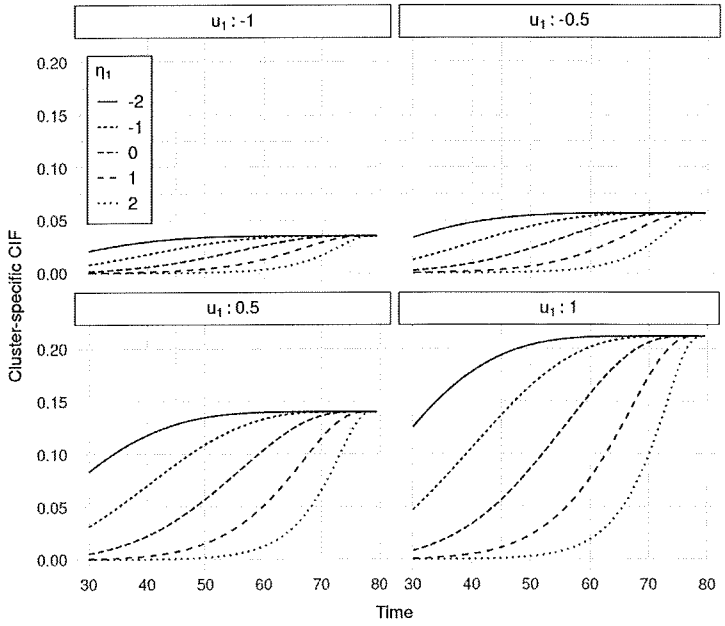
SOURCE: The author (2021).

Remains to talk about the within-cluster dependence induced by the latent effects in u and η. Unfortunately, they do not have an easy interpretation. To help in the discussion, Figure 8 illustrates the cluster-specific CIF for a given failure cause in a model without covariates, let us call it failure cause 1 (in total we have two).

The latent effects u_1 and u_2 always appear together in the cluster-specific risk level, as consequency they have a joint effect on the cumulative incidence of both causes. As we can see in Figure 8, an increase in u_k will increase the risk of failure from cause k. The interpretation of $\text{cov}(\eta_1, \eta_2)$ and $\text{cov}(u_1, u_2)$ is straightforward, and those values are in most of the cases positive, as said in Cederkvist et al. (2019). With regard to $\text{cov}(u_k, \eta_k)$, negative values are the common situation. A negative correlation between η_k and u_k imply that when η_k decreases, u_k increases and conversely when η_K increases, u_k decreases. In other words, an increased risk level is reached quickly and a decreased risk level is reached later, respectively.

Practical situations with a positive within-cause correlation are hard to find i.e., where an increased risk level is associated with a late onset and vice versa. However, a positive cross-cause correlation between η and u sounds much more realistic i.e., where late onset of one failure cause is associated with a high absolute risk of another failure cause.

FIGURE 8 – ILLUSTRATION OF A GIVEN CLUSTER-SPECIFIC CUMULATIVE INCIDENCE FUNCTION (CIF), PROPOSED BY Cederkvist et al. (2019), IN A MODEL WITH TWO COMPETING CAUSES OF FAILURE, WITHOUT COVARIATES AND THE FOLLOWING CONFIGURATION: $\beta_1 = -2$, $\beta_2 = -1$, $\gamma_1 = 1$, $w_1 = 3$ AND $u_2 = 0$. THE VARIATION BETWEEN FRAMES IS GIVEN BY THE LATENT EFFECTS u_1 AND η_1



SOURCE: The author (2021).

The latent effects $\{u_k, \eta_k\}$ are assumed independent across clusters and shared by individuals within the same cluster/family.

3.2 MODEL SPECIFICATION

The multiGLMM for clustered competing risks data is specified in the following hierarchical fashion. By simplicity, we focus on two competing causes of failure but an extension is straightforward.

For two competing causes of failure, a subject i, in the cluster/family j, in time

t, we have

$$y_{ijt} \mid \{u_{1j},\, u_{2j},\, \eta_{1j},\, \eta_{2j}\} \sim \text{Multinomial}(p_{1ijt},\, p_{2ijt},\, p_{3ijt})$$

$$\begin{bmatrix} u_1 \\ u_2 \\ \eta_1 \\ \eta_2 \end{bmatrix} \sim \text{MN}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2_{u_1} & \text{cov}(u_1,\, u_2) & \text{cov}(u_1,\, \eta_1) & \text{cov}(u_1,\, \eta_2) \\ & \sigma^2_{u_2} & \text{cov}(u_2,\, \eta_1) & \text{cov}(u_2,\, \eta_2) \\ & & \sigma^2_{\eta_1} & \text{cov}(\eta_1,\, \eta_2) \\ & & & \sigma^2_{\eta_2} \end{bmatrix} \right)$$

$$\begin{aligned} p_{kijt} &= \frac{\partial}{\partial t} F_k(t \mid x,\, u_1,\, u_2,\, \eta_k) \\ &= \frac{\exp\{x_{kij}\beta_k + u_{kj}\}}{1 + \sum_{m=1}^{K-1} \exp\{x_{mij}\beta_m + u_{mj}\}} \\ &\quad \times w_k \frac{\delta}{2\delta t - 2t^2}\, \phi\left(w_k \text{arctanh}\left(\frac{t - \delta/2}{\delta/2} \right) - x_{kij}\gamma_k - \eta_{kj} \right), \end{aligned} \tag{3.3}$$

$k = 1,\, 2.$

The probabilities are given by the derivative w.r.t. time t of the cluster-specific CIF. The choice of a multinomial logistic regression model ensures that the sum of the predicted cause-specific CIFs does not exceed 1.

Considering two competing causes of failure, we have a multinomial with three classes. The third class exists to handle the censorship and its probability is given by the complementary to reach 1. This framework in Equation 3.3 results in what we call multiGLMM, a multinomial GLMM to handle the CIF of clustered competing risks data. For a random sample, the corresponding marginal likelihood functions in given by

$$\begin{aligned} L(\theta\,;\, y) &= \prod_{j=1}^{J} \int_{\Re^4} \pi(y_j \mid r_j) \times \pi(r_j)\, \mathrm{d}r_j \\ &= \prod_{j=1}^{J} \int_{\Re^4} \underbrace{\left\{ \prod_{i=1}^{n_j} \prod_{t=1}^{n_{ij}} \left(\frac{(\sum_{k=1}^{K} y_{kijt})!}{y_{1ijt}!\, y_{2ijt}!\, y_{3ijt}!} \prod_{k=1}^{K} p_{kijt}^{y_{kijt}} \right) \right\}}_{\text{fixed effect component}} \times \\ &\qquad \underbrace{(2\pi)^{-2} |\Sigma|^{-1/2} \exp\left\{ -\frac{1}{2} r_j^\top \Sigma^{-1} r_j \right\}}_{\text{latent effect component}} \mathrm{d}r_j \\ &= \prod_{j=1}^{J} \int_{\Re^4} \underbrace{\left\{ \prod_{i=1}^{n_j} \prod_{t=1}^{n_{ij}} \prod_{k=1}^{K} p_{kijt}^{y_{kijt}} \right\}}_{\text{fixed effect}} \underbrace{(2\pi)^{-2} |\Sigma|^{-1/2} \exp\left\{ -\frac{1}{2} r_j^\top \Sigma^{-1} r_j \right\}}_{\text{latent effect component}} \mathrm{d}r_j, \end{aligned} \tag{3.4}$$

where $\theta = [\beta\ \gamma\ w\ \sigma^2\ \rho]^\top$ is the parameters vector to be maximized. In our framework, a subject can fail from just one competing cause or get censor, at a given time. Thus, the

fraction of factorials in the fixed effect component is made only by 0's and 1's. Finally, returning the value 1. The matrix Σ is the variance-covariance matrix, which parameters are given by σ^2 and ρ.

Now, Equation 3.4 in words. To each cluster/family j we have a product of two components. The fixed effect component, given by a multinomial distribution with its probabilities specified through the cluster-specific CIF (Equation 3.1) and, the latent effect component, given by a multivariate Gaussian distribution.

To each subject i that composes a cluster j we have its specific fixed effects contribution. The likelihood in Equation 3.4 is the most general as possible, allowing for repeated measures to each subject. Since all subjects of a given cluster shares the same latent effect, we have just one latent effect contribution multiplying the product of fixed effect contributions. As we do not observe the latent effect variables, r_j, we integrate out in it. With two competing causes of failure, we have four latent effects (a multivariate Gaussian distribution in four dimensions). Consequently, for each cluster, we approximate an integral in four dimensions. The product of these approximated integrals results in the called marginal likelihood, to be maximized in θ.

Independent of the parameters value choice, the probabilities for the failure causes will be small, providing always a bigger probability mass to the censorship. More will be talked about this model characteristic in the next chapter.

3.2.1 Parametrization

We have to choose in which terms we parameterize the variance-covariance matrix Σ. Besides the latent effects variances $\{\sigma^2\}$, we have to choose if we will estimate its covariances or correlations. By the name *variance-covariance* matrix, it is natural to think on covariance terms. However, this option is not very attractive since its interpretation is not clear. A more attractive choice is in terms of correlation.

The covariance between two terms is defined as a triple product: the two terms standard deviations times the correlation ρ. Still thinking in two competing causes of failure, we have then an Σ matrix with six correlations

$$\Sigma = \begin{bmatrix} \sigma^2_{u_1} & \rho_{u_1,u_2}\sigma_{u_1}\sigma_{u_2} & \rho_{u_1,\eta_1}\sigma_{u_1}\sigma_{\eta_1} & \rho_{u_1,\eta_2}\sigma_{u_1}\sigma_{\eta_2} \\ & \sigma^2_{u_2} & \rho_{u_2,\eta_1}\sigma_{u_2}\sigma_{\eta_1} & \rho_{u_2,\eta_2}\sigma_{u_2}\sigma_{\eta_2} \\ & & \sigma^2_{\eta_1} & \rho_{\eta_1,\eta_2}\sigma_{\eta_1}\sigma_{\eta_2} \\ & & & \sigma^2_{\eta_2} \end{bmatrix}.$$

With the matrix parametrization being chosen, we have that the parameters to be estimated are the components of the vector $\theta = [\beta\ \gamma\ w\ \sigma^2\ \rho]^\top$. There we have the fixed effects or mean components $\{\beta\ \gamma\ w\}$, the easiest to estimate in a statistical modeling framework; we have variance components $\{\sigma^2\}$, the intermediate ones; and the correla-

tion components $\{\rho\}$, the hardest ones. This idea of easy or hard to estimate may be justified by three, connected, arguments.

The first comes from the fact that we are modeling the mean of a probability distribution in a hierarchical and structured fashion, consequently, the easiest parameters to estimate will be the mean components. We may make the analogy that to estimate the mean parameters we need data (resources); to estimate the variance parameters we need more data (more resources), and to estimate the correlation parameters we need much more data (even more resources). The second argument comes also to explain the first one via the parametric space constraints.

Generally, the fixed effect components do not present constraints i.e., they can vary in all $\mathbb{R}$. The same can not be said from the variance components, constrained by definition into the $\mathbb{R}_*^+$. Finally, we have the correlation components, constrained to the interval $[-1, 1]$. These parametric space constraints drive us again to the first argument since we need more data/resources/information to be able to estimate coefficients constrained to some interval. Nevertheless, this may not be enough. Without providing some extra information in terms of an e.g., constrained algorithm, it is very reasonable to expect that during the optimization procedure some unrealistic areas of the parametric space could be visited and jeopardize the stability or even the whole optimization procedure. To overcome these possible difficulties, parameter reparametrizations are more than welcome.

The variance and correlation parameters are modeled in terms of the matrix Σ. This matrix is symmetric and more important, positive semi-definite. This last characteristic is also the third argument to justify why is so difficult to estimate these parameters. Since the estimates should lead to a positive semi-definite matrix, the employment of a parametrization is welcome to enforces this condition.

In the subject of choosing the components parametrization for a positive-definite matrix Σ, we have basically two big options available in the statistical modeling literature. One of them consists of just transform the scale. By practical reasons, let us think in a 2×2 matrix

$$\Sigma = \begin{bmatrix} \exp\{\log\sigma_1^2\} & z^{-1}(z(\rho_{1,2}))\sqrt{\exp\{\log\sigma_1^2\}}\sqrt{\exp\{\log\sigma_1^2\}} \\ & \exp\{\log\sigma_2^2\} \end{bmatrix}$$

i.e., in the main diagonal we may now estimate the log-variances and in the off-diagonal we may estimate Fisher z-transformed correlations.

The estimation of the log-variances has two big advantages:

- Since the natural logarithm is a real-valued function, we overcome the parametric space constraint problem;

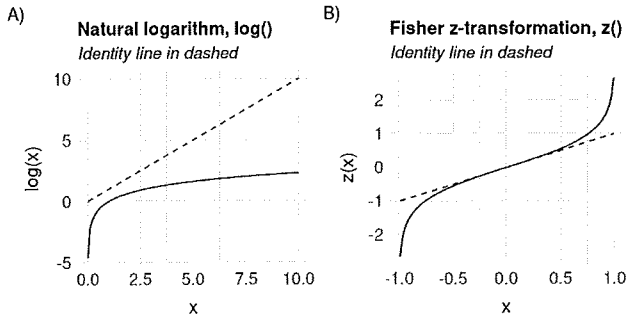
- High variances are problematic for many reasons but in the context of seeing them as the diagonal components of a restricted matrix, being able to control its magnitudes is a crucial task to the stability of any optimization routine. With the natural logarithm transformation we shrink the parametric space as illustrated in Figure 9 A), avoiding some eventual numerical cumbersome.

With the correlation components we proceed with the estimation of its Fischer z-transformation. This transformation, and its inverse, are defined as

$$z(\rho) = \frac{1}{2}\log\left(\frac{1+\rho}{1-\rho}\right) = \operatorname{arctanh}(\rho), \quad z^{-1}(\rho) = \frac{\exp\{2\rho\}-1}{\exp\{2\rho\}+1} = \tanh(\rho).$$

The Fisher z-transformation plays the role of stretching the small correlation parametric space but doing this in a smooth fashion, as illustrated in Figure 9 B).

FIGURE 9 – ILLUSTRATION OF THE PARAMETRIZATION BEHAVIOR FOR THE VARIANCE COMPONENTS, IN A), AND CORRELATION COMPONENTS, IN B)



SOURCE: The author (2021).

The other parametrization option consist in estimate the elements of a factorization or decomposition of the positive-definite matrix Σ. The most common is the Cholesky factorization or decomposition (PINHEIRO; BATES, 1996). For two competing causes of failure, a standard Cholesky decomposition of Σ may be expressed as

$$\Sigma = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ c_2 & c_3 & 0 & 0 \\ c_4 & c_5 & c_6 & 0 \\ c_7 & c_8 & c_9 & c_{10} \end{bmatrix} \begin{bmatrix} c_1 & c_2 & c_4 & c_7 \\ 0 & c_3 & c_5 & c_8 \\ 0 & 0 & c_6 & c_9 \\ 0 & 0 & 0 & c_{10} \end{bmatrix} = LL^\top,$$

where $\{c_i\}_{i=1}^{10}$ are the unconstrained coefficients to be estimated.

A disadvantage in the use of a decomposition as the Cholesky is the lack of a straightforward interpretation to the elements $\{c_i\}_{i=1}^{10}$. However, with the application of the delta method, already implemented in TMB's (KRISTENSEN et al., 2016)

`TMB::sdreport()`, it is straightforward to get back the Σ elements together with its respective standard errors. The main advantage of this parametrization apart from the fact that it ensures positive definiteness, is that it is computationally simple and stable.

Just to mention another viable possibilities, we could use a modified Cholesky decomposition (POURAHMADI, 2007) providing a better statistical interpretation of the decomposition elements or, we could parametrize the precision matrix, $Q = \Sigma^{-1}$. Since we use Σ^{-1} in the marginal likelihood of Equation 3.4, parametrizing directly its inverse save us some computations.

Besides the popularity of the Cholesky method, there is another factorization scheme available and efficiently implemented in TMB. It is a factorization based on a vector scale transformation of an unstructured correlation matrix. For two competing causes of failure the decomposition is specified in the following fashion

$$\Sigma = VD^{-1/2}LL^{\top}D^{-1/2}V^{\top},$$

where

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ c_1 & 1 & 0 & 0 \\ c_2 & c_3 & 1 & 0 \\ c_4 & c_5 & c_6 & 1 \end{bmatrix}, \quad D = \text{diag}(LL^{\top}) \quad \text{and} \quad W = \text{diag}\left(\{\sigma_i\}_{i=1}^{4}\right).$$

This scheme is based initially on the factorization of a correlation matrix defined as $D^{-1/2}LL^{\top}D^{-1/2}$. The elements $\{c_i\}_{i=1}^{6}$ to be estimated have the advantage of being unconstrained and guarantees the matrix symmetry and positive definiteness. The variances are scaled via the diagonal matrix V, its elements $\{\sigma_i\}_{i=1}^{4}$ are then the standard deviations to be estimated.

4 SIMULATION STUDY

This chapter describes how to simulate from our multiGLMM and describes the performed simulation studies. The general simulation procedure is addressed in Section 4.1. In Section 4.2 the simulation studies are presented in detail.

4.1 SIMULATING FROM THE MODEL

Being able to simulate data from a model is a key task, fundamental to assess the finite-sample properties and the estimation procedure liability of a given statistical model. The step-by-step describing the simulation procedure of our multiGLMM is presented on Algorithm 1, following the model hierarchical structure stipulated in Equation 3.3.

ALGORITHM 1 SIMULATING FROM A multiGLMM FOR CLUSTERED COMPETING RISKS DATA

1: Set J, the number of clusters
2: Set n_j, the number of cluster elements ▷ can be of different sizes
3: Set $K-1$, the number of competing causes of failure
4: Set the model parameter values $\boldsymbol{\theta} = [\beta\ \gamma\ w\ \sigma^2\ \varrho]^{\top}$
5: Sample J latent effect vectors from a Multivariate Normal$_{(K-1)\times(K-1)}(\mathbf{0}, \Sigma(\sigma^2, \varrho))$
6: Set δ ▷ maximum follow-up time
7: Set the failure times t_{ij}
8: Compute the competing risks probabilities

$$p_{kij} = \frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}}{1 + \sum_{m=1}^{K-1}\exp\{x_{mij}\beta_{mj} + u_{mj}\}} \times w_k \frac{\delta}{2\delta t_{ij} - 2t_{ij}^2}\, \phi\left(w_k \text{arctanh}\left(\frac{t_{ij} - \delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}\right),$$

$$\text{Censorship}: \quad p_{Kij} = 1 - \sum_{k=1}^{K-1} p_{kij}, \quad k = 1, 2, \ldots, K-1$$

9: Sample $J \times n_j$ vectors from a Multinomial$(p_{1ij}, p_{2ij}, \ldots, p_{Kij})$
10: If $t_{ij} = \delta$, moves to class K ▷ any failure at time δ is a censorship
11: **return** multinomial vectors and their respective failure/censoring times

SOURCE: The author (2021).

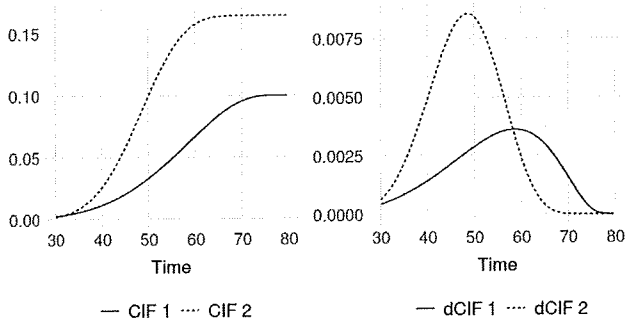
The model described in Equation 3.3 is in a general form, allowing for varying coefficients between clusters. However, we focus on a simpler structure with just fixed

intercepts. Fixing the latent effects in its distribution mean, zero, and using the following fixed effects configuration for two competing causes of failure

$$\begin{aligned} \beta &= [-2 \;\; 1.5]^\top \\ \gamma &= [1.2 \;\; 1]^\top \\ w &= [3 \;\; 5]^\top, \end{aligned} \tag{4.1}$$

we get the CIF's and failure probabilities (CIF derivatives w.r.t. time t, dCIF) presented respectively in Figure 10.

FIGURE 10 – CUMULATIVE INCIDENCE FUNCTIONS (CIF) AND RESPECTIVE DERIVATIVES (dCIF) W.R.T. TIME FOR A MODEL WITH TWO COMPETING CAUSES OF FAILURE, WITHOUT COVARIATES, LATENT EFFECTS IN ZERO, AND FIXED EFFECTS IN Equation 4.1



SOURCE: The author (2021).

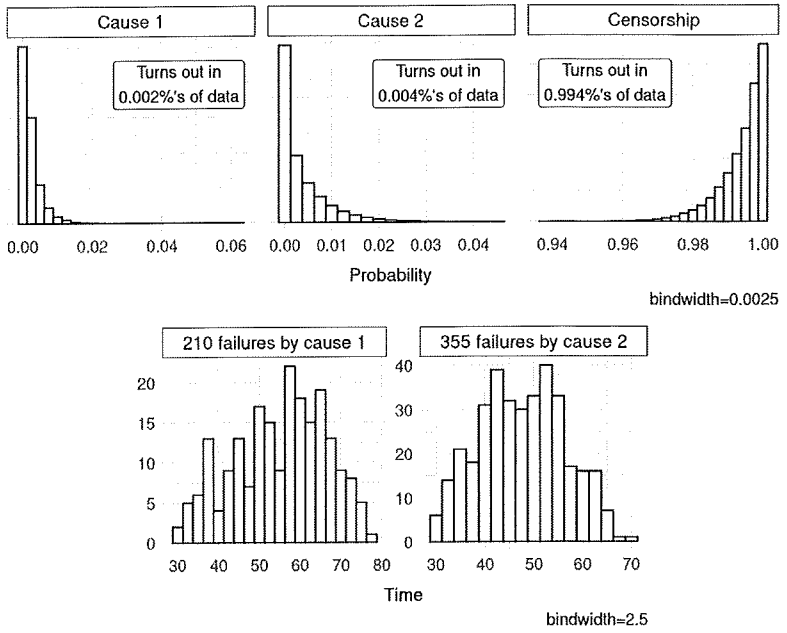
By adding a complete latent structure,

$$\begin{bmatrix} u_1 \\ u_2 \\ \eta_1 \\ \eta_2 \end{bmatrix} \sim \text{Multivariate Normal} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.4 & -0.5 & 0.4 \\ & 1 & 0.4 & -0.3 \\ & & 1 & 0.4 \\ & & & 1 \end{bmatrix} \right), \tag{4.2}$$

we are able to apply Algorithm 1 and generate a complete model sample with 50000 clusters of size two (pairs), summarized in Figure 11. The R function written to simulate the data is available in Appendix C.

Varying the parameters configuration we are able to build basically any CIF's format. However, its dCIF will be always small i.e., the generated probabilities for the failure causes will always be (really) small, passing all the probability mass to the censorship class. Low probabilities imply few failures, making the multiGLMM even harder to fit. All these behaviors are seen in Figure 11.

FIGURE 11 – HISTOGRAMS FOR SIMULATED PROBABILITIES WITH RESPECTIVE OUTPUT PCERCENTAGES AND FAILURE TIMES FOR A MODEL WITH TWO COMPETING CAUSES AND 50000 CLUSTERS OF SIZE TWO. THE SIMULATION FOLLOWED ALGORITHM 1 GUIDELINES WITH PARAMETER CONFIGURATIONS SPECIFIED IN Equation 4.1 AND Equation 4.2



SOURCE: The author (2021).

Thinking of epidemiological or public health problems, the reasonable CIF behaviors are the ones presented in Figure 10. In the simulation routine, the failure/-censorship times are based on the random sampling of values between 30 and 80-time units. Another approach could be performing this random sampling by age group, to have something similar to a realistic population pyramid. However, by performing some tests we saw that having a realistic CIF is enough. Even with a uniform time distribution, the failure times will not be uniform, as the CIF curves impose their form. We can see this happening in the bottom graphs of Figure 11.

Cederkvist et al. (2019) does something different through the sampling of the censorship times from a U(0, δ), the sampling of $\varsigma \sim$ U(0, 1), and the computation of the cause-specific failure times by solving

$$\varsigma = \Phi\left(w_k \text{arctanh}\left(\frac{t_{ij} - \delta/2}{\delta/2} \right) - x_{kij}\gamma_k - \eta_{kj} \right) \quad \text{for} \quad t_{ij},$$

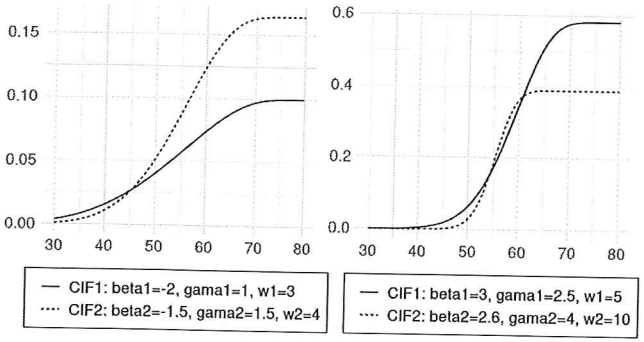
with i being the subject, j the cluster, and k the failure cause. This approach implies a parametric form for the failure times, which we do not know if holds in the real world.

4.2 SIMULATION STUDIES DESIGN

To stress the model and check the parameters identifiability, we propose seventy-two scenarios. All scenarios with two competing causes. Thus, a three classes multinomial distribution.

We consider two CIF scenarios, in summary, a low and a high CIF scenario. The parameter configurations are presented together with their curves in Figure 12. As mentioned before, independent of the configuration the censorship level is basically the same.

FIGURE 12 – CUMULATIVE INCIDENCE FUNCTIONS (CIF) FOR A MODEL WITH TWO COMPETING CAUSES OF FAILURE, WITHOUT COVARIATES, AND LATENT EFFECTS IN ZERO



SOURCE: The author (2021).

We use four multiGLMMs, that can be discriminated by their latent effect structures

$$\sigma^2 = [\sigma^2_{u_1} = 1 \quad \sigma^2_{u_2} = 0.6 \quad \sigma^2_{\eta_1} = 0.7 \quad \sigma^2_{\eta_2} = 0.9];$$

$$\rho = [\rho_{u_1,u_2} = 0.1 \quad \rho_{u_1,\eta_1} = -0.5 \quad \rho_{u_1,\eta_2} = 0.3 \quad \rho_{u_2,\eta_1} = 0.3 \quad \rho_{u_2,\eta_2} = -0.4 \quad \rho_{\eta_1,\eta_2} = 0.2].$$

In

$$\Sigma = \begin{bmatrix} R & C \\ C^\top & T \end{bmatrix} \quad \text{with}$$

$$R = \begin{bmatrix} \sigma^2_{u_1} & \text{cov}(\rho_{u_1,u_2}) \\ & \sigma^2_{u_2} \end{bmatrix}, \quad T = \begin{bmatrix} \sigma^2_{\eta_1} & \text{cov}(\rho_{\eta_1,\eta_2}) \\ & \sigma^2_{\eta_2} \end{bmatrix}, \quad C = \begin{bmatrix} \text{cov}(\rho_{u_1,\eta_1}) & \text{cov}(\rho_{u_1,\eta_2}) \\ \text{cov}(\rho_{u_2,\eta_1}) & \text{cov}(\rho_{u_2,\eta_2}) \end{bmatrix}.$$

They are:

risk model A model with latent effects only on the risk level i.e., $\Sigma_{2\times2} = R$;

time model A model with latent effects only on the trajectory time level i.e., $\Sigma_{2\times2} = T$;

block-diag model A model with latent effects on the risk and trajectory time level only i.e., $\Sigma_{4\times4} = \text{diag}(R, T)$;

complete model A model with a *complete* latent effects structure i.e, $\Sigma_{4\times4} = \Sigma$.

All models are based on some decomposition of the symmetric matrix

$$\Sigma = \begin{bmatrix} R & C \\ C^\top & T \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0775 & -0.4183 & 0.2846 \\ & 0.6 & 0.1944 & -0.2939 \\ & & 0.7 & 0.1587 \\ & & & 0.9 \end{bmatrix}.$$

Propose a 4×4 matrix like this is not so simple since we should not only look at the values coherence. The matrix should also be positive-definite, and more than that, the submatrices R, T, and $\text{diag}(R, T)$, should also be positive-definite since they are also used as variance-covariances matrices.

The inherent idea of these four models is to able to check if a latent structure on both CIF levels is really necessary. To accomplish this goal we propose the simpler versions i.e., a latent structure only on the risk and only on the trajectory time. More than just check the necessity of a complete latent specification, we have to check if we are able to properly estimate/learn all the covariance parameters. Thinking in these tasks, we have the block-diag and the complete model specifications, and also the CIF level configurations. Are we able to properly estimate the covariance parameters independent of the CIF level? Do we really need or can we estimate the cross-correlations? Those are the kind of questions that motivate the simulation study and consequently, the scenario specifications.

Together with the model specifications and CIF configurations, we have three sample sizes, 5000; 30000; and 60000 data points. They are combined with three cluster sizes, size 2; size 5; and size 10 clusters. Thus,

5000 data points,	*30000 data points,*	*60000 data points,*
• 2500 clusters of size 2	• 15000 clusters of size 2	• 30000 clusters of size 2
• 1000 clusters of size 5	• 6000 clusters of size 5	• 12000 clusters of size 5
• 500 clusters of size 10	• 3000 clusters of size 10	• 6000 clusters of size 10

Besides the sample size per se, by increasing it we also increase the number of Laplace approximations to be performed since each cluster implies a Laplace approximation. This affects hugely the computational time. About the failure/censorship times, we work on a grid between 30 and 80 years. The times are sampled from a Uniform distribution on that grid, as explained in Section 4.1. In summary, we have

Two CIF configurations, four latent effect structures, three sample sizes, and three cluster sizes; $2 \times 4 \times 3 \times 3 = 72$ scenarios. For each scenario, we simulate 300 samples and fitted the corresponding model. Thus, $72 \times 300 = 21600$ models/fits.

In Figure 13 we have the C++ code for the TMB implementation of the complete model.

FIGURE 13 – C++ CODE FOR THE TMB IMPLEMENTATION OF A multiGLMM WITH A COMPLETE 4x4 LATENT STRUCTURE

```
// multiGLMM: A MULTINOMIAL GLMM FOR CLUSTERED COMPETING RISKS DATA
// COMPLETE 4x4 LATENT STRUCTURE (COMPLETE MODEL)
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  using namespace density;
  DATA_MATRIX(Y);
  DATA_SPARSE_MATRIX(Z);
  DATA_VECTOR(time);
  DATA_SCALAR(delta);
  PARAMETER(beta1);
  PARAMETER(beta2);
  PARAMETER(gama1);
  PARAMETER(gama2);
  PARAMETER(w1);
  PARAMETER(w2);

  PARAMETER(logs2_1); Type s2_1=exp(logs2_1);
  PARAMETER(logs2_2); Type s2_2=exp(logs2_2);
  PARAMETER(logs2_3); Type s2_3=exp(logs2_3);
  PARAMETER(logs2_4); Type s2_4=exp(logs2_4);

  PARAMETER(rhoZ12); Type rho12=(exp(2*rhoZ12)-1)/(exp(2*rhoZ12)+1);
  PARAMETER(rhoZ13); Type rho13=(exp(2*rhoZ13)-1)/(exp(2*rhoZ13)+1);
  PARAMETER(rhoZ14); Type rho14=(exp(2*rhoZ14)-1)/(exp(2*rhoZ14)+1);
  PARAMETER(rhoZ23); Type rho23=(exp(2*rhoZ23)-1)/(exp(2*rhoZ23)+1);
  PARAMETER(rhoZ24); Type rho24=(exp(2*rhoZ24)-1)/(exp(2*rhoZ24)+1);
  PARAMETER(rhoZ34); Type rho34=(exp(2*rhoZ34)-1)/(exp(2*rhoZ34)+1);

  PARAMETER_MATRIX(U); matrix<Type> ZU=Z*U;
  Type risk1=0;
  Type risk2=0;
  Type level=0;
  // gt=atanh(2*time/delta-1); atanh(x)=0.5*log((1+x)/(1-x))
  vector<Type> gt=0.5*log(time/(delta-time));
  vector<Type> dgt=delta/(2*time*(delta-time));
  Type x1=0;
  Type x2=0;
  vector<Type> y(Y.cols());
  vector<Type> prob(Y.cols());
  parallel_accumulator<Type> nll(this);
  // Type nll=0;
  vector<Type> u(U.cols());

  Type cov12=rho12*sqrt(s2_1)*sqrt(s2_2);
  Type cov13=rho13*sqrt(s2_1)*sqrt(s2_3);
  Type cov14=rho14*sqrt(s2_1)*sqrt(s2_4);
  Type cov23=rho23*sqrt(s2_2)*sqrt(s2_3);
  Type cov24=rho24*sqrt(s2_2)*sqrt(s2_4);
  Type cov34=rho34*sqrt(s2_3)*sqrt(s2_4);
  matrix<Type> Sigma(4, 4);
  Sigma.row(0) << s2_1, cov12, cov13, cov14;
  Sigma.row(1) << cov12, s2_2, cov23, cov24;
  Sigma.row(2) << cov13, cov23, s2_3, cov34;
  Sigma.row(3) << cov14, cov24, cov34, s2_4;

  MVNORM_t<Type> dmvnorm(Sigma);
  for (int i=0; i<U.rows(); i++) {
    u=U.row(i);
    nll += dmvnorm(u);
  }
  for (int i=0; i<Y.rows(); i++) {
    risk1=exp(beta1 + ZU(i, 0));
    risk2=exp(beta2 + ZU(i, 1));
    level=1 + risk1 + risk2;
    x1=w1*gt(i) - gama1 - ZU(i, 2);
    x2=w2*gt(i) - gama2 - ZU(i, 3);
    prob(0)=risk1/level * w1*dgt(i) * dnorm(x1, Type(0), Type(1), false);
    prob(1)=risk2/level * w2*dgt(i) * dnorm(x2, Type(0), Type(1), false);
    prob(2)=1 - prob(0) - prob(1);
    y=Y.row(i);
    nll -= dmultinom(y, prob, true);
  }
  ADREPORT(s2_1);
  ADREPORT(s2_2);
  ADREPORT(s2_3);
  ADREPORT(s2_4);
  ADREPORT(rho12);
```

```
  ADREPORT(rho13);
  ADREPORT(rho14);
  ADREPORT(rho23);
  ADREPORT(rho24);
  ADREPORT(rho34);
  REPORT(Sigma);
  return nll;
}
```

SOURCE: The author (2021).

Since the other three models can be seen as special cases of the complete model, we show their TMB's implementation in the Appendix D (Section D.1, Section D.2, Section D.3). In the Appendix D (Section D.4), we also have the R code showing how to load and fit the models. It is a brief piece of code, which just shows how simple is to work with TMB.

5 RESULTS

This chapter presents the simulation study results. We have seventy-two simulation scenarios, as detailed in Chapter 4. For each scenario we simulate 300 samples. In total, we fit 21600 models.

5.1 SIMULATION STUDY

Let us just recap the parameter values used

$$\text{High CIF configuration}: \quad \{\beta_1 = -2,\ \beta_2 = -1.5,\ \gamma_1 = 1,\ \gamma_2 = 1.5,\ w_1 = 3,\ w_2 = 4\};$$
$$\text{Low CIF configuration}: \quad \{\beta_1 = 3,\ \beta_2 = 2.5,\ \gamma_1 = 2.6,\ \gamma_2 = 4,\ w_1 = 5,\ w_2 = 10\}.$$

$$\begin{matrix} \sigma^2_{u_1} = 1 \\ \sigma^2_{u_2} = 0.7, \\ \sigma^2_{\eta_1} = 0.6 \\ \sigma^2_{\eta_2} = 0.9 \end{matrix} \qquad \text{Correlation structure} = \begin{pmatrix} 1 & 0.1 & -0.5 & 0.3 \\ & 1 & 0.3 & -0.4 \\ & & 1 & 0.2 \\ & & & 1 \end{pmatrix} \begin{matrix} u_1 \\ u_2 \\ \eta_1 \\ \eta_2 \end{matrix}.$$

(columns: u_1, u_2, η_1, η_2)

The parameter values per se are not important. What is important is to keep in mind the behaviors implied by them, and see if the proposed model is able to learn the true values in several different scenarios and measure the quality of this learning.

The take-home message for the fixed-effect parameters, is to show that we can construct different level CIF scenarios. The βs are responsible for the curve maximum point or plateau, the γs and ws are responsible for basically the curve shape. Its interpretation is presented in detail in Chapter 3. About the latent-effects, the chosen covariance structure is considerably high but still acceptable. The underlying idea was to try to build a realistic covariance scenario and consequently be able to check how the model performs in such conditions.

In the following pages we have several graphs summarizing the parameters bias. In each figure, we have the parameter bias and its uncertainty described by a Wald-based interval i.e., ± 1.96 the bias standard deviation. This is a good uncertainty representation choice since it is symmetric and robust to outliers. In the Appendix D, we have the same parameters bias but with the corresponding 2.5 and 97.5% quantiles. We chose to use these uncertainty representations uniquely based on the point estimates instead of the standard error computations. In several scenarios, the model fails to compute all the standard errors, caused by Hessian instability problems.

The seventy-two scenarios are accommodated. We have up to four blocks of bars, each block representing a model. In each block we have eighteen bars, each bar representing the 300 fits in each of the eighteen scenarios, $4 \times 18 \times 300 = 21600$.

Each scenario name consists of a combination of three strings

- The cluster size (cs), 2, 5, and 10;
- The CIF configuration, high and low;
- The sample size, 5, 30, and 60 thousand.

We have tried to fit a total of 21600 models but not all converged. Besides that inconvenience, from the ones that converged do not all converged in the strict sense of the word. To show these two characteristics respectively, we control the bar widths and colors. The ideal convergence is the traditional one i.e., based on the gradient. However, in complicated optimization scenarios as the one we are here, we can reach a convergence based on a metric different from the gradient one.

In the `base::nlminb()` R's PORT implementation, when the default gradient-based convergence is reached we get a 0 flag message. When the optimization converges but not gradient-based, we get a 1 flag message. Looking at the technical report of Gay (1990), we see that this different convergence means that the last algorithm iterate x appears to be at a certain scaled distance of the locally optimal point x^*. This scaled distance, $p(x,x^*)$, is defined by

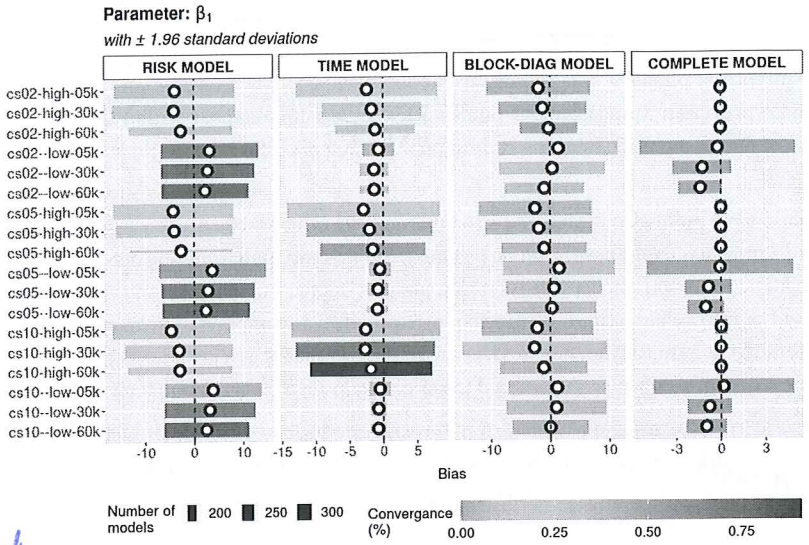
$$p(x,y) = \max_{1 \leq i \leq p} \{d_i |x_i - y_i|\} \Big/ \max_{1 \leq j \leq p} \{d_j |x_j| + |y_j|\},$$

where d is a certain scale vector. In all figures, the presented convergence percentage is from the gradient-based one.

Something specific can be said about each parameter but let us keep the focus on the general remarks. Starting from the fixed-effect parameters in Figure 14, Figure 15, Figure 16, Figure 17, Figure 18, and Figure 19, we have very nice results that already show a strong inclination towards the complete model's choice.

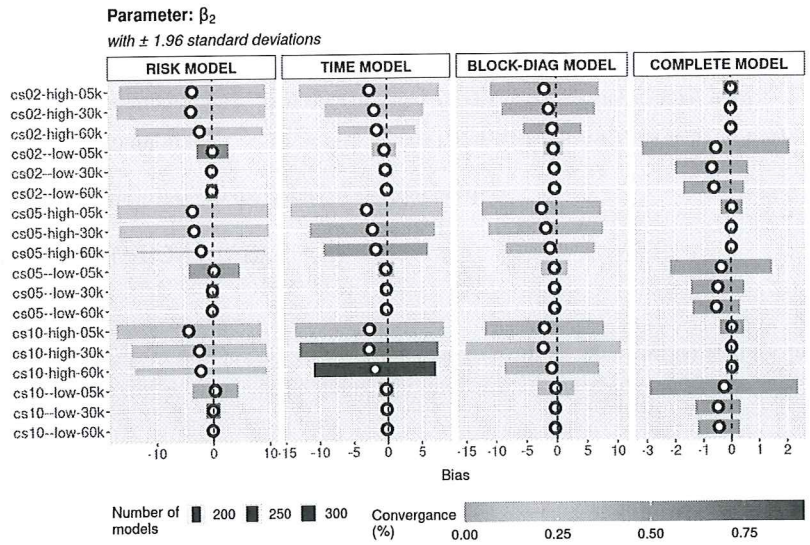
With a latent structure only in the risk level or in the trajectory time level, the low CIF scenarios are the ones with a much smaller bias-variance. In general, the mean-bias is small but the variances are high. When we have a latent structure on both levels but we still assume the cross-correlations as zero (block-diag model), the results get a little bit better. Nevertheless, when we assume a non-zero cross-correlation structure (complete model), basically everything changes for the better. The mean biases get even closer to zero, the standard deviations decrease 50% or more, and mainly, now the high CIF scenarios are the ones with a much smaller bias-variance. All this is accomplished through the consideration of the cross-correlations.

FIGURE 14 – PARAMETER β_1 BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



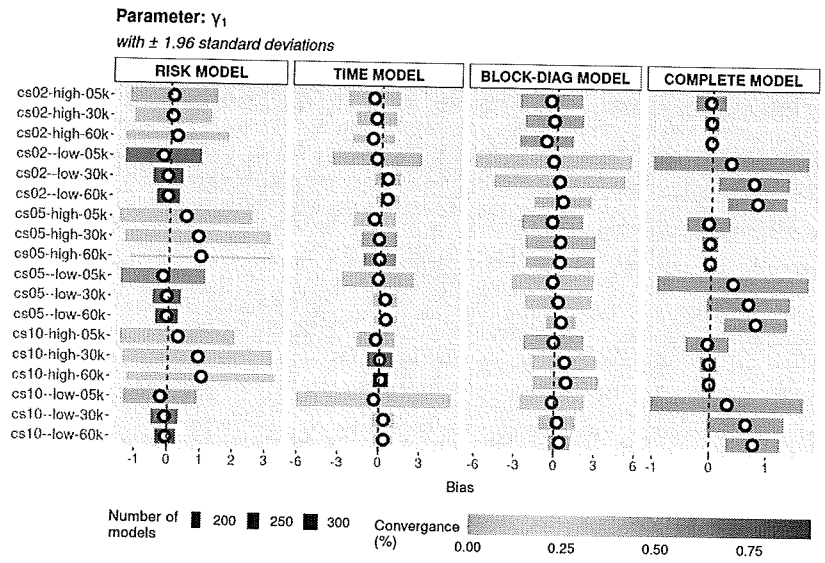
SOURCE: The author (2021).

FIGURE 15 – PARAMETER β_2 BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



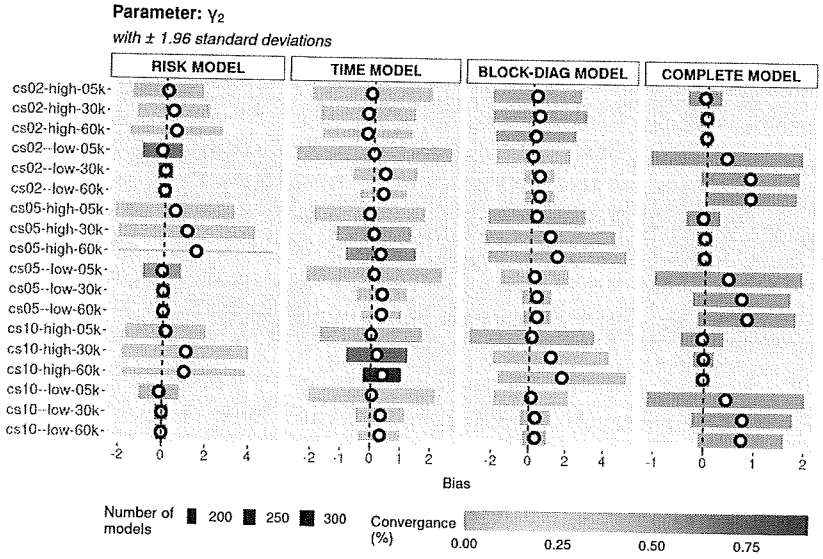
SOURCE: The author (2021).

FIGURE 16 – PARAMETER γ_1 BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



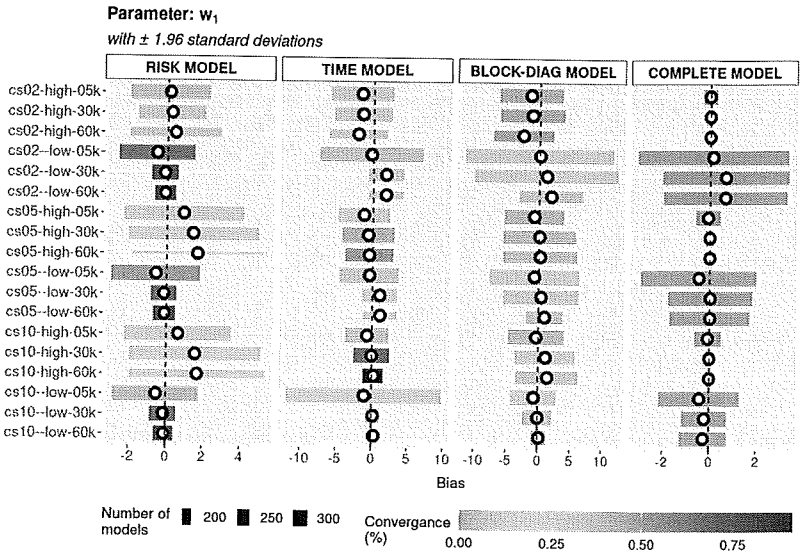
SOURCE: The author (2021).

FIGURE 17 – PARAMETER γ_2 BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



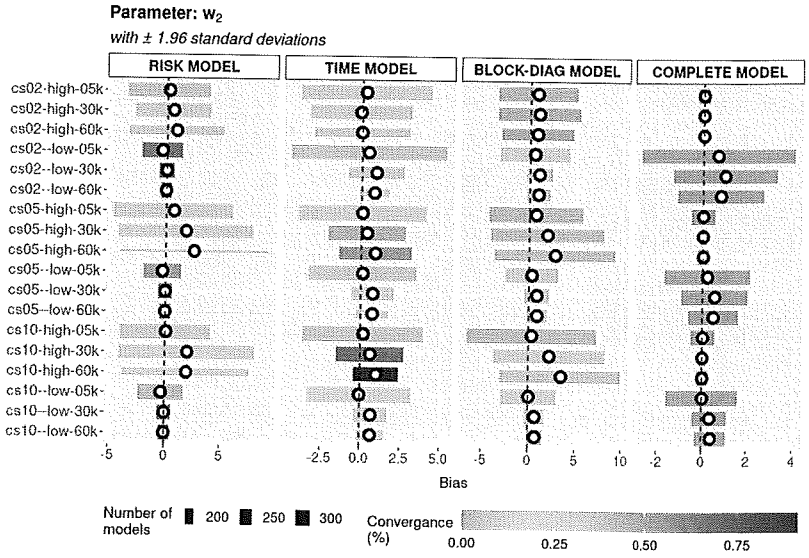
SOURCE: The author (2021).

FIGURE 18 – PARAMETER w_1 BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



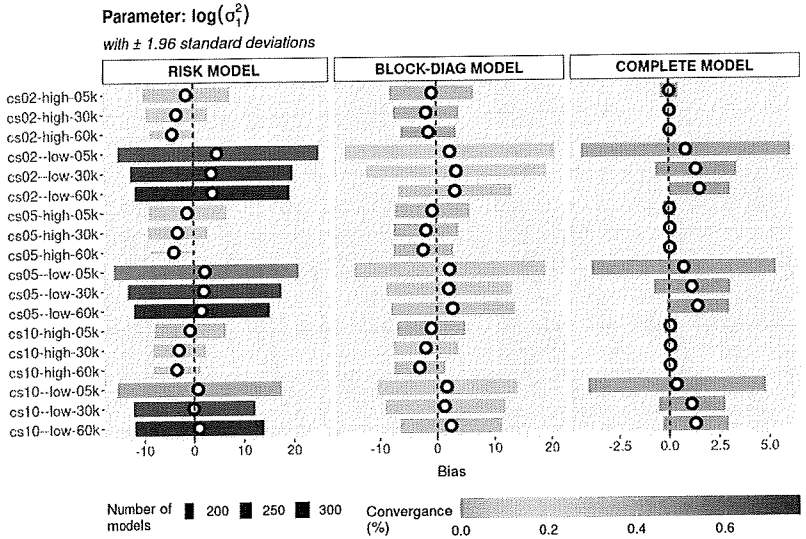
SOURCE: The author (2021).

FIGURE 19 – PARAMETER w_2 BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



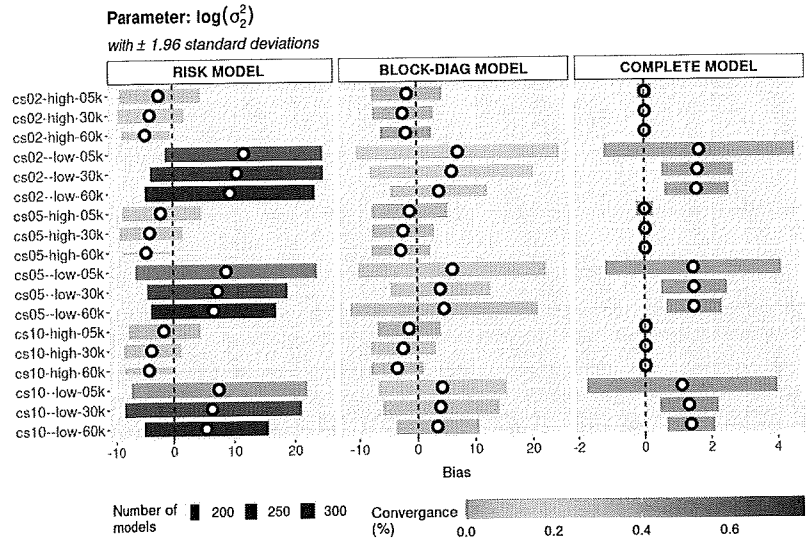
SOURCE: The author (2021).

FIGURE 20 – PARAMETER $\log(\sigma_1^2)$ BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



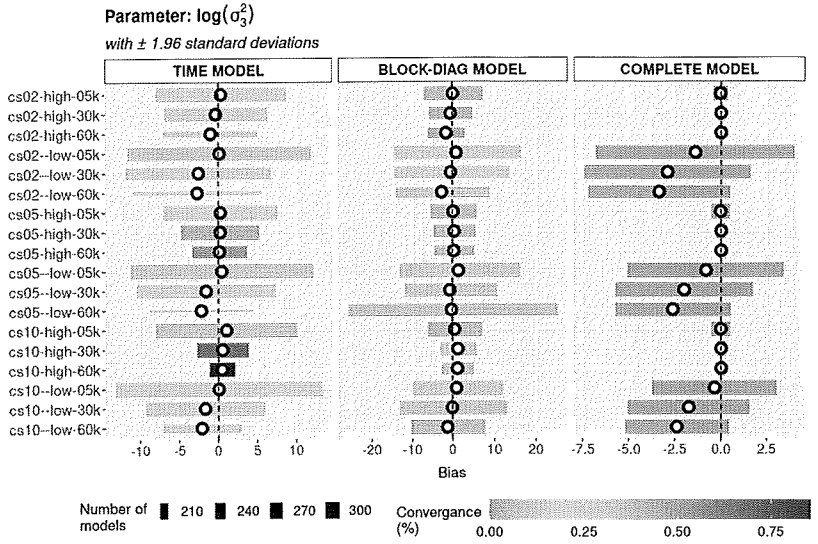
SOURCE: The author (2021).

FIGURE 21 – PARAMETER $\log(\sigma_2^2)$ BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



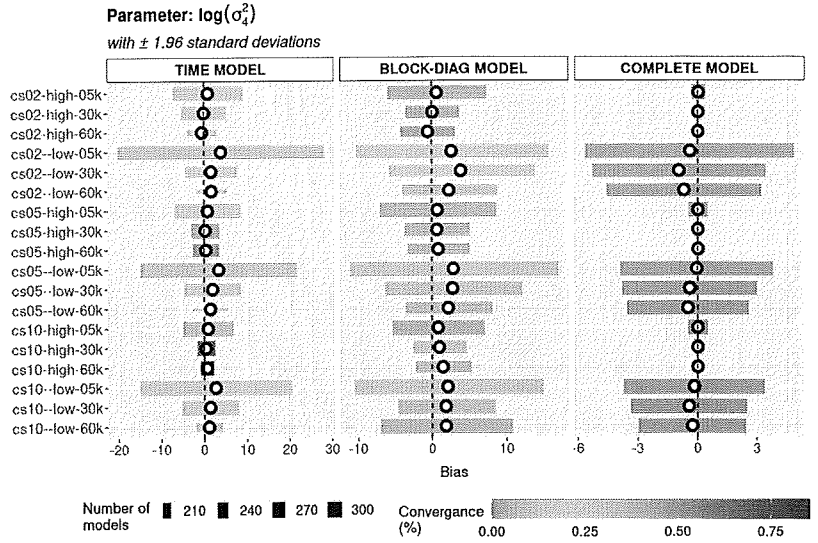
SOURCE: The author (2021).

FIGURE 22 – PARAMETER $\log(\sigma_3^2)$ BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



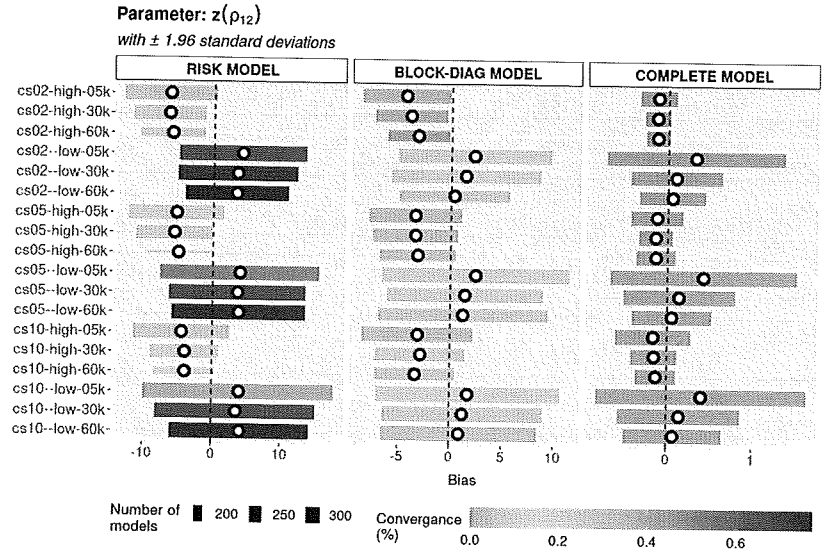
SOURCE: The author (2021).

FIGURE 23 – PARAMETER $\log(\sigma_4^2)$ BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



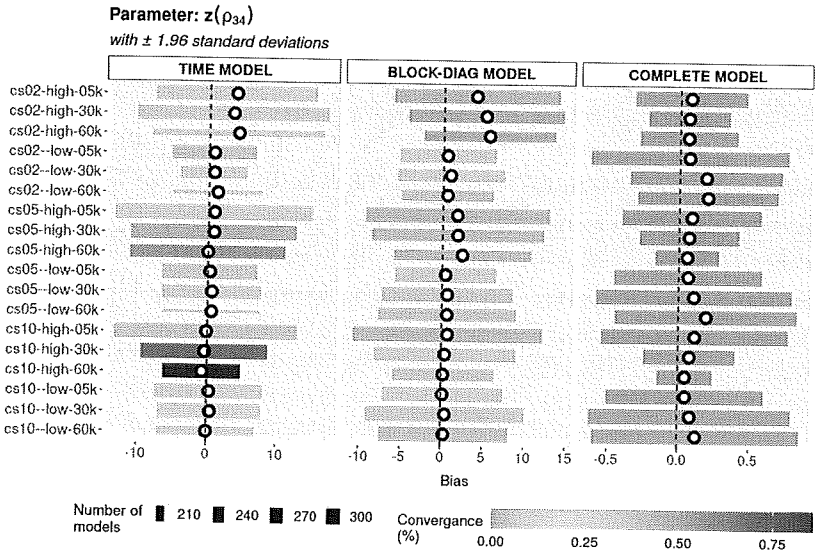
SOURCE: The author (2021).

FIGURE 24 – PARAMETER $z(\rho_{12})$ BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



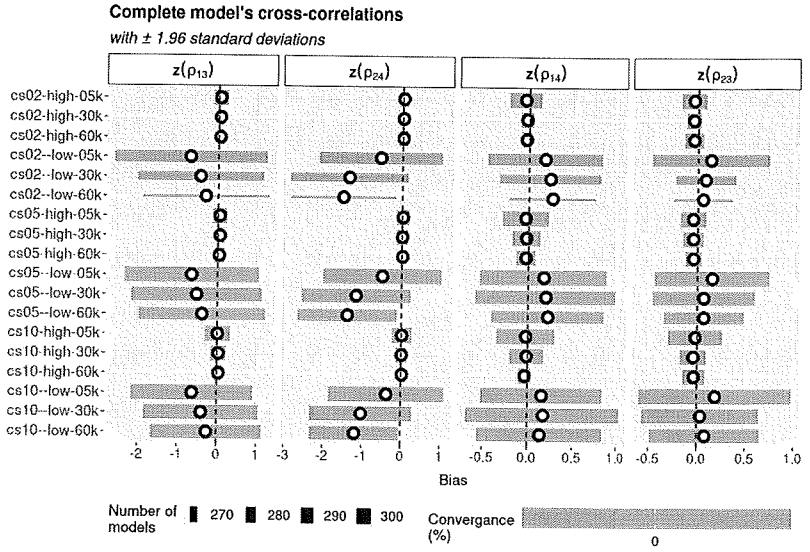
SOURCE: The author (2021).

FIGURE 25 – PARAMETER $z(\rho_{34})$ BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



SOURCE: The author (2021).

FIGURE 26 – PARAMETERS $\{z(\rho_{13}), z(\rho_{24}), z(\rho_{14}), z(\rho_{23})\}$ BIAS WITH $\pm$ 1.96 STANDARD DEVIATIONS



SOURCE: The author (2021).

In the *simpler* models, with a latent structure just in one level, is hard to see some significant difference between the clusters and sample sizes. With the complete model, in the other hand, the difference is clear: as we increase the clusters and the sample sizes, the bias-variance decreases. The mean-bias is basically always the same. In the risk and block-diag models is hard to point-out a scenario as the best or worst. For the time model, with the scenario of cluster size 10; low CiF; and 5 thousand data points, we get a much bigger standard deviation in two of the parameters.

With the log-variances presented in Figure 20, Figure 21, Figure 22, and Figure 23, we have instead a similar behavior through the models. For all the models, the high CIF scenarios are the ones with a smaller mean and bias-variances. From the risk/time model to the block-diag model, we do not see a significant improvement in terms of bias reduction. Such improvement, however, is clear when we look at the complete model. Again, the magick of considering the cross-correlations.

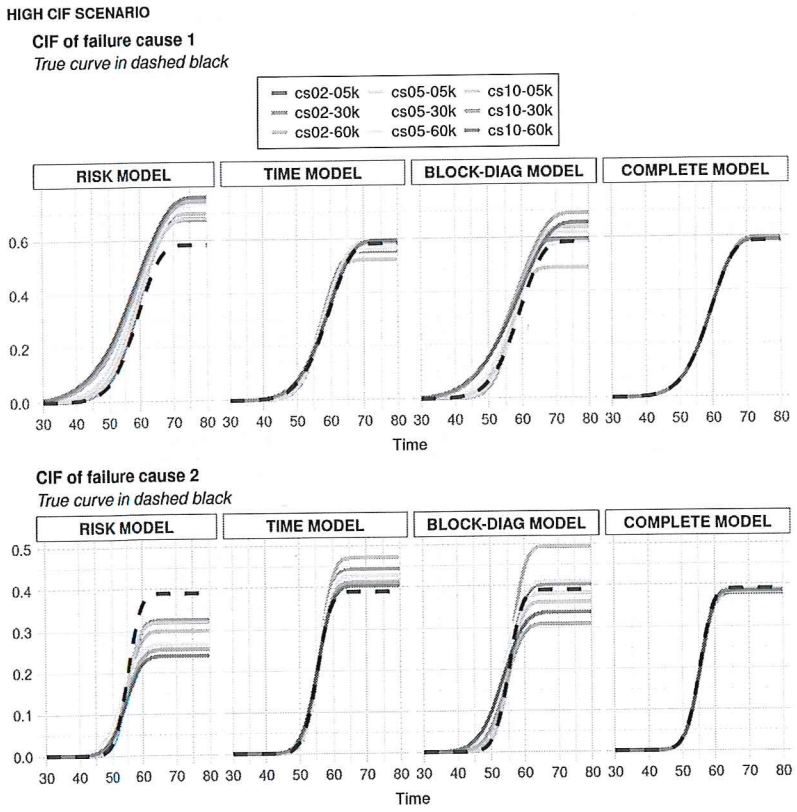
The same said about the log-variances, can be applied to the risk correlations in Figure 24, with one addendum: the bias reduction is even bigger. With the time correlation in Figure 25, we get the same behavior observed with the fixed-effect parameters i.e., with the simpler models, the smaller biases are observed in the low CIF scenarios. However, with the complete model, we get the opposite. With the cross-correlations in

Figure 26, the mean and bias-variances are much smaller in the high CIF scenarios.

The biggest bias-variances are obtained in the log-variances. A final remark to be made is about convergences. With the simpler models, not all of them work, having in some scenarios (generally the ones with 60 thousand data points) a 50∼60% convergence rate. However, from this 50∼60%, most reach the ideal gradient-based convergence. With the complete model is the complete opposite. Basically, all fits reach a convergence (∼100% performance) but never is the desired gradient-based convergence.

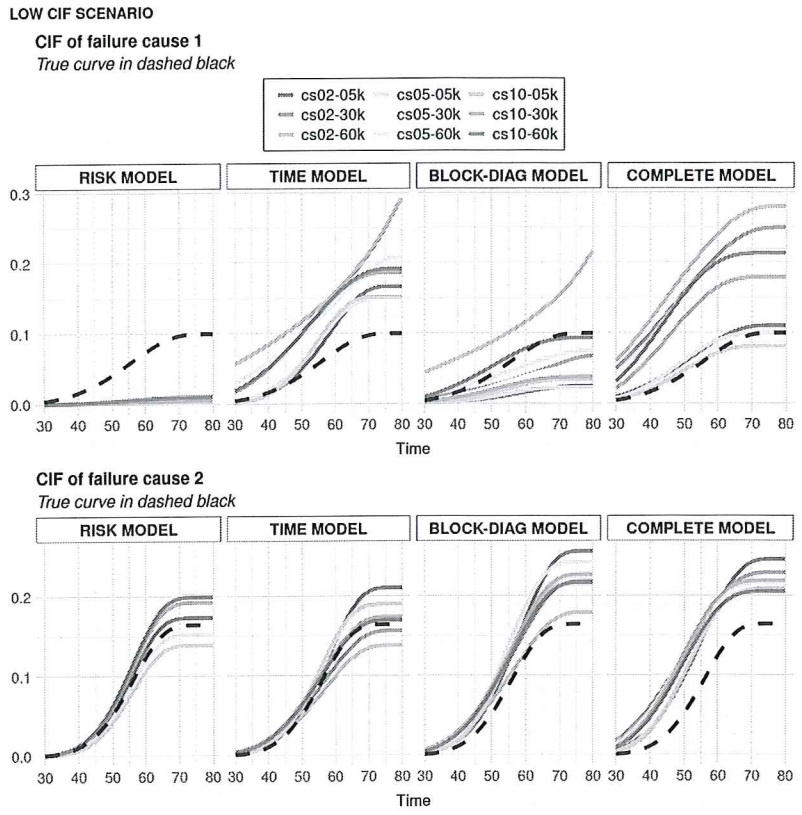
After looking at the parameter biases, let us take a look at the implied mean-CIF curves. To nicely accommodate all seventy-two scenarios we split the curves by level-CIF. In Figure 27 we have the high CIF scenario curves and in Figure 28 the low CIF scenario curves.

FIGURE 27 – HIGH CUMULATIVE INCIDENCE FUNCTION (CIF) SCENARIO CURVES



SOURCE: The author (2021).

FIGURE 28 – LOW CUMULATIVE INCIDENCE FUNCTION (CIF) SCENARIO CURVES



SOURCE: The author (2021).

Since for all the models we have a latent structure for the within-cluster dependency, the inherent idea is that this also affect the fixed-effect parameter estimates. By taking its average in each of the seventy-two scenarios, we are able to construct the mean CIF curves.

In Figure 27 we have all the thirty-six curves obtained in the high CIF scenarios. It is clear that with the complete model we get a perfect fit in all nine scenarios. The risk and time models estimate well the curve shape parameters but they fail to learn the max incidence. A compensation between curves is clear. In the risk model, there is a super estimation of β_1 in all scenarios. For failure cause 2, there is a sub estimation. With the time model, we observe the opposite compensation but on a smaller scale.

With the time model, we get much better curves than with the risk model. The block-diag model results are a middle term between them. For the time model, the scenario with cluster size 10 and 60 thousand data points is a highlight. For the block-diag model, the highlight is the scenario with cluster size 5 and 30 thousand data points.

In the low CIF scenarios in Figure 28, the estimation is clearly more difficult. The overall fits are bad, being impossible to select a scenario with overall good results. For one of the failure causes, the estimation quality is not so bad. The problem is when we look to the other. An interesting scenario is the one with cluster size 2 and 60 thousand data points. In this scenario we see the worst fits for failure cause 1, with a negative highlight in the block-diag configuration. However, with this same model, for failure cause 2, it is the only scenario, by far, were we learn the true curve. An interesting compensation phenomena. The best joint fit is still with the complete model.

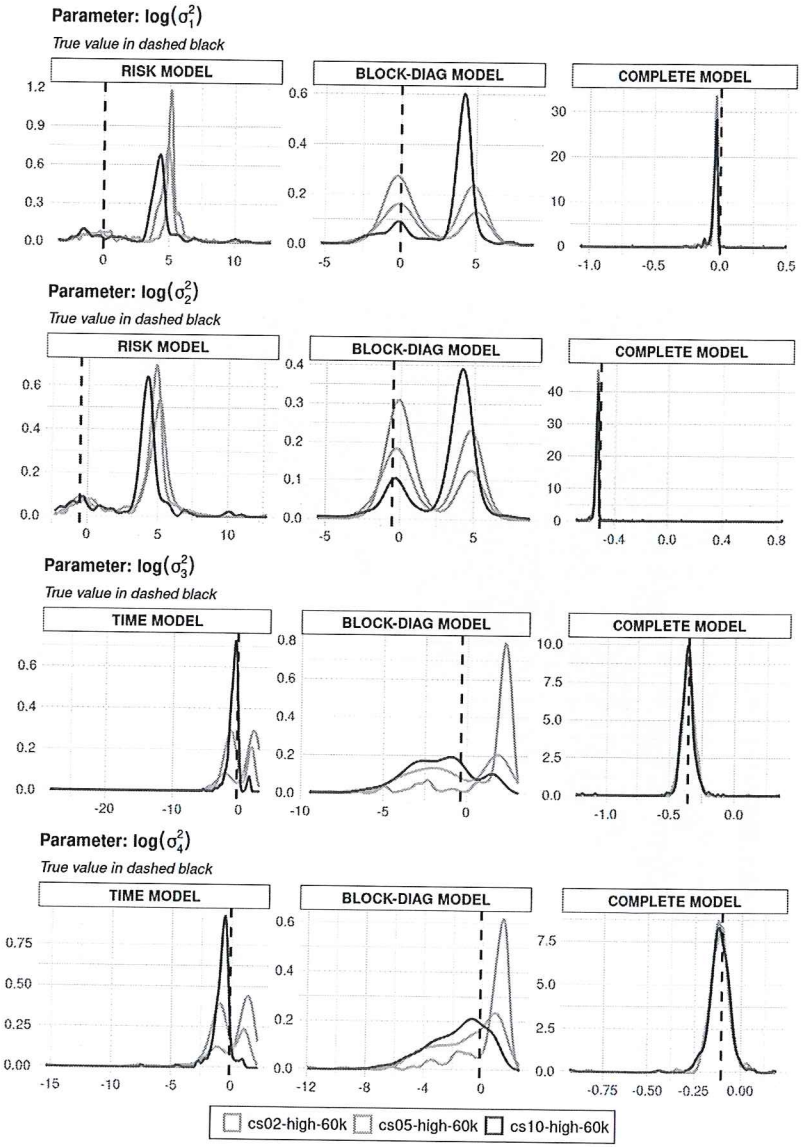
Now we look at how the latent-effect parameter estimates distribute themselves. Given the huge number of scenarios and the fact that is harder to estimate covariance parameters, we chose to plot the parameters just in the scenarios with better performances. By the metrics of small bias and CIF shape learning, the scenarios with better results are the ones with high CIF and bigger sample sizes. We have the densities for the variance parameters, in each of these scenarios, presented in Figure 29. In Figure 30 we have the same for the correlation parameters.

An interesting result is the clear difference between risk and time models' covariance parameters. With the risk model, we have an evident super estimation and bigger variances. With the time model we get much better results, but still with high variances. The block-diag model generally performs better than the risk model and worst than the time model, showing again to be a compromise between them. Besides the bias itself, we should also pay attention to the values. We model the variances in the log-scale, so a value 5, in reality, implies a variance of $\exp(4) = 148$. Terrible. This kind of problem do not sound to appear with the complete model.

All correlations are quite well estimated, in all three scenarios, with the complete model. Not only the correlations but the variances also. The lack of any considerable difference between the covariance densities, indicates no quality divergences in the results for different cluster sizes. The densities in Figure 29 and Figure 30 are the final corroboration indicating the good estimability of the complete model. Between the four tested models, the complete model was the one with the smallest biases, better CIF shape learning, and precisest covariance parameter estimates.

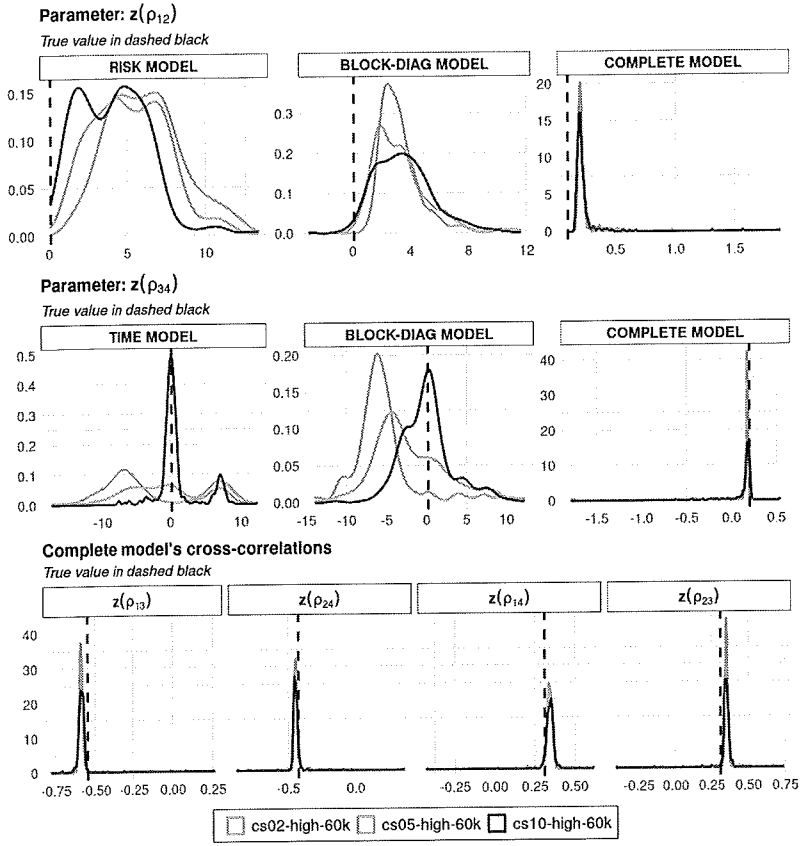
In Figure 31 we have a heat-map of the correlations between parameter estimates for the complete model in the scenario with clusters of size 10, high CIF, and 60 thousand data points.

FIGURE 29 – VARIANCE PARAMETERS DENSITIES IN THE SCENARIOS OF HIGH CIF AND 60 THOUSAND DATA POINTS



SOURCE: The author (2021).

FIGURE 30 – CORRELATION PARAMETERS DENSITIES IN THE SCENARIOS OF HIGH CIF AND 60 THOUSAND DATA POINTS



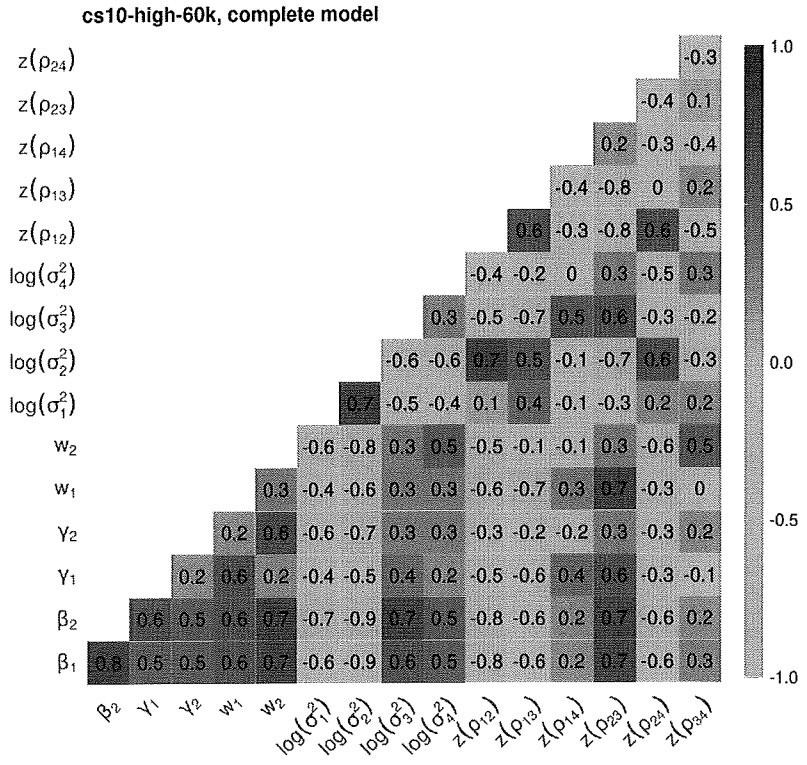
SOURCE: The author (2021).

We have a little bit of everything in Figure 31. Some correlations are very close to zero, but we also have strong positive and negative correlations. We can mention some curiosities, but nothing pathological appears to happen, at least nothing clear.

All fixed-effect parameters are positive correlated, with an emphasis on the correlation between β_1 and β_2, and the one of the βs with the ws. Another interesting observation is the negative correlation between the βs and the risk log-variances, and also the positive correlation between the βs and the trajectory time log-variances. The risk log-variances are positively correlated. So do the time trajectory ones. The

correlations between the log-variances of different levels are negative.

FIGURE 31 – COMPLETE MODEL'S PARAMETERS CORRELATION HEAT-MAP IN THE SCENARIO OF CLUSTER SIZE 10, HIGH CIF, AND SIXTY-THOUSAND DATA POINTS



SOURCE: The author (2021).

6 FINAL CONSIDERATIONS

The general goal of this master thesis was the proposition of a new regression model for the analysis of clustered competing risks data. Focused on the probability scale, by means of the cumulative incidence function (CIF), instead of the hazard scale usual on the survival modeling literature (KALBFLEISCH; PRENTICE, 2002). We model the clustered competing risks on a latent-effects framework, a generalized linear mixed model (GLMM) (MCCULLOCH; SEARLE, 2001), with a multinomial distribution for the competing risks and censorship, conditioned on the latent-effects. The within-cluster latent dependency is accommodated by a multivariate Gaussian distribution and is modeled via its covariance matrix parameters.

The failures by the competing causes and their respective censorships are modeled in the probability scale, by means of the CIF (KALBFLEISCH; PRENTICE, 2002; ANDERSEN et al., 2012). The CIF is accommodated in our GLMM framework in terms of the link function (MCCULLAGH; NELDER, 1989), as the product of two functions, one responsible to model the instantaneous risk and the other the trajectory time. The shape of these functions is described in detail in Chapter 3. This particular GLMM formulation is what makes our model, particular. Thus, we have what we call a multiGLMM: a multinomial GLMM for clustered competing risks data.

The two-function product CIF formulation was taken from Cederkvist et al. (2019) but there they use a different framework, a composite likelihood framework (LINDSAY, 1988; COX; REID, 2004; VARIN; REID; FIRTH, 2011). Here we do a full likelihood analysis instead. A composite approach is generally used when a full likelihood approach is impossible or computationally impracticable. Our goal here was to ~~try~~ a full likelihood framework taking advantage of state-of-the-art computational libraries and very efficient algorithm implementations. We have all this with the `R` (R Core Team, 2021) package TMB (KRISTENSEN et al., 2016).

In Chapter 5 our main results are presented, and it is clear that the multiGLMM works under certain circumstances. The next step was to compare our results with the ones obtained in Cederkvist et al. (2019), with the composite approach. In the GitHub repository ⟨https://github.com/kkholst/mcif/⟩ the authors provide their code. In `mcif/inst/examples/datasim.R` they show how to simulate from the model, and in `mcif/src/loglik.cpp` they have their marginal log-likelihood function. We tried to optimize their marginal log-likelihood over its parameters using basically all `R` `base::optim()` and `base::nlminb()` available methods, in the paper was used the BFGS, one of them. We made several scenarios, using their own simulation scripts and ours, and to our surprise, the model basically does not work.

The optimization in its majority fails, via any gradient-based algorithm (BFGS (NOCEDAL; WRIGHT, 2006), PORT (GAY, 1990; DENNIS; GAY; WELSCH, 1981), conjugate gradient (CG) (FLETCHER; REEVES, 1964)), generally by Hessian matrix instability problems, a problem which our multiGLMM also suffers from when we try to compute the parameters standard errors. When the model works, it is because we are using the parameter true values as initial guesses i.e. if the algorithm needs to walk on the log-likelihood surface following the gradient, it fails. Even when it works, the estimates are not always good. We also tried with a SANN and a Nelder-Mead algorithm. SANN (BELISLE, 1992) is a variant of a simulated annealing method, based on a Metropolis algorithm. Since it is based on simulation, it takes a lot of time and as the gradient-based methods, do not work most of the time. The best results were with the Nelder-Mead (NELDER; MEAD, 1965), a free-gradient method. Still, it only works when we use the parameter true values as initial guesses. This situation is completely the opposite of what is shown in the paper, making impossible any reasonable comparison between the models. We will enter in contact with the authors to see what is happening.

All models from the simulation study were run, in a parallelized fashion, in one of the two following Linux systems:

System 1 12 Intel (R) Core (TM) i7-8750H CPU @ 2.20GHz processors with 16GB RAM;

System 2 30 Intel (R) Xeon (R) CPU E5-2690 v2 @ 3.00GHz processors and 206GB RAM.

Each risk and time model run is not so time-consuming, generally never taking more than 5 minutes. The inherent idea is that for each cluster we are always performing two-dimension integral approximations and we have *just* three covariance parameters. With the block-diag model, we are theoretically in four dimensions. However, since the covariance matrix is, block-diagonal, we experienced several numerical instability problems. The solution, as can be seen in the Section D.3 (Appendix D) code, was to split it into two two-dimension matrices, since the 4×4 covariance matrix is block-diagonal. This simple solution solved all numerical instability problems. The computational time was only a little bit bigger than with the risk and time models.

Finally, the complete model. In the biggest scenario, with 60 thousand data points and clusters of size 2 i.e., with 30 thousand four-dimension integral approximations (ten parameters in the covariance matrix), the model fitting takes 30 minutes, in parallel, with TMB. Before doing the TMB implementation, to really understand what we were doing, we did a complete `R` implementation. We wrote the marginal log-likelihood in `R`, based on our own Laplace approximation (BONAT; RIBEIRO, 2016) and Newton-Raphson implementation (the gradients, Appendix A, and Hessian, Appendix B, were computed by hand and implemented). Running this complete `R` implementation in a scenario with 20 thousand data points and clusters of size 2, took around 30 hours,

parallelizing it between all threads of system 1. In summary, by using TMB we were able to increase the model size 3 times and to decrease the computational time 60 times. An incredible performance gain.

Still, with the complete model, we performed a Bayesian analysis via `tmbstan` (MONNAHAN; KRISTENSEN, 2018). `tmbstan` enables MCMC sampling (GELFAND; SMITH, 1990; DIACONIS, 2009) from a TMB model object using Stan (Stan Development Team, 2019; Stan Development Team, 2020). Sampling can be performed with or without a Laplace approximation for the random effects. We performed just one Bayesian model fitting in a modest scenario with 5 thousand data points and clusters of size 2. It took around 1 whole week of parallelized processing in system 1. The results were basically the same as the ones obtained with TMB but this high computational time just reinforces the MCMC framework limitation.

An important point to be made here is about TMB's memory consumption. As the sample size increases, the dimension of the model matrices also increases. This, summed to a high number of clusters (Laplace approximations to be performed), turns out to be a computational nightmare. For several models, even the 16GB RAM of system 1 was not enough. The bottleneck appears to be in the AD tape, which is made in parallel, by default, if the model fitting is in parallel. By turning this option off (line 11 of Section D.4 (Appendix D) code), we were able to save a lot of memory, making several models practicable.

Model the CIF of clustered competing risks data is far from being trivial or straightforward. The formulation in Equation 3.1 implies the desired curve behavior, Figure 10. However, in counterpart, its derivatives w.r.t. time, generates very small probabilities for the failure competing causes, ending by concentrating almost everything on censorship, Figure 11. For each competing cause with poor data representativity, we have three curve shape parameters to estimate, implying the necessity of having a lot of data to then have enough information about the causes.

As if the low data representativity is not a big enough problem, it is also complicated to increase the model in terms of competing causes. With two competing causes, we have a 4×4 covariance matrix for the latent effects, which implies ten covariance parameters, which is already a lot. Besides that, by increasing the number of competing causes we also increase the dimension of the integral to be approximated/Laplace approximation, in each cluster. Also, thinking in epidemiological applications, the number of clusters/families tends to be generally big.

We proposed for our multiGLMM an ideally complete latent-effects formulation i.e., correlated latent effects on both levels, instantaneous risk and trajectory time. The main underlying idea of the Chapter 5 simulation study was to see in which scenarios we would be able to learn all the involved mean and covariance parameters. As part of

that, simpler formulations were proposed i.e., latent-effects in only one level, or in both but without cross-correlations. As result, we got that latent effects only in the risk level did not work. The optimization appears to get lost as if something is missing. Inserting latent effects only in the trajectory time sounds to be a lot better, but still. In most of the evaluated scenarios, the block-diagonal model appeared to be in the middle of them, as a compromise. The best results were obtained with the complete model i.e. when we consider the cross-correlations between levels. In general, we still observe some high variances between the parameter estimates, but given all the problem characteristics mentioned earlier, sounds to be reasonable. On average, the complete model works fine, mainly in the scenarios of high CIF configuration, and also as expected, as the sample size increases. We can also say that as the cluster size increases, the estimates get better but we did not have very strong results supporting that.

6.1 FUTURE WORKS

As show in Chapter 5 results, our model does not work smoothly, always presenting a not irrelevant variance. In terms of a traditional GLMM specification (MCCULLOCH; SEARLE, 2001), we do not have a lot more to do. We are already using a smart quasi-Newton algorithm (DENNIS; GAY; WELSCH, 1981), the most efficient derivatives computation technique (AD) (PEYRé, 2020), and an also efficient Laplace approximation routine (WOOD, 2015; BONAT; RIBEIRO, 2016), via TMB (KRISTENSEN et al., 2016). We could change the Laplace approximation for an adaptative Gaussian quadrature (PINHEIRO; CHAO, 2006), but we do not see any good reason to do that.

There are two possible paths here. We could instead of a conditional modeling framework (GLMM/latent-effects model), employ a marginal modeling framework. In this framework, instead of caring about the specification of a probability distribution to the competing causes conditioned on the latent effects, we just care about the specification of a mean and a variance structure. This approach does not have a likelihood function per se, but the estimation procedure tends to be easier than with the GLMM one. A marginal modeling framework that can be used here is the multivariate covariance generalized linear model (McGLM) (BONAT; JØRGENSEN, 2016; BONAT, 2018). How to exactly model the CIF of clustered competing risks data in this framework, is something to still be figured out.

The other path is by the use of a different way of modeling the dependence structure. Instead of a latent-effects approach, we could use copulas (KRUPSKII; JOE, 2013; CHENG; FINE, 2012; SCHEIKE; ZHANG; JENSEN, 2010; EMBRECHTS, 2009). How to do that is something to still be figured out by us, in terms of which kind (conditional or marginal) and version (Archimedean-, Gauss-, Maltesian-, t-, hyperbolic-, zebra-, and elliptical-) of copula to use, besides the estimability issue.

BIBLIOGRAPHY

ANDERSEN, P. K.; GESKUS, R. B.; WITTE, T. de; PUTTER, H. Competing risks in epidemiology: possibilities and pitfalls. *International Journal of Epidemiology*, v. 31, n. 1, p. 861–870, 2012. Cited 2 times on pages 16 and 68.

BATES, D.; MAECHLER, M. *Matrix: Sparse and Dense Matrix Classes and Methods*. R Foundation for Statistical Computing. Vienna, Austria, 2019. R package version 1.2-18 ⟨https://CRAN.R-project.org/package=Matrix⟩. Cited on page 33.

BELISLE, C. J. P. Convergence theorems for a class of simulated annealing algorithms on R^d. *Journal of Applied Probability*, v. 29, n. 1, p. 885–895, 1992. Cited on page 69.

BONAT, W. H. Multiple Response Variables Regression Models in R: The `mcglm` Package. *Journal of Statistical Software*, v. 84, n. 4, 2018. Cited on page 71.

BONAT, W. H.; JØRGENSEN, B. Multivariate covariance generalized linear models. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, v. 65, n. 5, p. 649–675, 2016. Cited on page 71.

BONAT, W. H.; RIBEIRO, P. J. Practical likelihood analysis for spatial generalized linear mixed models. *Environmetrics*, v. 27, n. 1, p. 83–89, 2016. Cited 4 times on pages 22, 24, 69, and 71.

CEDERKVIST, L.; HOLST, K. K.; ANDERSEN, K. K.; SCHEIKE, T. H. Modeling the cumulative incidence function of multivariate competing risks data allowing for within-cluster dependence of risk and timing. *Biostatistics*, v. 20, n. 2, p. 199–217, 2019. Cited 12 times on pages 6, 7, 8, 14, 16, 18, 36, 37, 38, 39, 47, and 68.

CHENG, Y.; FINE, J. P. Cumulative Incidence Association Models for Bivariate Competing Risks Data. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, v. 74, n. 2, p. 183–202, 2012. Cited on page 71.

CLAYTON, D. G. A model for association in bivariate life tables and its application in epidemiological studies of familial rendency in chronic disease incidence. *Biometrika*, v. 65, n. 1, p. 141–151, 1978. Cited on page 16.

COX, D. R.; REID, N. A note on pseudolikelihood constructed from marginal densities. *Biometrika*, v. 91, n. 3, p. 729–737, 2004. Cited 2 times on pages 16 and 68.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood estimation from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B (Methodological)*, v. 39, n. 1, p. 1–38, 1977. Cited on page 22.

DENNIS, J. E.; GAY, D. M.; WELSCH, R. E. An Adaptive Nonlinear Least-Squares Algorithm. *ACM Transactions on Mathematical Software*, v. 7, n. 3, p. 348–368, 1981. Cited 3 times on pages 26, 69, and 71.

DIACONIS, P. The Markov chain Monte Carlo revolution. *Bulletin (New Series) of the American Mathematical Society*, v. 46, n. 2, p. 179–205, 2009. Cited 2 times on pages 21 and 70.

EMBRECHTS, P. Copulas: A Personal View. *The Journal of Risk and Insurance*, v. 76, n. 3, p. 639–650, 2009. Cited on page 71.

FLETCHER, R.; REEVES, C. M. Function minimization by conjugate gradients. *Computer Journal*, v. 7, n. 1, p. 148–154, 1964. Cited on page 69.

FOURNIER, D. A.; SKAUG, H. J.; ANCHETA, J.; IANELLI, J.; MAGNUSSON, A.; MAUNDER, M. N.; NIELSEN, A.; SIBERT, J. AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. *Optimization Methods and Software*, v. 27, n. 2, p. 233–249, 2012. Cited on page 30.

GAY, D. M. *Usage summary for selected optimization routines*. Computing Science Technical Report 153, AT&T Bell Laboratories. Murray Hill, NJ, 1990. Cited 3 times on pages 26, 54, and 69.

GELFAND, A. E.; SMITH, A. F. M. Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, v. 85, n. 410, p. 398–409, 1990. Cited 2 times on pages 21 and 70.

GUENNEBAUD, G.; JACOB, B. et al. *Eigen v3*. 2010. ⟨http://eigen.tuxfamily.org⟩. Cited on page 31.

KALBFLEISCH, J. D.; PRENTICE, R. L. *The Statistical Analysis of Failure Time Data*. Second Edition. Hoboken, New Jersey: John Wiley & Sons, Inc., 2002. Cited 4 times on pages 14, 15, 36, and 68.

KRISTENSEN, K.; NIELSEN, A.; BERG, C. W.; SKAUG, H. J.; BELL, B. M. TMB: Automatic Differentiation and Laplace Approximation. *Journal of Statistical Software*, v. 70, n. 5, p. 1–21, 2016. Cited 9 times on pages 6, 7, 18, 20, 30, 33, 43, 68, and 71.

KRUPSKII, P.; JOE, H. Factor copula models for multivariate data. *Journal of Multivariate Analysis*, v. 120, n. 1, p. 85–101, 2013. Cited on page 71.

LINDSAY, B. G. Composite likelihood methods. *Comtemporary Mathematics*, v. 80, n. 1, p. 221–239, 1988. Cited 2 times on pages 16 and 68.

MCCULLAGH, P.; NELDER, J. A. *Generalized linear models*. Second edition. London: Chapman & Hall, 1989. Cited 2 times on pages 17 and 68.

MCCULLOCH, C. E.; SEARLE, S. R. *Generalized, Linear, and Mixed Models*. New York: John Wiley & Sons, Inc., 2001. Cited 4 times on pages 17, 20, 68, and 71.

MOLENBERGHS, G.; VERBEKE, G. *Models for Discrete Longitudinal Data*. New York: Springer, 2005. Cited on page 22.

MONNAHAN, C.; KRISTENSEN, K. No-U-turn sampling for fast Bayesian inference in ADMB and TMB: Introducing the `adnuts` and `tmbstan` R packages. *PloS ONE*, v. 13, n. 5, 2018. Cited on page 70.

NELDER, J. A.; MEAD, R. A simplex algorithm for function minimization. *Computer Journal*, v. 7, n. 1, p. 308–313, 1965. Cited on page 69.

NELDER, J. A.; WEDDERBURN, R. W. M. Generalized linear models. *Journal of the Royal Statistical Society, Series A*, v. 135, n. 3, p. 370–384, 1972. Cited on page 17.

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. Second Edition. New York: Springer, 2006. (Springer Series in Operations Research and Financial Engineering). Cited 4 times on pages 25, 26, 27, and 69.

PEYRé, G. *Course notes on Optimization for Machine Learning*. 2020. May 10, ⟨https://mathematical-tours.github.io/book-sources/optim-ml/OptimML.pdf⟩. CNRS & DMA, École Normale Supérieure. Cited 3 times on pages 27, 28, and 71.

PINHEIRO, J. C.; BATES, D. M. Unconstrained parametrizations for variance-covariance matrices. *Statistics and Computing*, v. 6, n. 3, p. 289–296, 1996. Cited on page 43.

PINHEIRO, J. C.; CHAO, E. C. Efficient Laplacian and Adaptive Gaussian Quadrature Algorithms for Multilevel Generalized Linear Mixed Models. *Journal of Computational and Graphical Statistics*, v. 15, n. 1, p. 58–81, 2006. Cited 2 times on pages 22 and 71.

POURAHMADI, M. Cholesky decompositions and estimation of a covariance matrix: orthogonality of variance-correlation parameters. *Biometrika*, v. 94, n. 4, p. 1006–1013, 2007. Cited on page 44.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2021. ⟨https://www.R-project.org/⟩. Cited 6 times on pages 6, 7, 20, 26, 30, and 68.

SCHEIKE, T.; ZHANG, Y. S. M.; JENSEN, T. K. A semiparametric random effects model for multivariate competing risks. *Biometrika*, v. 97, n. 1, p. 133–145, 2010. Cited on page 71.

SHUN, Z.; MCCULLAGH, P. Laplace approximation of high dimentional integrals. *Journal of the Royal Statistical Society, Series B (Methodological)*, v. 57, n. 4, p. 749–760, 1995. Cited on page 22.

Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual*. Version 2.26, 2019. ⟨https://mc-stan.org⟩. Cited on page 70.

Stan Development Team. *RStan: the R interface to Stan*. 2020. ⟨https://mc-stan.org/⟩. R package version 2.21.2. Cited on page 70.

TIERNEY, L.; KADANE, J. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, v. 81, n. 393, p. 82–86, 1986. Cited on page 22.

VALPEL, J. W.; MANTON, K. G.; STALLARD, E. The impact of heterogeneity in Individual Frailty on the Dynamics of Mortality. *Demography*, v. 16, n. 1, p. 439–454, 1979. Cited on page 16.

VARIN, C.; REID, N.; FIRTH, D. An overview of composite likelihood methods. *Statistica Sinica*, v. 21, n. 1, p. 5–42, 2011. Cited 2 times on pages 16 and 68.

Ver HOEF, J. M. Who Invented the Delta Method? *The American Statistician*, v. 66, n. 2, p. 124–127, 2012. Cited on page 34.

WOOD, S. N. *Core Statistics*. IMS: Institute of Mathematical Statistics, Textbooks, 2015. Cited 4 times on pages 22, 23, 27, and 71.

Appendix

APPENDIX A – ANALYTIC GRADIENT OF THE LATENT EFFECTS FOR THE JOINT LOG-LIKELIHOOD FUNCTION OF THE MULTINOMIAL GLMM FOR CLUSTERED COMPETING RISKS DATA

The following gradient components are computed by cluster, to be used e.g., in a Newton optimization. Subject i at cluster j and for competing cause k

$$
\begin{aligned}
&\frac{\partial}{\partial u_{kj}} \log L(\theta \mid y_j, r_j) = \\
&y_{kij} \frac{1 + \sum_{m \neq k}^{K-1} \exp\{x_{mij}\beta_{mj} + u_{mj}\}}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} - \left(\sum_{m \neq k}^{K-1} y_{mij} \right) \frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} - \\
&y_{Kij} \frac{1}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} \Bigg(\\
&\frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\} \left(1 + \sum_{m \neq k}^{K-1} \exp\{x_{mij}\beta_{mj} + u_{mj}\}\right)}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} \times \\
&\frac{w_k \frac{\delta}{2\delta t - 2t^2} \phi[w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}]}{1 - w_n \frac{\delta}{2\delta t - 2t^2} \phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}]} - \frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} \times \\
&\frac{\sum_{m \neq k}^{K-1} w_m \frac{\delta}{2\delta t - 2t^2} \phi[w_m \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{mij}\gamma_{mj} - \eta_{mj}] \exp\{x_{mij}\beta_{mj} + u_{mj}\}}{1 - w_n \frac{\delta}{2\delta t - 2t^2} \phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}]} \Bigg) - \\
&e_k^\top Q r_j,
\end{aligned}
$$

$$
\begin{aligned}
&\frac{\partial}{\partial \eta_{kj}} \log L(\theta \mid y_j, r_j) = \\
&y_{kij}(w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}) - \\
&y_{Kij} \frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} \times \\
&\frac{w_k \frac{\delta}{2\delta t - 2t^2} (w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}) \phi[w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}]}{1 - \sum_{n=1}^{K-1} \frac{\exp\{x_{nij}\beta_{nj} + u_{nj}\}}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} w_n \frac{\delta}{2\delta t - 2t^2} \phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}]} - \\
&e_k^\top Q r_j,
\end{aligned}
$$

with $e_k^\top$ begin a vector with 1 at the k-th position and zero elsewhere.

APPENDIX B – ANALYTIC HESSIAN OF THE LATENT EFFECTS FOR THE JOINT LOG-LIKELIHOOD FUNCTION OF THE MULTINOMIAL GLMM FOR CLUSTERED COMPETING RISKS DATA

The following hessian components are computed by cluster, to be used e.g., in a Newton optimization. Subject i at cluster j and for competing cause k

$$
\begin{aligned}
&\frac{\partial^2}{\partial u_{kj}^2}\log L(\theta \mid y_j, r_j) = \\
&-\frac{\left(\sum_{k=1}^{K-1} y_{kij}\right)\exp\{x_{kij}\beta_{kj}+u_{kj}\}\left(1+\sum_{m\neq k}^{K-1}\exp\{x_{mij}\beta_{mj}+u_{mj}\}\right)}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}\right)^2}+ \\
&\frac{y_{Kij}\exp\{x_{kij}\beta_{kj}+u_{kj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}}\times \\
&\frac{\sum_{m\neq k}^{K-1} w_m\frac{\delta}{2\delta t-2t^2}\phi[w_m\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{mij}\gamma_{mj}-\eta_{mj}]\exp\{x_{mij}\beta_{mj}+u_{mj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])}- \\
&\frac{y_{Kij}w_k\frac{\delta}{2\delta t-2t^2}\phi[w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj}]}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}}\times \\
&\frac{\exp\{x_{kij}\beta_{kj}+u_{kj}\}\left(1+\sum_{m\neq k}^{K-1}\exp\{x_{mij}\beta_{mj}+u_{mj}\}\right)}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])}- \\
&\frac{y_{Kij}\exp\{x_{kij}\beta_{kj}+u_{kj}\}}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}\right)^2}\Bigg(\\
&\frac{\sum_{m\neq k}^{K-1} w_m\frac{\delta}{2\delta t-2t^2}\phi[w_m\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{mij}\gamma_{mj}-\eta_{mj}]\exp\{x_{mij}\beta_{mj}+u_{mj}\}}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])\right)^2}- \\
&\frac{w_k\frac{\delta}{2\delta t-2t^2}\phi[w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj}]\left(1+\sum_{m\neq k}^{K-1}\exp\{x_{mij}\beta_{mj}+u_{mj}\}\right)}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])\right)^2}\Bigg) \\
&\times\Bigg(\Bigg(1+ \\
&\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])\Bigg)+ \\
&\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}\right)\times \\
&(1-w_k\frac{\delta}{2\delta t-2t^2}\phi[w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj}])\Bigg)-e_k^\top Q,
\end{aligned}
$$

$$
\begin{aligned}
&\frac{\partial^2}{\partial \eta_{kj}^2}\log L(\theta \mid y_j, r_j) = \\
&-y_{kij}-y_{Kij}\frac{\exp\{x_{kij}\beta_{kj}+u_{kj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}}\Bigg(\\
&w_k\frac{\delta}{2\delta t-2t^2}\phi[w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj}]\times \\
&\frac{\left(w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj}\right)^2-1}{1-\sum_{n=1}^{K-1}\frac{\exp\{x_{nij}\beta_{nj}+u_{nj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}}w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}]}- \\
&\frac{\left(w_k\frac{\delta}{2\delta t-2t^2}(w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj})\phi[w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj}]\right)^2}{\left(1-\sum_{n=1}^{K-1}\frac{\exp\{x_{nij}\beta_{nj}+u_{nj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}}w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}]\right)^2} \\
&\Bigg)-e_k^\top Q,
\end{aligned}
$$

$$
\begin{aligned}
&\frac{\partial^2}{\partial u_{kj}u_{mj}}\log L(\theta \mid y_j, r_j) = \\
&\left(\sum_{k=1}^{K-1} y_{kij}\right)\frac{\exp\{x_{kij}\beta_{kj}+u_{kj}\}\exp\{x_{mij}\beta_{mj}+u_{mj}\}}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}\right)^2}+ \\
&\frac{y_{Kij}\exp\{x_{kij}\beta_{kj}+u_{kj}\}\exp\{x_{mij}\beta_{mj}+u_{mj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}}\Bigg(\\
&\frac{w_m\frac{\delta}{2\delta t-2t^2}\phi[w_m\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{mij}\gamma_{mj}-\eta_{mj}]}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])}- \\
&\frac{w_k\frac{\delta}{2\delta t-2t^2}\phi[w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj}]}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])}\Bigg)- \\
&\frac{y_{Kij}}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}\right)^2}\Bigg(\exp\{x_{kij}\beta_{kj}+u_{kj}\}\Bigg(\\
&\frac{\sum_{m\neq k}^{K-1} w_m\frac{\delta}{2\delta t-2t^2}\phi[w_m\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{mij}\gamma_{mj}-\eta_{mj}]\exp\{x_{mij}\beta_{mj}+u_{mj}\}}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])\right)^2}- \\
&\frac{w_k\frac{\delta}{2\delta t-2t^2}\phi[w_k\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{kij}\gamma_{kj}-\eta_{kj}]\left(1+\sum_{m\neq k}^{K-1}\exp\{x_{mij}\beta_{mj}+u_{mj}\}\right)}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}(1-w_n\frac{\delta}{2\delta t-2t^2}\phi[w_n\text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right)-x_{nij}\gamma_{nj}-\eta_{nj}])\right)^2}\Bigg)
\end{aligned}
$$

$$\Bigg) \times \Bigg(\exp\{x_{mij}\beta_{mj} + u_{mj}\}\Big(1+$$
$$\sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}(1 - w_n \frac{\delta}{2\delta t - 2t^2}\phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}])\Big) +$$
$$\exp\{x_{mij}\beta_{mj} + u_{mj}\}(1 - w_m \frac{\delta}{2\delta t - 2t^2}\phi[w_m \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{mij}\gamma_{mj} - \eta_{mj}])\Big(1+$$
$$\sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}\Big)\Bigg) - e_k^\top Q,$$

$$\frac{\partial^2}{\partial \eta_{kj}\eta_{mj}} \log L(\theta \mid y_j, r_j) =$$
$$- y_{Kij} \frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} \times$$
$$\frac{w_k \frac{\delta}{2\delta t - 2t^2}(w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj})\phi[w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}]}{\left(1 - \sum_{n=1}^{K-1} \frac{\exp\{x_{nij}\beta_{nj}+u_{nj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}} w_n \frac{\delta}{2\delta t - 2t^2}\phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}]\right)^2} \times$$
$$\frac{\exp\{x_{mij}\beta_{mj} + u_{mj}\}}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} w_m \frac{\delta}{2\delta t - 2t^2}(w_m \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{mij}\gamma_{mj} - \eta_{mj}) \times$$
$$\phi[w_m \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{mij}\gamma_{mj} - \eta_{mj}] - e_k^\top Q,$$

$$\frac{\partial^2}{\partial \eta_{kj}u_{kj}} \log L(\theta \mid y_j, r_j) =$$
$$y_{Kij} \frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}}{1 + \sum_{n=1}^{K-1} \exp\{x_{nij}\beta_{nj} + u_{nj}\}} \times$$
$$\frac{w_k \frac{\delta}{2\delta t - 2t^2}(w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj})\phi[w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj})]}{\left(1 - \sum_{n=1}^{K-1} \frac{\exp\{x_{nij}\beta_{nj}+u_{nj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}} w_n \frac{\delta}{2\delta t - 2t^2}\phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}]\right)^2} \times$$
$$\left(\sum_{n\neq k}^{K-1} \frac{\exp\{x_{nij}\beta_{nj} + u_{nj}\}\exp\{x_{kij}\beta_{kj} + u_{kj}\}}{\left(1 + \sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj} + u_{nj}\}\right)^2} \times \right.$$
$$w_n \frac{\delta}{2\delta t - 2t^2}\phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}] -$$
$$\frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}\left(\left(1 + \sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj} + u_{nj}\}\right) - \exp\{x_{kij}\beta_{kj} + u_{kj}\}\right)}{\left(1 + \sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj} + u_{nj}\}\right)^2} \times$$
$$\left. w_k \frac{\delta}{2\delta t - 2t^2}\phi[w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}]\right) -$$

$$y_{Kij} \frac{\frac{\exp\{x_{kij}\beta_{kj}+u_{kj}\}\left(\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}\right)-\exp\{x_{kij}\beta_{kj}+u_{kj}\}\right)}{\left(1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}\right)^2}}{1 - \sum_{n=1}^{K-1} \frac{\exp\{x_{nij}\beta_{nj}+u_{nj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}} w_n \frac{\delta}{2\delta t - 2t^2}\phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}]} \times$$
$$w_k \frac{\delta}{2\delta t - 2t^2}(w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}) \times$$
$$\phi[w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj}] - e_k^\top Q,$$

$$\frac{\partial^2}{\partial \eta_{kj}u_{mj}} \log L(\theta \mid y_j, r_j) =$$
$$y_{Kij} \frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}\exp\{x_{mij}\beta_{mj} + u_{mj}\}}{\left(1 + \sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj} + u_{nj}\}\right)^2} \times$$
$$\frac{w_k \frac{\delta}{2\delta t - 2t^2}(w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj})\phi[w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj})]}{1 - \sum_{n=1}^{K-1} \frac{\exp\{x_{nij}\beta_{nj}+u_{nj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}} w_n \frac{\delta}{2\delta t - 2t^2}\phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}]} +$$
$$y_{Kij} \frac{\exp\{x_{kij}\beta_{kj} + u_{kj}\}}{1 + \sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj} + u_{nj}\}} \times$$
$$\frac{w_k \frac{\delta}{2\delta t - 2t^2}(w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj})\phi[w_k \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{kij}\gamma_{kj} - \eta_{kj})]}{\left(1 - \sum_{n=1}^{K-1} \frac{\exp\{x_{nij}\beta_{nj}+u_{nj}\}}{1+\sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj}+u_{nj}\}} w_n \frac{\delta}{2\delta t - 2t^2}\phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}]\right)^2} \times$$
$$\left(\sum_{n\neq m}^{K-1} \frac{\exp\{x_{nij}\beta_{nj} + u_{nj}\}\exp\{x_{mij}\beta_{mj} + u_{mj}\}}{\left(1 + \sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj} + u_{nj}\}\right)^2} \times \right.$$
$$w_n \frac{\delta}{2\delta t - 2t^2}\phi[w_n \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{nij}\gamma_{nj} - \eta_{nj}] -$$
$$\frac{\exp\{x_{mij}\beta_{mj} + u_{mj}\}\left(\left(1 + \sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj} + u_{nj}\}\right) - \exp\{x_{mij}\beta_{mj} + u_{mj}\}\right)}{\left(1 + \sum_{n=1}^{K-1}\exp\{x_{nij}\beta_{nj} + u_{nj}\}\right)^2} \times$$
$$\left. w_m \frac{\delta}{2\delta t - 2t^2}\phi[w_m \text{arctanh}\left(\frac{t-\delta/2}{\delta/2}\right) - x_{mij}\gamma_{mj} - \eta_{mj}]\right) - e_k^\top Q,$$

with $e_k^\top$ begin a vector with 1 at the k-th position and zero elsewhere.

APPENDIX C – R CODE TO SIMULATE FROM A multiGLMM WITH TWO COMPETING CAUSES AND CLUSTERS OF SIZE TWO. FOR MORE INFORMATION CHECK SECTION 4.1

```
library(mvtnorm)   ## install.packages('mvtnorm')
library(tidyverse) ## install.packages('tidyverse')
library(mc2d)      ## install.packages('mc2d')

datasimu <- function(J,    ## number of clusters
                     cs,   ## clusters size (all the same size)
                     time, ## failure, censorship times
                     Z,    ## latent effects design-matrix
                     S,    ## variance-covariance matrix
                     delta=80,
                     beta =c( beta1=-2.0,  beta2=-1.5),
                     gamma=c(gamma1= 1.2, gamma2= 1.0),
                     w    =c(    w1= 3.0,     w2= 5.0),
                     seed1=NULL,
                     seed2=NULL)
{
    out <- tibble::tibble(i=rep(seq(cs), times=J), ## cluster element
                          j=rep(seq(J),  each=cs), ## cluster
                          time=time,
                          p1=NA,
                          p2=NA,
                          p3=NA)

    K     <- dim(S)[1]/2 + 1
    ladim <- 2*(K-1) ## latent effects dimension
    set.seed(seed1)
    U     <- mvtnorm::rmvnorm(J, mean=rep(0, ladim), sigma=S)
    ZU    <- Z%*%U
    risk1 <- exp(beta['beta1'] + ZU[, 1])
    risk2 <- exp(beta['beta2'] + ZU[, 2])
    level <- 1 + risk1 + risk2
    gt    <- atanh(2*time/delta - 1)
    dgt   <- delta/(2*time*(delta - time))
    x1    <- w['w1']*gt - gamma['gamma1'] - ZU[, 3]
    x2    <- w['w2']*gt - gamma['gamma2'] - ZU[, 4]

    out$p1 <- risk1/level*w['w1']*dgt*dnorm(x1)
    out$p2 <- risk2/level*w['w2']*dgt*dnorm(x2)

    out    <- out %>% dplyr::mutate(p3=1-p1-p2)
    set.seed(seed2)
    y      <- mc2d::rmultinomial(cs*J, 1, prob=out%>%select(p1:p3))

    out    <- out %>%
```



```
        dplyr::bind_cols(tibble::as_tibble(y)) %>%
        dplyr::rename(y1=V1, y2=V2, y3=V3)
    return(out)
}
J    <- 50e3
cs   <- 2
time <- runif(n=cs*J, min=30, max=79.9)
Z    <- Matrix::bdiag(replicate(J, rep(1, cs), simplify=FALSE))
S    <- matrix(c( 1.0,  0.4, -0.1,  0.4,
                  0.4,  1.0,  0.4, -0.1,
                 -0.1,  0.4,  1.0,  0.4,
                  0.4, -0.1,  0.4,  1.0), 4)
dat  <- datasimu(J=J, cs=cs, time=time, Z=Z, S=S, seed1=1, seed2=2)
```

SOURCE: The author (2021).

APPENDIX D – C++ CODES FOR THE TMB IMPLEMENTATION OF THE multiGLMM COMPLETE MODEL'S SPECIAL CASES

D.1 C++ CODE FOR THE TMB IMPLEMENTATION OF A multiGLMM WITH A 2×2 LATENT STRUCTURE ON THE RISK LEVEL

```
// multiGLMM: A MULTINOMIAL GLMM FOR CLUSTERED COMPETING RISKS DATA
// 2x2 LATENT STRUCTURE ON THE RISK LEVEL (RISK MODEL)
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  using namespace density;
  DATA_MATRIX(Y);
  DATA_SPARSE_MATRIX(Z);
  DATA_VECTOR(time);
  DATA_SCALAR(delta);
  PARAMETER(beta1);
  PARAMETER(beta2);
  PARAMETER(gama1);
  PARAMETER(gama2);
  PARAMETER(w1);
  PARAMETER(w2);
  PARAMETER(logs2_1); Type s2_1=exp(logs2_1);
  PARAMETER(logs2_2); Type s2_2=exp(logs2_2);
  PARAMETER(rhoZ12); Type rho12=(exp(2*rhoZ12)-1)/(exp(2*rhoZ12)+1);

  PARAMETER_MATRIX(U); matrix<Type> ZU=Z*U;
  Type risk1=0;
  Type risk2=0;
  Type level=0;
  // gt=atanh(2*time/delta-1); atanh(x)=0.5*log((1+x)/(1-x))
  vector<Type> gt=0.5*log(time/(delta-time));
  vector<Type> dgt=delta/(2*time*(delta-time));
  Type x1=0;
  Type x2=0;
  vector<Type> y(Y.cols());
  vector<Type> prob(Y.cols());
  parallel_accumulator<Type> nll(this);
  // Type nll=0;
  vector<Type> u(U.cols());

  Type cov12=rho12*sqrt(s2_1)*sqrt(s2_2);
  matrix<Type> Sigma(2, 2);
  Sigma.row(0) << s2_1, cov12;
  Sigma.row(1) << cov12, s2_2;
```



```

  MVNORM_t<Type> dmvnorm(Sigma);
  for (int i=0; i<U.rows(); i++) {
    u=U.row(i);
    nll += dmvnorm(u);
  }
  for (int i=0; i<Y.rows(); i++) {
    risk1=exp(beta1 + ZU(i, 0));
    risk2=exp(beta2 + ZU(i, 1));
    level=1 + risk1 + risk2;
    x1=w1*gt(i) - gama1;
    x2=w2*gt(i) - gama2;
    prob(0)=risk1/level * w1*dgt(i) * dnorm(x1, Type(0), Type(1), false);
    prob(1)=risk2/level * w2*dgt(i) * dnorm(x2, Type(0), Type(1), false);
    prob(2)=1 - prob(0) - prob(1);
    y=Y.row(i);
    nll -= dmultinom(y, prob, true);
  }
  ADREPORT(s2_1);
  ADREPORT(s2_2);
  ADREPORT(rho12);
  REPORT(Sigma);
  return nll;
}
```

SOURCE: The author (2021).

D.2 C++ CODE FOR THE TMB IMPLEMENTATION OF A multiGLMM WITH A 2×2 LATENT STRUCTURE ON THE TRAJECTORY TIME LEVEL

```
// multiGLMM: A MULTINOMIAL GLMM FOR CLUSTERED COMPETING RISKS DATA
// 2x2 LATENT STRUCTURE ON THE TRAJECTORY TIME LEVEL (TIME MODEL)
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  using namespace density;
  DATA_MATRIX(Y);
  DATA_SPARSE_MATRIX(Z);
  DATA_VECTOR(time);
  DATA_SCALAR(delta);
  PARAMETER(beta1);
  PARAMETER(beta2);
  PARAMETER(gama1);
  PARAMETER(gama2);
  PARAMETER(w1);
```

```
  PARAMETER(w2);
  PARAMETER(logs2_3); Type s2_3=exp(logs2_3);
  PARAMETER(logs2_4); Type s2_4=exp(logs2_4);
  PARAMETER(rhoZ34); Type rho34=(exp(2*rhoZ34)-1)/(exp(2*rhoZ34)+1);

  PARAMETER_MATRIX(U); matrix<Type> ZU=Z*U;
  Type risk1=exp(beta1);
  Type risk2=exp(beta2);
  Type level=1 + risk1 + risk2;
  // gt=atanh(2*time/delta-1); atanh(x)=0.5*log((1+x)/(1-x))
  vector<Type> gt=0.5*log(time/(delta-time));
  vector<Type> dgt=delta/(2*time*(delta-time));
  Type x1=0;
  Type x2=0;
  vector<Type> y(Y.cols());
  vector<Type> prob(Y.cols());
  parallel_accumulator<Type> nll(this);
  // Type nll=0;
  vector<Type> u(U.cols());

  Type cov34=rho34*sqrt(s2_3)*sqrt(s2_4);
  matrix<Type> Sigma(2, 2);
  Sigma.row(0) << s2_3, cov34;
  Sigma.row(1) << cov34, s2_4;

  MVNORM_t<Type> dmvnorm(Sigma);
  for (int i=0; i<U.rows(); i++) {
    u=U.row(i);
    nll += dmvnorm(u);
  }
  for (int i=0; i<Y.rows(); i++) {
    x1=w1*gt(i) - gama1 - ZU(i, 0);
    x2=w2*gt(i) - gama2 - ZU(i, 1);
    prob(0)=risk1/level * w1*dgt(i) * dnorm(x1, Type(0), Type(1), false);
    prob(1)=risk2/level * w2*dgt(i) * dnorm(x2, Type(0), Type(1), false);
    prob(2)=1 - prob(0) - prob(1);
    y=Y.row(i);
    nll -= dmultinom(y, prob, true);
  }
  ADREPORT(s2_3);
  ADREPORT(s2_4);
  ADREPORT(rho34);
  REPORT(Sigma);
  return nll;
}
```

SOURCE: The author (2021).

D.3 C++ CODE FOR THE TMB IMPLEMENTATION OF A multiGLMM WITH A BLOCK-DIAG 4×4 LATENT STRUCTURE

```
// multiGLMM: A MULTINOMIAL GLMM FOR CLUSTERED COMPETING RISKS DATA
// BLOCK-DIAG LATENT STRUCTURE i.e., RISK, TRAJECTORY TIME, AND
// CROSS-CORRELATIONS SET AT ZERO (BLOCK-DIAG MODEL)
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  using namespace density;
  DATA_MATRIX(Y);
  DATA_SPARSE_MATRIX(Z);
  DATA_VECTOR(time);
  DATA_SCALAR(delta);
  PARAMETER(beta1);
  PARAMETER(beta2);
  PARAMETER(gama1);
  PARAMETER(gama2);
  PARAMETER(w1);
  PARAMETER(w2);

  PARAMETER(logs2_1); Type s2_1=exp(logs2_1);
  PARAMETER(logs2_2); Type s2_2=exp(logs2_2);
  PARAMETER(logs2_3); Type s2_3=exp(logs2_3);
  PARAMETER(logs2_4); Type s2_4=exp(logs2_4);

  PARAMETER(rhoZ12); Type rho12=(exp(2*rhoZ12)-1)/(exp(2*rhoZ12)+1);
  PARAMETER(rhoZ34); Type rho34=(exp(2*rhoZ34)-1)/(exp(2*rhoZ34)+1);

  PARAMETER_MATRIX(U1); matrix<Type> ZU1=Z*U1;
  PARAMETER_MATRIX(U2); matrix<Type> ZU2=Z*U2;
  Type risk1=0;
  Type risk2=0;
  Type level=0;
  // gt=atanh(2*time/delta-1); atanh(x)=0.5*log((1+x)/(1-x))
  vector<Type> gt=0.5*log(time/(delta-time));
  vector<Type> dgt=delta/(2*time*(delta-time));
  Type x1=0;
  Type x2=0;
  vector<Type> y(Y.cols());
  vector<Type> prob(Y.cols());
  parallel_accumulator<Type> nll(this);
  // Type nll=0;
  vector<Type> u1(U1.cols());
  vector<Type> u2(U2.cols());

```

```
  Type cov12=rho12*sqrt(s2_1)*sqrt(s2_2);
  Type cov34=rho34*sqrt(s2_3)*sqrt(s2_4);
  matrix<Type> Sigma1(2, 2);
  Sigma1.row(0) << s2_1, cov12;
  Sigma1.row(1) << cov12, s2_2;
  matrix<Type> Sigma2(2, 2);
  Sigma2.row(0) << s2_3, cov34;
  Sigma2.row(1) << cov34, s2_4;

  MVNORM_t<Type> dmvnorm1(Sigma1);
  MVNORM_t<Type> dmvnorm2(Sigma2);
  for (int i=0; i<U1.rows(); i++) {
    u1=U1.row(i);
    nll += dmvnorm1(u1);
  }
  for (int i=0; i<U2.rows(); i++) {
    u2=U2.row(i);
    nll += dmvnorm2(u2);
  }
  for (int i=0; i<Y.rows(); i++) {
    risk1=exp(beta1 + ZU1(i, 0));
    risk2=exp(beta2 + ZU1(i, 1));
    level=1 + risk1 + risk2;
    x1=w1*gt(i) - gama1 - ZU2(i, 0);
    x2=w2*gt(i) - gama2 - ZU2(i, 1);
    prob(0)=risk1/level * w1*dgt(i) * dnorm(x1, Type(0), Type(1), false);
    prob(1)=risk2/level * w2*dgt(i) * dnorm(x2, Type(0), Type(1), false);
    prob(2)=1 - prob(0) - prob(1);
    y=Y.row(i);
    nll -= dmultinom(y, prob, true);
  }
  ADREPORT(s2_1);
  ADREPORT(s2_2);
  ADREPORT(s2_3);
  ADREPORT(s2_4);
  ADREPORT(rho12);
  ADREPORT(rho34);
  REPORT(Sigma1);
  REPORT(Sigma2);
  return nll;
}
```

SOURCE: The author (2021).

D.4 R CODE SHOWING HOW TO LOAD AND FIT THE multiGLMM VERSIONS

```
## choose the desired MODEL to fit (risk, time, block-diag, complete)
```



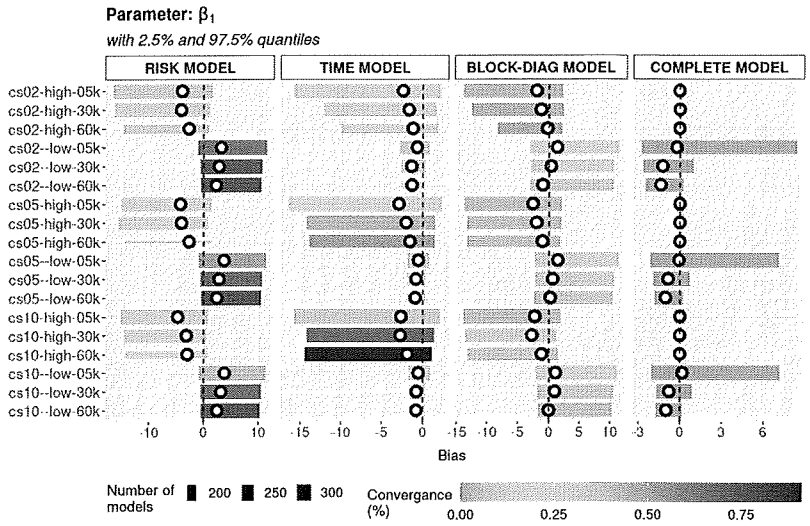
```
dll <- 'MODEL'

library(TMB)      ## install.packages('TMB')
library(parallel) ## install.packages('parallel')
library(Matrix)   ## install.packages('Matrix')

filename <- paste0(dll, '.cpp')
TMB::compile(filename)
dyn.load(TMB::dynlib(dll))
TMB::config(tape.parallel=FALSE, DLL=dll) ## saves a lot of memory usage
## if you want to make a multi-thread model fitting
TMB::openmp(parallel::detectCores())

J      <- 500 ## choose the number of clusters
cs     <- 2 ## choose the cluster sizes
time   <- runif(n=cs*J, min=30, max=79.9) ## generate the failure times
delta  <- 80
blocks <- replicate(J, rep(1, cs), simplify=FALSE)
Z      <- Matrix::bdiag(blocks) ## build the latent-effect design-matrix

## set the fixed-effect parameters
beta  <- c(beta1=-2, beta2=-1.5)
gamma <- c(gamma1=1.2, gamma2=1)
w     <- c(w1=3, w2=5)
## set the variances and correlations
s2_1  <- 1.0
s2_2  <- 0.6
s2_3  <- 0.7
s2_4  <- 0.9
rho12 <-  0.1
rho13 <- -0.5
rho14 <-  0.3
rho23 <-  0.3
rho24 <- -0.4
rho34 <-  0.2
## auxiliary function to build and check if the Sigma is
## positive-definite (PD)
buildSigma <- function(s2_1, s2_2, s2_3, s2_4,
                       rho12, rho13, rho14, rho23, rho24, rho34)
{
    cov12 <- rho12*sqrt(s2_1)*sqrt(s2_2)
    cov13 <- rho13*sqrt(s2_1)*sqrt(s2_3)
    cov14 <- rho14*sqrt(s2_1)*sqrt(s2_4)
    cov23 <- rho23*sqrt(s2_2)*sqrt(s2_3)
    cov24 <- rho24*sqrt(s2_2)*sqrt(s2_4)
    cov34 <- rho34*sqrt(s2_3)*sqrt(s2_4)
    Sigma <- matrix(c(s2_1, cov12, cov13, cov14,
```

```
                    cov12, s2_2, cov23, cov24,
                    cov13, cov23, s2_3, cov34,
                    cov14, cov24, cov34, s2_4), nrow=4)
    ## Sigma will only be returned if PD
    if (
        is.matrix(chol(Sigma))
    )
        return(Sigma)
}
Sigma <- buildSigma(s2_1, s2_2, s2_3, s2_4,
                    rho12, rho13, rho14, rho23, rho24, rho34)

## generate data via the function datasimu() from APPENDIX C. to make it
## simpler, you may save the function in a file and then load in the
## current section
source('datasimu.R')
dat <- datasimu(J=J, cs=cs, time=time,
                Z=Z, S=Sigma, delta=delta,
                beta=beta, gamma=gamma, w=w, seed1=1, seed2=2)
y <- as.matrix( dat%>%dplyr::select(y1:y3) )

## latent-effects matrix U filled with zeros (initial guesses)
## ncol has to be 2 or 4, depending of the chosen MODEL
U <- matrix(0, nrow=J, ncol=4)
## the model fit per se starts now
obj <- TMB::MakeADFun(data=list(Y=y, Z=Z, time=time, delta=delta),
                      parameters=list(beta1  =beta['beta1'],
                                      beta2  =beta['beta2'],
                                      gama1  =gamma['gamma1'],
                                      gama2  =gamma['gamma2'],
                                      w1     =w['w1'],
                                      w2     =w['w2'],
                                      logs2_1=log(s2_1),
                                      logs2_2=log(s2_2),
                                      logs2_3=log(s2_3),
                                      logs2_4=log(s2_4),
                                      rhoZ12 =atanh(rho12),
                                      rhoZ13 =atanh(rho13),
                                      rhoZ14 =atanh(rho14),
                                      rhoZ23 =atanh(rho23),
                                      rhoZ24 =atanh(rho24),
                                      rhoZ34 =atanh(rho34),
                                      U      =U),
                      DLL=dll, random='U', hessian=TRUE, silent=TRUE)
opt <- with(obj, nlminb(par, fn, gr))
```

SOURCE: The author (2021).

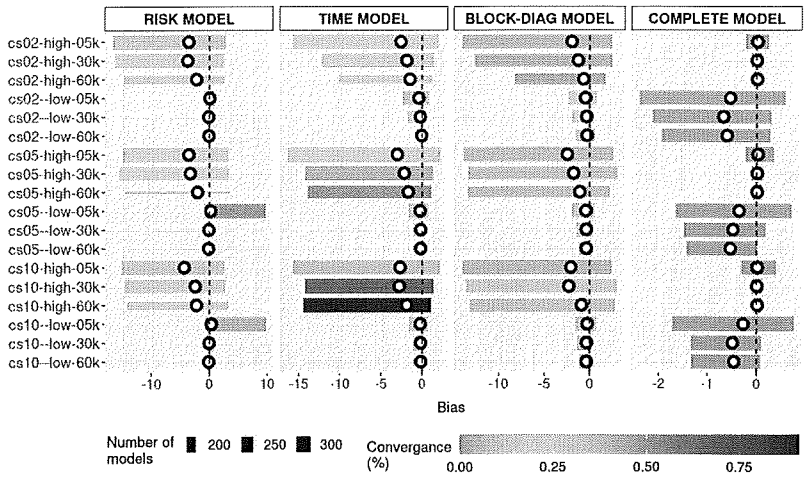
APPENDIX E – MODEL PARAMETERS BIAS WITH 2.5% AND 97.5% QUANTILES

FIGURE 32 – PARAMETER β_1 BIAS WITH 2.5% AND 97.5% QUANTILES



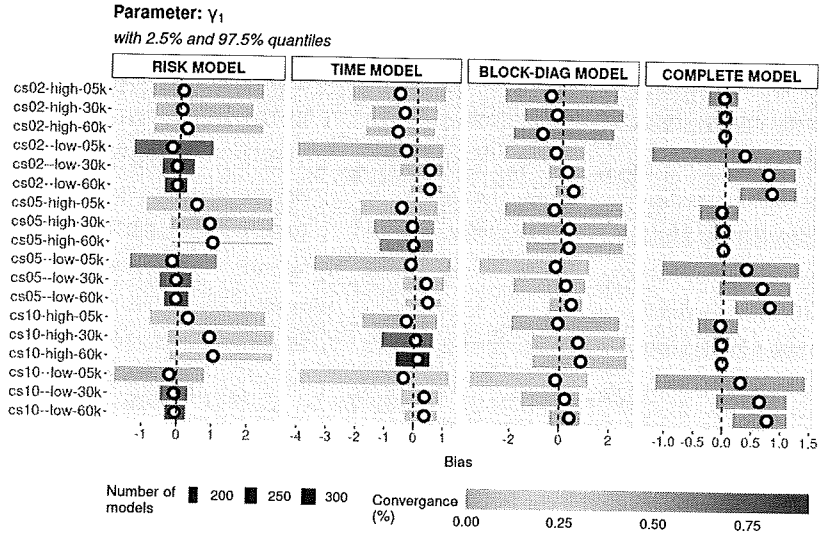
SOURCE: The author (2021).

FIGURE 33 – PARAMETER β_2 BIAS WITH 2.5% AND 97.5% QUANTILES



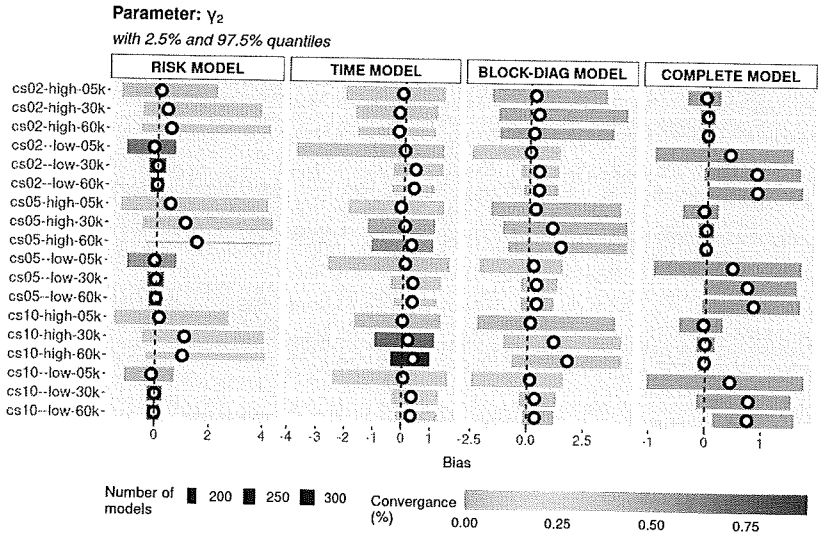
SOURCE: The author (2021).

FIGURE 34 – PARAMETER γ_1 BIAS WITH 2.5% AND 97.5% QUANTILES



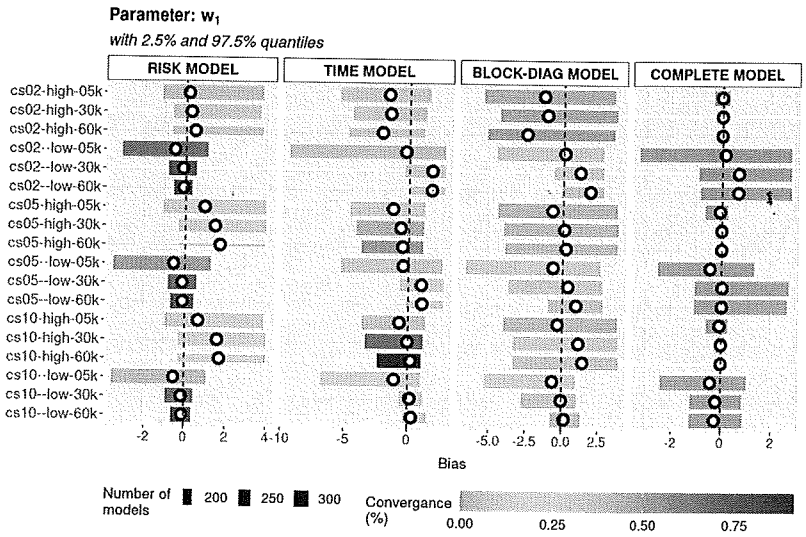
SOURCE: The author (2021).

FIGURE 35 – PARAMETER γ_2 BIAS WITH 2.5% AND 97.5% QUANTILES



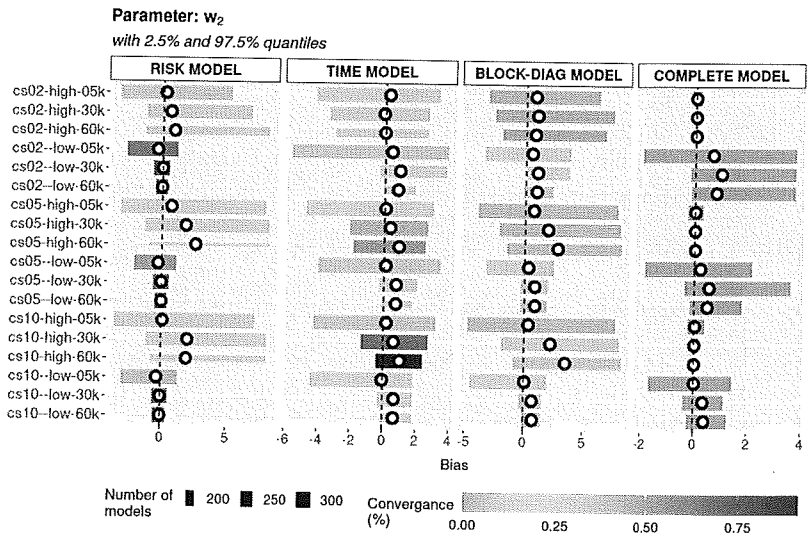
SOURCE: The author (2021).

FIGURE 36 – PARAMETER w_1 BIAS WITH 2.5% AND 97.5% QUANTILES



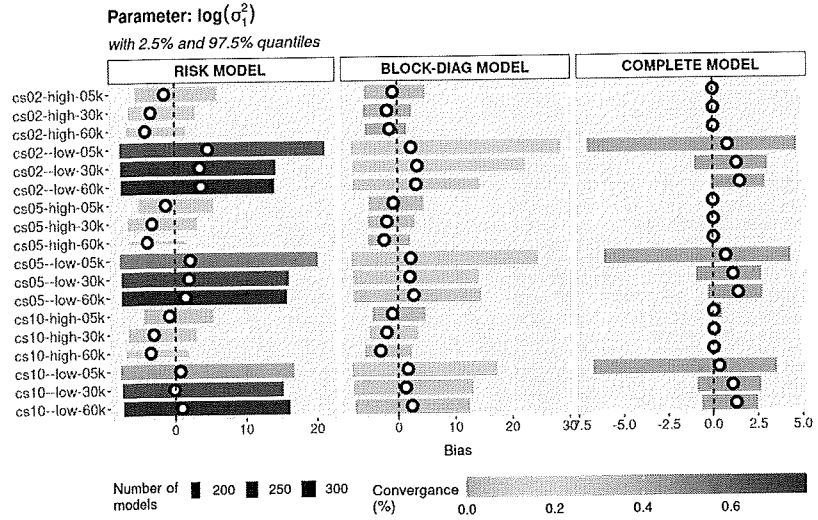
SOURCE: The author (2021).

FIGURE 37 – PARAMETER w_2 BIAS WITH 2.5% AND 97.5% QUANTILES



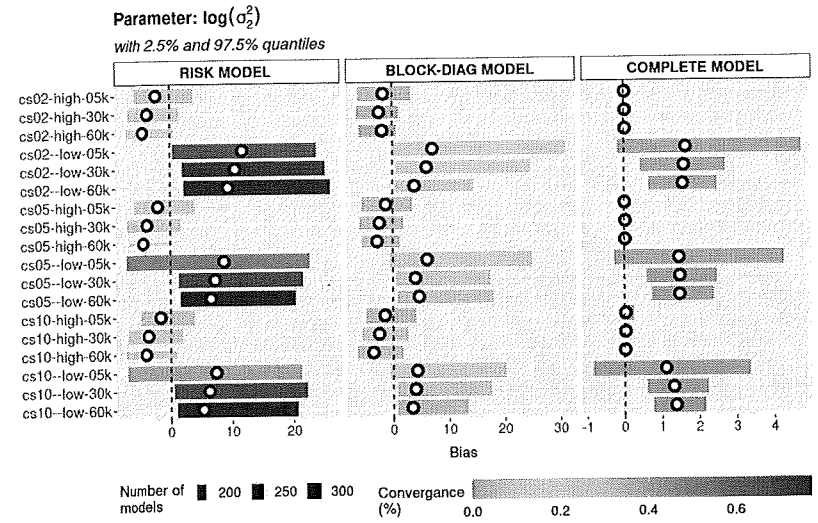
SOURCE: The author (2021).

FIGURE 38 – PARAMETER $\log(\sigma_1^2)$ BIAS WITH 2.5% AND 97.5% QUANTILES



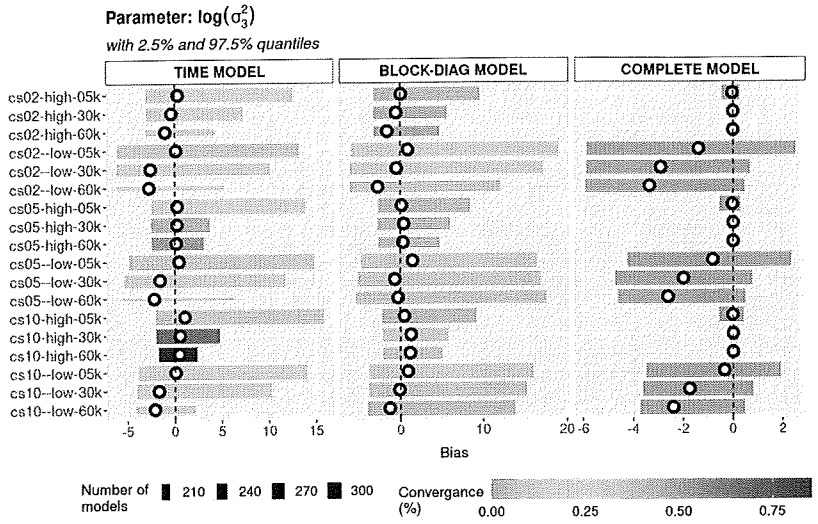
SOURCE: The author (2021).

FIGURE 39 – PARAMETER $\log(\sigma_2^2)$ BIAS WITH 2.5% AND 97.5% QUANTILES



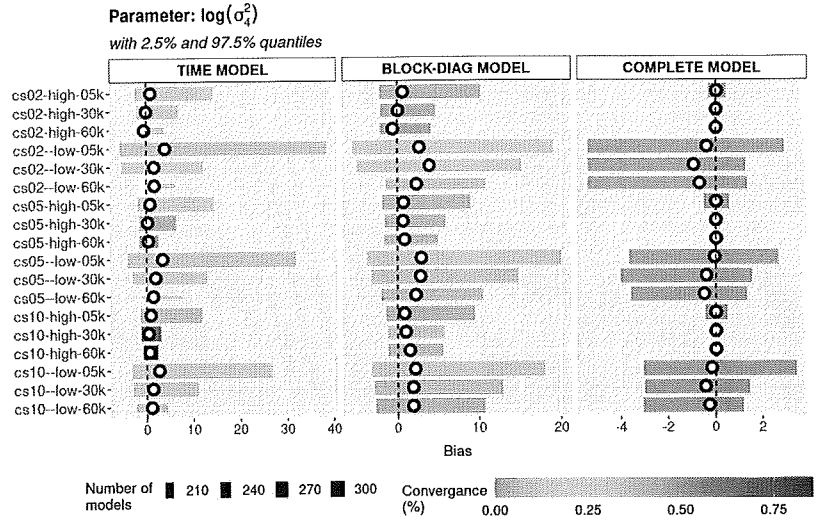
SOURCE: The author (2021).

FIGURE 40 – PARAMETER $\log(\sigma_3^2)$ BIAS WITH 2.5% AND 97.5% QUANTILES



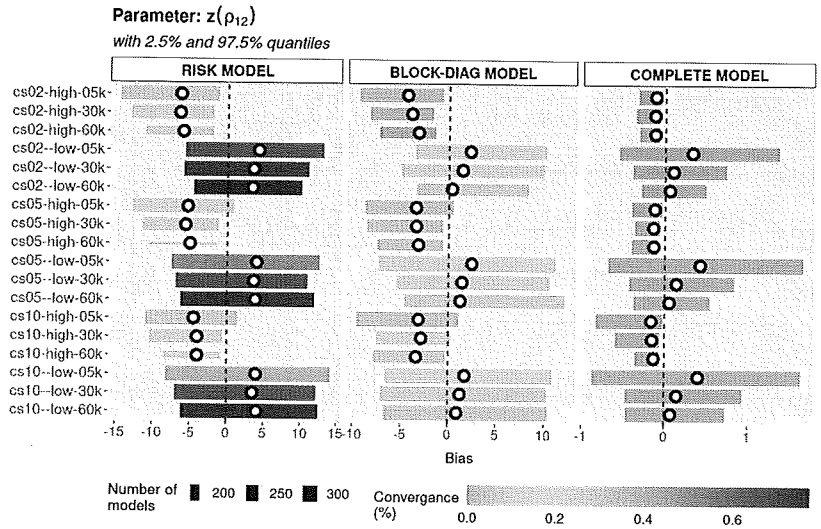
SOURCE: The author (2021).

FIGURE 41 – PARAMETER $\log(\sigma_4^2)$ BIAS WITH 2.5% AND 97.5% QUANTILES



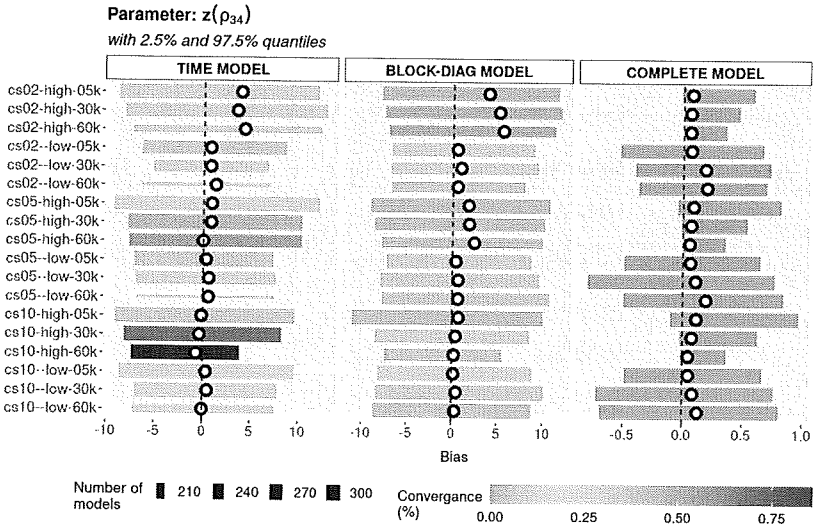
SOURCE: The author (2021).

FIGURE 42 – PARAMETER $z(\rho_{12})$ BIAS WITH 2.5% AND 97.5% QUANTILES



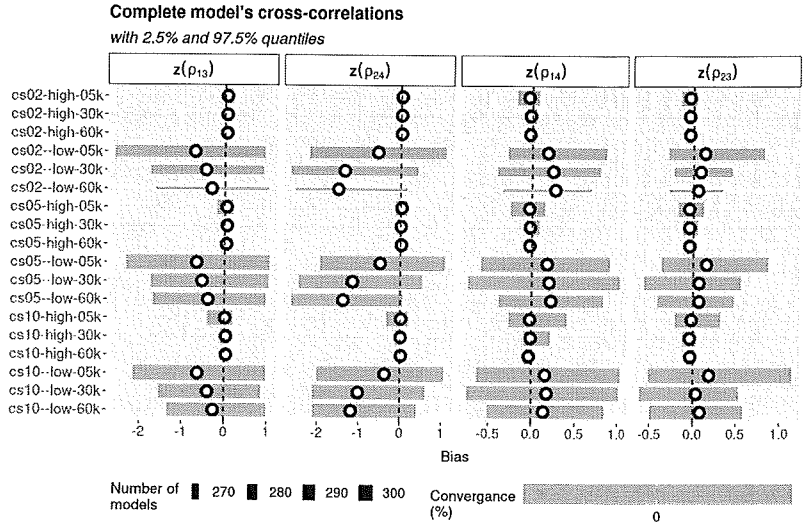
SOURCE: The author (2021).

FIGURE 43 – PARAMETER $z(\rho_{34})$ BIAS WITH 2.5% AND 97.5% QUANTILES



SOURCE: The author (2021).

FIGURE 44 – PARAMETERS $\{z(\rho_{13}),\ z(\rho_{24}),\ z(\rho_{14}),\ z(\rho_{23})\}$ BIAS WITH 2.5% AND 97.5% QUANTILES



SOURCE: The author (2021).