Extending the code in the open-source saemix package to fit joint models of longitudinal and time-to-event data

Alexandra Lavalley-morelle, France Mentré, Emmanuelle Comets, Jimmy Mullaert

Highlights

- Extension of the code of saemix package initially designed for nonlinear mixed-effects models to the case of joint models.
- Integration of a recently developed stochastic algorithm having interesting properties for standard error estimation.
- Good performances for parameter and standard error estimations.
- New flexible tool allowing users to fit very specific joint models by defining a personalized likelihood function.

ScienceDirect
www.sciencedirect.com

Computer Methods and Programms in Biomedicine 00 (2024) 1–63

Extending the code in the open-source *saemix* package to fit joint models of longitudinal and time-to-event data

Alexandra Lavalley-Morelle[a,*], France Mentré[a,b], Emmanuelle Comets[a,c,1], Jimmy Mullaert[a,b,d,1]

[a] *Université Paris Cité, INSERM, IAME, F-75018 Paris, France*
[b] *Department of Epidemiology, Biostatistics and Clinical Research, AP-HP, Bichat-Claude Bernard University Hospital, F-75018 Paris*
[c] *Université de Rennes, Inserm, EHESP, Irset - UMRS 1085, F-35000 Rennes, France*
[d] *Université Paris-Saclay, UVSQ, Institut Curie, Cancer et Génome, 92210 Saint-Cloud, France*

Abstract

Background and Objective: Joint modeling of longitudinal and time-to-event data has gained attention over recent years with extensive developments including nonlinear models for longitudinal outcomes and flexible time-to-event models for survival outcomes, possibly involving competing risks. However, in popular software such as R, the function used to describe the biomarker dynamic is mainly linear in the parameters, and the survival submodel relies on pre-implemented functions (exponential, Weibull, ...). The objective of this work is to extend the code from the *saemix* package (version 3.1 on CRAN) to fit parametric joint models where longitudinal submodels are not necessary linear in their parameters, with full user control over the model function.

Methods: We used the *saemix* package, designed to fit nonlinear mixed-effects models (NLMEM) through the Stochastic Approximation Expectation Maximization (SAEM) algorithm, and extended the main functions to joint model estimation. To compute standard errors (SE) of parameter estimates, we implemented a recently developed stochastic algorithm. A simulation study was proposed to assess (i) the performances of parameter estimation, (ii) the SE computation and (iii) the type I error when testing independence between the two submodels. Four joint models were considered in the simulation study, combining a linear or nonlinear mixed-effects model for the longitudinal submodel, with a single terminal event or a competing risk model.

Results: For all simulation scenarios, parameters were precisely and accurately estimated with low bias and uncertainty. For complex joint models (with NLMEM), increasing the number of chains of the algorithm was necessary to reduce bias, but earlier censoring in the competing risk scenario still challenged the estimation. The empirical SE of parameters obtained over all simulations were very close to those computed with the stochastic algorithm. For more complex joint models (involving NLMEM), some estimates of random effects variances had higher uncertainty and their SE were moderately under-estimated. Finally, type I error was controlled for each joint model.

Conclusions: saemix is a flexible open-source package and we adapted it to fit complex parametric joint models that may not be estimated using standard tools. Code and examples to help users get started are freely available on Github.

Key words: competing risks, joint modeling, mixed-effects model, SAEM algorithm, *saemix*, time-to-event

*Corresponding author: alexandra.lavalley-morelle@inserm.fr

[1]Both authors contributed equally to the work

1. Introduction

In many clinical applications, both longitudinal and survival outcomes are observed. Different approaches exist to analyse separately or jointly both outcomes [1, 2]. A well established method is joint modeling [2], which consists in simultaneously estimating the parameters of both models linked by shared random effects, and has been extensively studied over the last decade [3, 4, 5]. The first developments for joint models involved a linear mixed-effects model describing a single biomarker evolution and a time-to-event model (Cox proportional hazard model) describing the instantaneous risk of event [6]. Many extensions have been proposed including nonlinear mixed-effects models [3], multiple longitudinal models [7], categorical/count data [8] or also competing risks [5].

Software for the joint analysis of longitudinal and event time data have been available for many years, and we present here a non exhaustive list of the tools dedicated to joint modeling available in the more popular software. In a frequentist framework, R [9] has several user-friendly packages as *joineR* [10], *JM*[11], and *joineRML* [12] which allow the user to describe the biomarker dynamics using functions that are linear in the parameters (linear over time or using linear combination of spline functions). Different baseline hazard parametrizations are pre-implemented (exponential, Weibull, competing risks with a cause specific approach, etc.). Similar tools are available in SAS or Stata, such as the macros *%JM* [13] and *%JMfit* [14] for SAS, and the package *stjm* [15] for Stata. To handle multivariate longitudinal submodels, several tools can be used (R packages *joineRML* [12] or *JMbayes* in a Bayesian framework [16]). Another class of joint models allows to stratify the population into classes of patients that are homogeneous both with respect to the evolution of the biomarker(s) and to the occurrence of the event(s). Those joint latent class models are available using the R package *lcmm* [17]). All these tools however only allow to describe the evolution of the biomarker(s) using functions that are linear in the parameters. To the best of our knowledge, only the package *frailtypack* [18] in R allows for a joint model with a mechanistic nonlinear model defined with ordinary differential equations (ODE). However, only one system of ODE is currently implemented and no other expressions can be considered.

Hence, all of these tools reach limitations when it comes to the description of biomarker dynamics because only functions that are linear in the parameters or pre-defined ODE can be used, although some of them do allow for the use of a linear combination of spline functions, giving some flexibility to the model. Therefore, if the evolution of biomarkers obey nonlinear mechanistic models or involve non-linear functions, these packages may not constitute ideal choices. In that case, pharmacometrics software such as NONMEM [19] and Monolix [20] may be a good alternative. They allow to specify very flexible nonlinear joint models and recent studies showed good performance in complex survival settings such as competing risks [21, 5]. Those tools estimate model parameters by maximizing the likelihood using the SAEM algorithm [22]. Theoretical convergence of the algorithm is guaranteed provided that the joint likelihood belongs to the curved exponential family. Although this assumption does not strictly hold for some complex models, including joint models, several published studies successfully used the SAEM algorithm in this setting, showing it converges toward the true values [3, 4, 5].

An important step in parameter estimation is to provide an associated uncertainty. The observed Fisher information matrix (FIM) is usually computed to obtain the variance-covariance matrix associated with maximum-likelihood estimates. Several methods have been described in the literature to compute the FIM exactly or through an approximation [23, 24, 25]. In the context of

nonlinear joint models involving survival data, the most common approach is to use a stochastic approximation algorithm using Louis decomposition principle [23]. However, this approach involves the computation of the second derivative of the complete data likelihood (see [26] for more details), proving computationally cumbersome. Recently, Delattre and Kuhn proposed a new numerical method to obtain a stochastic approximation of the observed FIM in latent variables models [27]. This method has interesting properties, in particular it only requires the calculation of the first derivatives of the complete log-likelihood and it is fully integrated within the SAEM algorithm. While it has been evaluated for nonlinear mixed-effects models and mixture models [27], it hasn't been tested yet in joint models.

In this work, we extended the main functions of the *saemix* package, an open source implementation in R of the SAEM [22] algorithm which provides a flexible and fast algorithm to estimate population parameters in nonlinear mixed-effects models, to handle multiple responses and joint models. We integrated the stochastic algorithm recently developed by Delattre and Kuhn [27] to compute standard errors of parameter estimates. This extension allows us to leverage a tool in a widely used software to fit complex joint models when it cannot be estimated using standard tools. The user has full control over the expression of the joint model (linear or nonlinear functions in the parameters for the longitudinal submodels and parametric survival submodels). This tool has no pre-implemented functions or equations to define the submodels and is therefore less user-friendly than other existing tools, but this is offset by its flexibility, providing an alternative when joint model estimation is not possible using standard tools. We then present a simulation study to show the good properties of estimation of both parameters and standard errors, and a small real-case study based on data available in R software.

This paper is organized as follows: Section 2 describes the methods referring to the *saemix* extension, with the parameters and standard errors estimation procedures, and then the simulation study. In Section 3, we present the results of the simulation study, and the results of an application based on real data. We close the paper with a discussion in Section 4.

2. Methods

2.1. Description of the SAEM algorithm for standard mixed-effects models

A mixed-effects model describing the vector of observed data $y = (y_{ij}; 1 \leq i \leq N,\ 1 \leq j \leq n_i)$ is commonly defined by the following equation:

$$y_{ij} = m(t_{ij}, \psi_i) + g(t_{ij}, \psi_i, \sigma)\, \epsilon_{ij} \tag{1}$$

where m is the structural function describing the observations, $\psi = (\psi_i; 1 \leq i \leq N)$ the vector of unobserved individual parameters involving fixed and individual random effects, and ϵ_{ij} a Gaussian residual error with mean 0 and variance 1 (independent and identically distributed). The objective is to estimate the vector of parameters $\theta = (\mu, \Omega, \sigma)$, with μ the fixed-effects parameters, Ω the variance-covariance matrix of random effects, and σ the residual error model parameters by maximizing the likelihood defined as an integral over the random effects and noted $l(y; \theta)$:

$$l_y(y; \theta) = \prod_i \int_{\mathcal{D}_{\psi_i}} \prod_j p(y_{ij} \,|\, \psi_i; \theta) p(\psi_i; \theta)\, \mathrm{d}\psi_i \tag{2}$$

where $p(y_{ij}\,|\,\psi_i;\theta)$ denotes the density of the longitudinal observations conditionally to the individual parameters, and $p(\psi_i;\theta)$ the density of the individual parameters.

In the case of linear mixed-effects models (LMEM), the estimation can be treated with the regular iterative EM algorithm [28]. At iteration k, data are completed by drawing individual parameters realizations, the E-step computes the conditional expectation of the complete log-likelihood $Q_k(\theta) = \mathbb{E}\left(\log l(y,\psi;\theta)\,|\,y;\theta_{k-1}\right)$, and the M-step computes the value θ_k which maximize $Q_k(\theta)$. Note that the complete data of the model (y,ψ) is composed of the vector of observations and unobserved individual parameters and the complete likelihood is then:

$$l(y,\psi;\theta) = \prod_{i,j} p(y_{ij}\,|\,\psi_i;\theta) \prod_i p(\psi_i;\theta) \tag{3}$$

For nonlinear mixed-effects model (NLMEM), where the structural function does not linearly depend on the random effects, the E-step cannot be computed in a closed-form. An alternative is to use a stochastic version of the EM algorithm, the SAEM algorithm, divided in three steps. At iteration k of the algorithm:

1. Simulation-step : draw $\psi^{(k)}$ from the conditional distribution $p(\cdot|y;\theta_k)$.
2. Stochastic approximation : update $Q_k(\theta)$ according to

$$Q_k(\theta) = (1-\gamma_k)\,Q_{k-1}(\theta) + \gamma_k(\log l(y,\psi^{(k)};\theta)) \tag{4}$$

   where (γ_k) is a decreasing sequence of positive numbers with $\gamma_1 = 1$.
3. Maximization-step : update θ_k according to

$$\theta_{k+1} = \operatorname{Arg}\max_{\theta} Q_k(\theta)$$

In practice, for the mixed-effects model defined in 1, we have:

$$p(y_{ij}\,|\,\psi_i;\theta) = \frac{1}{\sqrt{2\pi}g(t_{ij},\psi_i,\sigma)}\ \exp\left(-\frac{(y_{ij}-m(t_{ij},\psi_i))^2}{2g(t_{ij},\psi_i,\sigma)^2}\right),$$

while the density of individual parameters can be expressed in the case of general Ω (non-diagonal) as:

$$p(\psi_i;\theta) = \frac{1}{(2\pi)^{D/2}|\Omega|^{1/2}}\ \exp\left(-\frac{1}{2}(\psi_i-\mu)^T\ \Omega^{-1}\ (\psi_i-\mu)\right)$$

where D is the number of random effects, and $|.|$ refers to the matrix determinant.

For simple forms of g including the constant and proportional residual error models, the complete likelihood therefore belongs to the curved exponential family and can be decomposed as:

$$l(y,\psi;\theta) = \exp\left(-\Psi(\theta) + < S(y,\psi),\phi(\theta) >\right) \tag{5}$$

The stochastic approximation (step 2) of the algorithm reduces to updating the sufficient statistics (at iteration k):

$$\begin{aligned} S_{1,k} &= (1-\gamma_k)\, S_{1,k-1} + \gamma_k \sum_{i,j} \left(y_{ij} - m(t_{ij}, \psi_i^{(k)})\right)^2 \\ S_{2,k} &= (1-\gamma_k)\, S_{2,k-1} + \gamma_k \sum_{i} \psi_i^{(k)} \\ S_{3,k} &= (1-\gamma_k)\, S_{3,k-1} + \gamma_k \sum_{i} \psi_i^{(k)}\, \psi_i^{(k)\,T} \end{aligned}$$

An example of the derivation of sufficient statistics for a Gaussian mixed-effects model is given in Appendix A.1.

For more general residual variance models where $g = \sigma_a + \sigma_b m^{\sigma_c}$, the complete model does not belong to the exponential family and an additional optimisation step is required to estimate the variance of error parameters.

2.2. Extension of the algorithm for joint models

We now suppose that we also observe survival data, and denote T the variable describing the time-to-event distribution, C the non-informative censoring distribution and δ the event indicator. The couple $(\tilde{T}, \delta)$ is observed with $\tilde{T} = \min(T, C)$. To introduce joint model principle, we consider a continuous biomarker and a single terminal event. In *saemix*, we only consider parametric survival models. A joint model describing simultaneously the longitudinal data by a mixed-effects model and the survival data by a parametric function is usually defined by:

$$h(t \,|\, \psi_i) = h_0(t) \times \exp(\alpha\, m(t, \psi_i)) \tag{6}$$

along with the definition of $m(t, \psi_i)$ in equation 1. $h(t \,|\, \psi_i)$ is the instantaneous risk of event under the proportional hazard assumption. $h_0(t)$ defines the baseline hazard function with parameters h_b, and α is the link coefficient. Here, the vector of population parameters to estimate becomes $\theta = (\mu, \Omega, \sigma, h_b, \alpha)$. The complete data is now (y, ψ, T, δ) and its likelihood is defined by:

$$l(y, T, \delta, \psi; \theta) = \prod_{i,j} p(y_{ij} \,|\, \psi_i; \theta) \prod_i p(T_i, \delta_i \,|\, \psi_i; \theta) \prod_i p(\psi_i; \theta) \tag{7}$$

where $p(T_i, \delta_i \,|\, \psi_i; \theta)$ denotes the density of the survival observations.

The 3 steps of the SAEM algorithm remain as described in the previous subsection. However, the log likelihood has an additional term corresponding to the contribution of the survival part (see equation 7). Survival parameters usually only have a population value without random effects. In *saemix*, we use the exponentiation approach proposed by Kuhn and Lavielle [26]. This trick consists in adding artificial decreasing variability to these parameters in the first K_1 iterations of the algorithm (exploratory phase) and forcing it to decrease progressively to 0 before the final smoothing phase. Using the exponentiation trick, the complete joint likelihood can be expressed again as a function belonging to the curved exponential family. This adaptive method demonstrated good convergence properties [29]. An example of the derivation of sufficient statistics for a joint model is given in Appendix A.2. During the smoothing phase (after iteration K_1), an optimization algorithm is used at each iteration (Nelder-Mead simplex algorithm [30]) to get the estimates of the survival parameters.

2.3. Computation of standard errors

Standard errors (SE) of parameter estimates are derived from the inverse of the observed Fisher Information Matrix (FIM), which is defined in the following equation:

$$\begin{aligned} I(\theta) &= -\partial^2_\theta \log l_{y,\tilde{T},\delta}(y,\tilde{T},\delta;\theta) \\ &= \left(\partial_\theta \log l_{y,\tilde{T},\delta}(y,\tilde{T},\delta;\theta)\right)\left(\partial_\theta \log l_{y,\tilde{T},\delta}(y,\tilde{T},\delta;\theta)\right)^t \end{aligned} \tag{8}$$

As the equation 8 involves the marginal log likelihood, not directly computed in the SAEM algorithm, Delattre and Kuhn [27] proposed a decomposition in the same spirit as the Louis principle [23], based on the score functions of the complete log likelihood:

$$I_{sco}(\theta) = \mathbb{E}_{\psi|y,\tilde{T},\delta;\theta}\left[\partial_\theta \log l(y,\tilde{T},\delta,\psi;\theta)\right]\mathbb{E}_{\psi|y,\tilde{T},\delta;\theta}\left[\left(\partial_\theta \log l(y,\tilde{T},\delta,\psi;\theta)\right)^t\right] \tag{9}$$

This decomposition only requires the calculation of the first derivatives of the complete log likelihood and is therefore interesting from a computational point of view. They also proposed a stochastic approximation method to evaluate this quantity that can be easily integrated within the SAEM algorithm as follows:
Initialize $\Delta_i^0 = O_D$ (matrix of zeroes of dimension D), then

1. Simulation-step : draw $\psi^{(k)}$ from the conditional distribution $p(\cdot|y;\theta_k)$.
2. Stochastic approximations : update $Q_k(\theta)$ and Δ^k according to

$$\begin{aligned} Q_k(\theta) =& (1-\gamma_k)\, Q_{k-1}(\theta) + \gamma_k(\log l(y,\tilde{T},\delta,\psi^{(k)};\theta)) \\ \Delta_k =& (1-\gamma_k)\, \Delta_{k-1} + \gamma_k(\partial_\theta \log l(y,\tilde{T},\delta,\psi^{(k)};\theta_k)) \end{aligned}$$

3. Maximization-step : update θ_k according to

$$\theta_{k+1} = \operatorname{Arg}\max_\theta Q_k(\theta)$$

When convergence is reached (iteration K), we obtain $\hat{\theta}$ the maximum likelihood estimator of θ and the approximation of the FIM $I_{sco}(\hat{\theta})$ based on the score functions is given by:

$$I_{sco}(\hat{\theta}) = \Delta_K \Delta_K^t \tag{10}$$

We implemented this stochastic procedure to evaluate the observed FIM and then derive the variance-covariance matrix of $\hat{\theta}$ defined by $\hat{\Sigma} = I_{sco}(\hat{\theta})^{-1}$.

2.4. Extending the code in saemix to joint models

In this work, we used the development version of *saemix* available on Github, based on the 3.1 version available on CRAN. We modified the main estimation functions to simultaneously fit several outcomes and handle joint models. We also implemented the

algorithm developed by Delattre and Kuhn [27] to estimate the SE of parameters. As users can themselves define the likelihood of the model, *saemix* allows considerable modeling flexibility in a parametric framework.

Our modified code along with examples to use our extensions is available on Github at the following address: `https://github.com/saemixdevelopment/saemixextension/tree/master/joint`. To guide the user, a real case example based on joint model estimation using the code is shown in Appendix D.

In the following, we evaluate the performances of our extension for computing parameter estimates as well as their SE. This evaluation is based on a simulation study covering different scenarios including competing risk settings.

2.5. Simulation study

2.5.1. Objectives

The objective of the simulation is threefold. Firstly, we aim to assess the new *saemix* extension for the estimation of six different models: a linear mixed-effects model (LMEM), a nonlinear mixed-effect model (NLMEM), a joint model with a linear mixed-effects model and a single time-to-event model (JM LMEM-TTE) or a competing risks model (JM LMEM-CR), and a joint model with a nonlinear mixed-effects model and a single time-to-event model (JM NLMEM-TTE) or a competing risks model (JM NLMEM-CR). Secondly, we aim to assess the performances of the SE computation for these models. Finally, in the cases of joint models, we aim to assess the type I error when testing independence between submodels using a Wald test, as the effect of the biomarker evolution on the risk of event is of particular interest in the context of joint modeling. We used simulations to determine whether the type I error of this test is controlled.

2.5.2. Data-generating mechanism

For each of the 6 models presented above, we simulated $M = 200$ datasets of $N = 100$ patients. We assumed a rich design and parameters were chosen in order to have about 50% of failures for single event models, and about 45% of failures from event 1 and 45% of failures from event 2 for competing risks models. Parameter values are shown in Table 1. Code for generating datasets is provided in Appendix B. We now detail the equations of each model.

LMEM and NLMEM

We assumed one biomarker daily measured from t=0 to t=30, with $\{y_{i1}, ..., y_{i30}\}$ the vector of longitudinal observations of subject i (for $i = 1, ..., N$). For LMEM, we define:

$$\begin{aligned} y_{ij} =& m_l(t_{ij}, \psi_i) + g[m_l(t_{ij}, \psi_i), \sigma]\epsilon_{ij} \\ =& \psi_{i0} + \psi_{i1} \times t_{ij} + \sigma\epsilon_{ij} \\ =& (\mu_0 + \eta_{0i}) + (\mu_1 + \eta_{1i}) \times t_{ij} + \sigma\epsilon_{ij} \end{aligned} \tag{11}$$

$$\text{with} \begin{pmatrix} \eta_{i0} \\ \eta_{i1} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \omega_0^2 & 0 \\ 0 & \omega_1^2 \end{pmatrix}\right)$$

For NLMEM, we define:

$$\begin{aligned} y_{ij} =& m_{nl}(t_{ij}, \psi_i) + g[m_{nl}(t_{ij}, \psi_i), \sigma]\epsilon_{ij} \\ =& \psi_{i0} + \psi_{ia} \times \left[\exp\left(-\psi_{i1}\, t_{ij}\right) - \exp\left(-\psi_{i2}\, t_{ij}\right)\right] + \sigma\epsilon_{ij} \\ =& (\mu_0 + \eta_{0i}) + (\mu_a \exp(\eta_{ai})) \times \left[\exp\left(-(\mu_1 \exp(\eta_{1i}))\, t_{ij}\right) - \exp\left(-(\mu_2 \exp(\eta_{2i}))\, t_{ij}\right)\right] + \sigma\epsilon_{ij} \end{aligned} \tag{12}$$

$$\text{with} \begin{pmatrix} \eta_{i0} \\ \eta_{i1} \\ \eta_{i2} \\ \eta_{ia} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \omega_0^2 & 0 & 0 & 0 \\ 0 & \omega_1^2 & 0 & 0 \\ 0 & 0 & \omega_2^2 & 0 \\ 0 & 0 & 0 & \omega_a^2 \end{pmatrix}\right)$$

JM with a LMEM/NLMEM and a time-to-event model

Again, we assumed that biomarker measurements were available every day. Measurements were reported until the time the event occurred or the study ended (t=30). Longitudinal data were simulated according to equation (11) or (12) (LMEM or NLMEM). Time-to-event data were simulated given the cumulative incidence function defined as follows:

$$F_i(t) = \mathbb{P}(T_i < t|\psi_i; \theta) = 1 - \exp\left(-\int_0^t h(s, \psi_i)\, \mathrm{d}s\right) \tag{13}$$

$$h(t, \psi_i) = \begin{cases} h_0 \times \exp\left(\alpha\, m_l(t, \psi_i)\right) & \text{for the linear model} \\ h_0 \times \exp\left(\alpha\, m_{nl}(t, \psi_i)\right) & \text{for the nonlinear model} \end{cases}$$

where h_0 is the baseline constant risk and α the coefficient linking the current predicted biomarker value to the instantaneous risk of event. For each simulated patient, longitudinal observations after the simulated event times were discarded.

JM with a LMEM/NLMEM and a competing risks model (subdistribution approach)

As previously, biomarker measurements were available every day. We now assumed that measurements were reported until time to event 1 (event of interest), time to event 2 (competing event), or end of the study period (t=30). We modeled the competing risks in the subdistribution hazard framework: we defined $T_e = T \times \mathbb{1}_{\delta=e} + \infty \times \mathbb{1}_{\delta\neq e}$ for $e = \{1, 2\}$. The cumulative incidence function for the event 1 is given by:

$$F_1(t) = \mathbb{P}(T_{1i} < t|\psi_i; \theta) = 1 - \exp\Big(-\int_0^t h_1(s, \psi_i)\mathrm{d}s\Big) \tag{14}$$

where

$$h_1(t,\psi_i;\theta) = \begin{cases} h_0(t)\ \exp(\alpha \times m_l(t,\psi_i)) & \text{for the linear model} \\ h_0(t)\ \exp(\alpha \times m_{nl}(t,\psi_i)) & \text{for the nonlinear model} \end{cases}$$

where $h_0(t)$ is the baseline hazard function of Gompertz type defined as : $h_0(t) = \frac{p_1 g_1 \exp(-g_1 t)}{1-p_1(1-exp(-g_1 t)}$.

The subdistribution for the competing risk is then obtained using an exponential distribution with rate $\frac{t}{b}$:

$$\mathbb{P}(T_{2i} < t|\psi_i;\theta) = (1 - F_1(\infty)) \times \left(1 - \exp\left(-\frac{t}{b}\right)\right) \tag{15}$$

and the instantaneous hazard is derived from the previous equation:

$$h_{2i}(t,\psi_i;\theta) = \frac{1}{b}\frac{(1 - F_1(\infty))\exp(-t/b)}{1 - (1 - F_1(\infty))\,(1 - \exp(-t/b))} \tag{16}$$

	LMEM	NLMEM
Fixed effects		
μ_0	4	4
μ_1	0.20	0.15
μ_2	-	0.20
μ_a	-	10
Random effects		
ω_0^2	4	4
ω_1^2	0.10	0.10
ω_2^2	-	0.10
ω_a^2	-	0.50
Error model		
σ	1	1

	JM LMEM-TTE	JM NLMEM-TTE	JM LMEM-CR	JM NLMEM-CR
Survival model				
h_0	0.005	0.005	-	-
p_1	-	-	0.15	0.15
g_1	-	-	0.10	0.10
α	0.20	0.30	0.20	0.30
b	-	-	15	15

Table 1: Values used to simulate longitudinal data (left) and survival data (right).

JM with no-link

In order to evaluate the type I error of the Wald test on the link function, we simulated $M = 1000$ datasets using the models presented in section 2.5.2, but with $\alpha = 0$. We also changed h_0 and p_1 values to keep the same proportion of failures for each model. Hence, h_0 was set to 0.025 and p_1 to 0.50. For each of the replicates, we tested the hypothesis: $H_0 : \alpha = 0 \ \ vs \ \ H_1 : \alpha \neq 0$.

2.5.3. Estimands

For each model and simulated dataset, we estimated θ, the vector of population parameters:

- $\theta = \{\mu, \Omega, \sigma\}$ for LMEM/NLMEM;
- $\theta = \{\mu, \Omega, \sigma, h_0, \alpha\}$ for JM with LMEM/NLMEM and TTE;
- $\theta = \{\mu, \Omega, \sigma, p_1, g_1, \alpha, b\}$ for JM with LMEM/NLMEM and CR

and we also computed $\hat{\Sigma}$, the variance-covariance matrix of $\hat{\theta}$ according to the method described in section 2.3.

2.5.4. Performance measures

In the following, θ denotes the true set of parameters used to simulate data. For the first objective of assessing estimation performances, we report Relative Bias (RB), Relative Root Mean Square Errors (RRMSE) and Relative Estimation Errors (REE).

$$\mathrm{RB}(\hat{\theta}) = \frac{1}{M}\sum_{m=1}^{M}\frac{\hat{\theta}^{(m)}-\theta}{\theta}\times 100 \qquad \mathrm{RRMSE}(\hat{\theta}) = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(\frac{\hat{\theta}^{(m)}-\theta}{\theta}\times 100\right)^2} \qquad \mathrm{REE}^{(m)}(\hat{\theta}) = \frac{\hat{\theta}^{(m)}-\theta}{\theta}\times 100$$

For the second objective of assessing the SE, we report the relative standard errors (RSE) computed using the stochastic algorithm and compare them to the relative empirical SE.

$$\mathrm{SE}_{emp}(\hat{\theta}) = \sqrt{\frac{\sum_m(\hat{\theta}^{(m)}-\bar{\hat{\theta}})^2}{m-1}} \qquad RSE_{emp}(\hat{\theta}) = \frac{\mathrm{SE}_{emp}(\hat{\theta})}{\bar{\hat{\theta}}}$$

where $\bar{\hat{\theta}}$ is the mean estimates of θ.

We also report the coverage rates (CR) of the link coefficients and their 95% confidence intervals (CI), which are of particular interest. CR were defined as the proportion of datasets for which α belonged to $[\hat{\alpha} - 1.96\,\mathrm{SE}(\hat{\alpha}), \hat{\alpha} + 1.96\,\mathrm{SE}(\hat{\alpha})]$. The 95% CI was obtained using the exact Clopper Pearson method.

Finally, for the third objective of assessing the type I error when testing independence between the two submodels, we computed the Wald test statistic: $z^{(m)} = \frac{\hat{\alpha}^{(m)}}{SE^{(m)}_{stoc}(\hat{\alpha}^{(m)})}$, where $\mathrm{SE}^{(m)}_{stoc}$ refers to the stochastic SE computed in simulation m.

2.5.5. Implementation

For each model and each simulated dataset, parameters were estimated by maximizing the likelihood using the extension of *saemix* available on Github. The algorithm is composed of an exploratory and a smoothing phases: in the first K_1 iterations, the algorithm explores the parameter space without memory (that is to say $\gamma_k = 1$) to converge quickly to a neighborhood of the maximum likelihood estimator. In the subsequent K_2 iterations, the stochastic approximation is performed and $\gamma_k = 1/(k - K_1 + 1)$. To achieve faster convergence, we used the "true" values presented in Table 1 as initial values for parameter estimates. We also tested an alternative scenario where initial guesses for parameter estimates were chosen without knowing the true values (Appendix C). For that, we considered the first simulated data set for each model, and fitted only the longitudinal part. We used these parameter estimates to initialize longitudinal submodel parameters. To initialize survival submodel parameters, we used the Kaplan-Meier (for single events) or cumulative incidence function (for competing risks) estimates to decide plausible starting values. The link coefficients were initialized to 0. More details on initial parameters values are given in Appendix C.1. For both scenarios, the default simulated annealing option was kept during the first $K_1/2$ iterations (constraining the variance of the random effects and the residual error parameters to decrease by maximum 3%). Parameters without variability are estimated with an artificial variability during the simulated annealing phase, progressively forced down to 0. We initialized their artificial variances to 1% of their values. To account for model complexity, the number of chains of the algorithm was set to 3 and 10 for joint models involving LMEM and NLMEM, respectively. We also increased the number of iterations to ensure convergence, setting $K_1 = 1000$ iterations in the

exploratory phase (including 500 for the simulated annealing phase) and $K_2 = 500$ for the smoothing phase for all joint models. As no convergence criteria is currently implemented in *saemix*, we used the plots showing the parameter estimation across iterations (provided automatically) for graphical assessment.

We also evaluated computation time by reporting time elapsed for fitting two different joint models according to a given number of patients included in the analysis. The first one is the joint model with a linear mixed-effects model and a time to event model (JM-LMEM-TTE) presented in 2.5.2, and the second one is the joint model with a nonlinear mixed-effects model and a competing risks model (JM-NLMEM-CR) presented in 2.5.2. The JM-LMEM-TTE does not require numerical integration in the definition of the likelihood function and thus is supposed to be fast. The JM-NLMEM-CR requires numerical integration because the likelihood function has no closed form and thus is expected to be more time-consuming. We considered one simulated dataset of $N = 100, 500, 100, 1500$ patients for each model, and reported the time elapsed to estimate parameters and SE.

2.6. Real case studies

2.6.1. Prothrombin example

We consider the R data set *prothro* available in *JM* package, where prothrombin measurements are followed at most 12 days in 488 liver cirrhosis patients. Patients have a median of 6 (Q1-Q3=[3,9]) measurements. Status (dead/censored) at the end of the follow-up is available for each patient. About 60% of patients died during the study. We implemented the joint model presented in equation 17 with a linear mixed-effects model to model the individual prothrombin evolution and a parametric time-to-event model to model the instantaneous risk of death in liver cirrhosis patients.

The linear joint model follows:

$$\begin{aligned} y_{ij} &= m(t_{ij}, \psi_i, z_i) + \sigma\epsilon_{ij} \\ &= (\mu_0 + \eta_{0\,i}) + (\mu_1 + \eta_{1\,i})\, t_{ij} + \sigma\epsilon_{ij} \\ h_i(t, \psi_i) &= h_0 \, \exp(\alpha\, m(t, \psi_i)) \end{aligned} \tag{17}$$

Details on code implementation are given in Appendix D. For the estimation, the default settings were kept (1 chain, 300/100 iterations in the exploratory/smoothing phases). We report parameter and SE estimates.

2.6.2. COVID-19 example

Finally we considered a real data application where we recently applied joint models using the Monolix software [31]. We focused on the prognosis of patients hospitalized for SARS-CoV-2 infection, with the follow-up of 59 biomarkers until death or discharge for at most 30 days. In this work, a selection strategy was defined and applied to build a multivariate joint model with competing risks predicting the risk of patient death involving three biomarkers: blood neutrophil counts, arterial pH and C-reactive protein (CRP). A baseline covariate representing the 4C-score[32] of the patients was also included as a covariate in the survival model. Here we considered the values of those three longitudinal biomarkers, the baseline 4C-score values, and the survival data with the event type (death, discharge or censored observations) and the follow-up times. About 14% of the patients died 30 days

after hospitalization and 72% were discharged. The design is sparser than the previous example: blood neutrophil counts, arterial pH and CRP were reported in $N = 326$, 246 and 318 patients respectively, with a median $[Q_1, Q_3]$ number of observations at 4 [3-8], 3 [2-6] and 4 [2-7] respectively.

We considered the following joint model where y_{ijk} is the observation j of biomarker k in patient i (where biomarkers 1, 2 and 3 refer respectively to neutrophils, pH and CRP):

$$\begin{aligned} y_{ij1} &= m_1(t_{ij1}, \psi_{i1}) + \sigma_1\, m_1(t_{ij1}, \psi_{i1})\epsilon_{ij1} \\ y_{ij2} &= m_2(t_{ij2}, \psi_{i2}) + \sigma_2\epsilon_{ij2} \\ y_{ij3} &= m_3(t_{ij3}, \psi_{i3}) + \sigma_3\epsilon_{ij3} \end{aligned} \tag{18}$$

$$\begin{aligned} h_1(t, \psi_i, Score_i; \theta) &= h_0 \exp\left[\sum_{k=1}^{3}\left(\alpha_k \times (m_k(t, \psi_{ik}) - med_k)\right) + \beta \cdot Score_i\right] \\ h_2(t, \psi_i, Score_i; \theta) &= \frac{1}{b}\frac{(1 - F_1(\infty))\exp(-t/b)}{1 - (1 - F_1(\infty))\,(1 - \exp(-t/b))} \end{aligned}$$

with:

$$\begin{aligned} m_1(t_{ij1}, \psi_{i1}) &= \psi_{i01} + \psi_{ia1} \times \left[\exp\left(\psi_{i11}\, t_{ij1}\right) - \exp\left(\psi_{i21}\, t_{ij1}\right)\right] \\ m_2(t_{ij2}, \psi_{i2}) &= \psi_{i02} + \psi_{i12} \times t_{ij2} \\ m_3(t_{ij3}, \psi_{i3}) &= \psi_{i03} + \psi_{i13} \times t_{ij3} \end{aligned} \tag{19}$$

$Score_i$ refers to the 4C-Score of patient i, and med_k to the median biomarker k value (centering to avoid numerical issues during estimation). Each individual parameter follows a normal distribution, except ψ_{ia1} following a log-normal one. The residual errors $\epsilon_{ijk} \sim \mathcal{N}(0, 1)$. More precisely, biomarker 1 referring to the neutrophils was modeled with a nonlinear mixed-effects model and a proportional error model, biomarker 2 and 3 referring to the pH and the CRP respectively were modeled with linear mixed-effects models and additive error models.

Details on code implementation are given in Appendix D. For the estimation, we first fitted each univariate joint model involving a single biomarker. Initial guesses for parameter estimates were set using the two separate fits for the longitudinal and the survival part. We specified 3 chains and 1000/500 iterations in the exploratory/smoothing phases. We then fitted the multivariate joint model with the initial parameters set at the estimates found during the univariate stage. We used 10 chains to accomodate model complexity, as in the previous published work. As previously, we report parameter and SE estimates.

3. Results

3.1. Results of the simulation study

3.1.1. Parameter and standard errors estimation

Table G.9 presents the RB and RRMSE for the joint models presented in the Methods section, when initial parameter estimates were set to the true values. The link coefficients, which are of particular interest in the context of joint modeling, were precisely and accurately estimated. For these parameters, the RB and RRMSE were low (< 7% for RB and < 25% for RRMSE) for all joint models. Overall, joint model parameters were very well estimated, with increased uncertainty in estimates for some variances of random effects in particular for nonlinear modeling. For these, the RRMSE were greater than 50% but the RB remained moderate. Similar parameter estimates for the corresponding parameters were obtained when fitting only the continuous biomarker alone (LMEM and NLMEM) (see Appendix E.1). We observed more bias in parameter μ_a for JM-NLMEM-CR with a RB around 20%. We investigated the reasons of this bias and present this in Appendix F. The estimation was affected by (1) model complexity, (2) fewer longitudinal observations in this scenario (see Figure F.8 and table F.8) and (3) the presence of extreme values since the median estimation error was 7%. Of note, RB and RRMSE for joint models involving NLMEM computed with 3 chains for the algorithm (similarly to the joint models involving LMEM) showed more bias for μ_a and for the variances of random effects (results shown in Appendix G). Increasing the number of chains of the algorithm helped decrease bias on longitudinal parameters, however the JM-NLMEM-CR scenario still proved too challenging as it made the model difficult to identify with early censoring.

	JM LMEM-TTE		**JM NLMEM-TTE**		**JM LMEM-CR**		**JM NLMEM-CR**	
	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)
Longitudinal model								
Fixed effects								
μ_0	0.10	5.4	0.70	5.1	-0.25	5.6	-0.81	5.4
μ_1	0.93	15.2	-0.33	8.3	-0.38	18.8	0.46	10.8
μ_2	-	-	0.10	7.4	-	-	0.13	10.0
μ_a	-	-	5.5	29.7	-	-	19.7	52.1
Random effects								
ω_0^2	0.95	15.1	-1.7	16.1	-2.1	15.1	-0.49	15.7
ω_1^2	1.5	15.1	1.1	60.6	-3.0	18.1	0.59	84.8
ω_2^2	-	-	0.90	50.0	-	-	-0.52	84.8
ω_a^2	-	-	12.6	50.2	-	-	4.9	66.0
Error model								
σ	0.24	1.5	0.04	1.6	0.15	2.2	0.09	2.5
Survival model								
h_0	-0.20	30.4	0.74	35.4	-	-	-	-
p_1	-	-	-	-	-3.4	35.9	-1.8	32.7
g_1	-	-	-	-	2.2	18.9	-2.0	22.3
α	2.4	13.5	1.7	18.6	6.1	21.6	4.1	20.8
b	-	-	-	-	3.4	20.6	-1.3	16.2

Table 2: Relative Bias (RB) and Relative Root Mean Square Error (RRMSE) obtained on 200 simulations, for the four joint models and when initial parameter estimates were set to the true values

The distribution of REE in following figures (Figure 1 to 4) provide a similar conclusion. It is an efficient and simple way to see

the good properties of parameter estimation, highlighting higher uncertainty on random effect variances for nonlinear joint models. Of note, we obtained similar results for parameters estimates in the NLMEM scenario (see Appendix E.2).

The figures also provide results for RSE estimation. For each joint model, SE estimates were very close to the empirical ones, suggesting good estimation properties. Note however that the RSE for the variances of the random effects were under-estimated for nonlinear models. Similar results were obtained for NLMEM (see Appendix E.2), highlighting an issue inherent to the model and not due to the joint modeling. This finding may be related to the higher uncertainty of REE for those parameters. This issue with the variances however did not affect the SE of the link coefficients. For all link coefficients, the coverage rates were good since all confidence intervals contained 0.95: CR [95% CI] = 0.975 [0.943,0.992], 0.960 [0.922,0.983], 0.935 [0.891,0.965] and 0.930 [0.885,0.961] for JM-LMEM-TTE, JM-NLMEM-TTE, JM-LMEM-CR and JM-NLMEM-CR respectively.

The results under the alternative scenario where initial parameter estimates were set using separate fits are given in Appendix C.2 and Appendix C.3. Briefly, we demonstrate good properties of the algorithms under this scenario, with no inflation of the bias or the RMMSEs. Most importantly, we obtained the same results for SE estimation and coverage rates of the link coefficients.

To assess convergence, we provide in Appendix H the plots reporting parameter estimates across the iterations of the algorithm, for the first simulated dataset using JM-NLMEM-CR, the most complex model expected to require more iterations to achieve convergence. We did not detect convergence issues for both scenario of initial parameters. We also provide in Appendix G the same plots when estimating the model with 3 chains and highlight more variability across iterations of the algorithm compared to the setting with 10 chains.

Results concerning computing time are given in Appendix I. The time needed to estimate joint models increased as the number of patients involved in the analysis increases. Moreover, when the likelihood of the joint model had no analytical expression, the time spent for the estimation process was much more important. However, the complexity of the estimation remained linear with respect to the number of patients involved for both joint models (JM-LMEM-TTE and JM-NLMEM-CR).

3.1.2. Type-I error of Wald test on link coefficients

Figure 5 (plot A) shows the empirical type-1 errors computed over the 1000 simulated data sets (with $\alpha = 0$). Type-I error was controlled for each joint model. With the most complex scenario (JM NLMEM-CR) the Wald test was conservative (empirical type-1 error = 3.2% with 95%CI = [2.2, 4.5]. On plot B, distribution of empirical versus theoretical p-values (on the $-\log_{10}$) yields the same conclusion. We observe a deviation to the right, for JM NLMEM-CR around p-values between 1 and 5%. In particular, observed p-values were lower than the expected ones which corresponds to a conservative test for this range of p-values.

3.2. Real-case applications

3.2.1. Prothrombin example

Table 3 presents parameter estimates for the joint model applied to the *prothro* data. Parameters were well estimated with small RSE (< 30% except for the slope of the linear structural function). The dynamic of prothrombin was predictive of death in liver cirrhosis patients, while decrease of prothrombin over time was associated with a higher risk of death ($\alpha = -0.039$ with

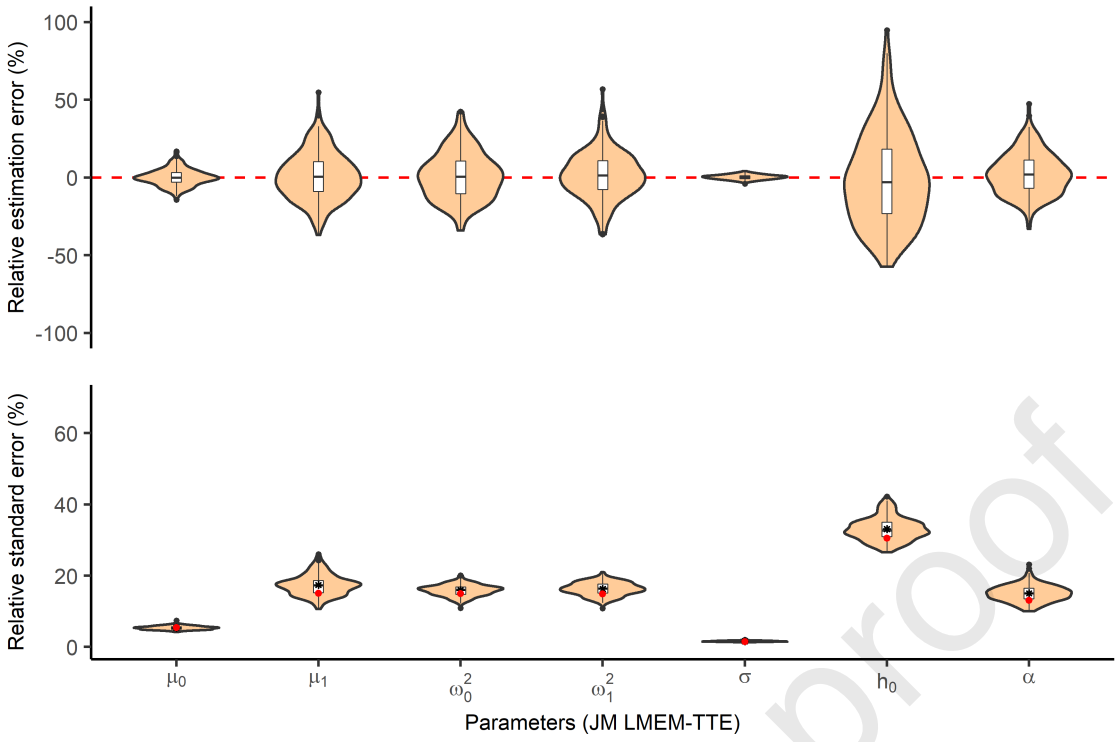


Figure 1: Distribution of the relative estimation errors (top) and relative standard errors (RSE) (bottom) for joint models with a linear mixed-effects model and a time-to-event model, when initial parameter estimates are set to the true values. Stars correspond to the mean of the RSE distribution. Red points correspond to the empirical RSE obtained over the 200 simulations.

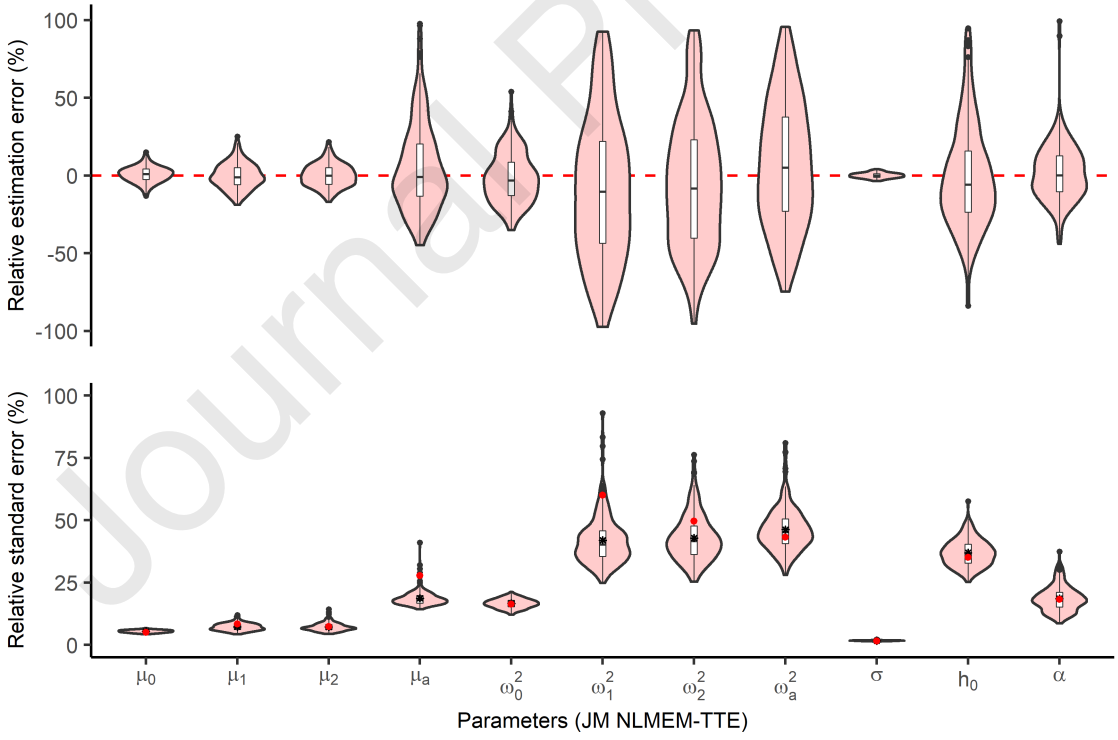


Figure 2: Distribution of the relative estimation errors (top) and relative standard errors (RSE) (bottom) for joint models with a nonlinear mixed-effects model and a time-to-event model, when initial parameter estimates are set to the true values. Stars correspond to the mean of the RSE distribution. Red points correspond to the empirical RSE obtained over the 200 simulations.

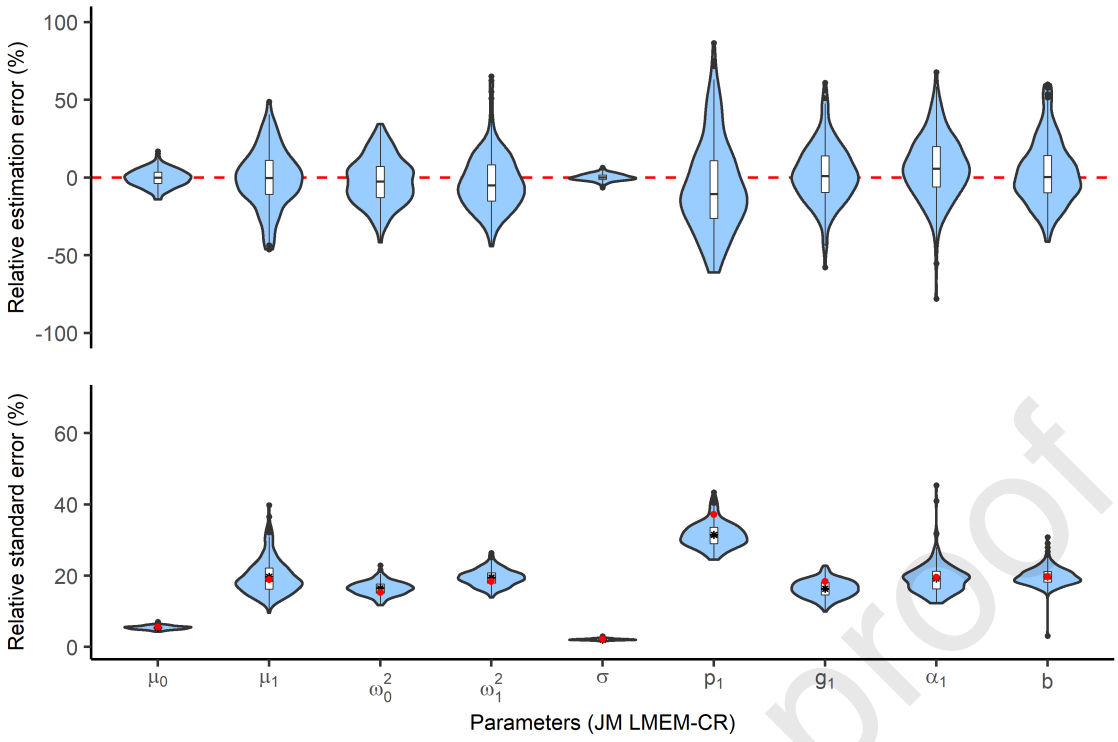


Figure 3: Distribution of the relative estimation errors (top) and relative standard errors (RSE) (bottom) for joint models with a linear mixed-effects model and a competing risks model, when initial parameter estimates are set to the true values. Stars correspond to the mean of the RSE distribution. Red points correspond to the empirical RSE obtained over the 200 simulations.

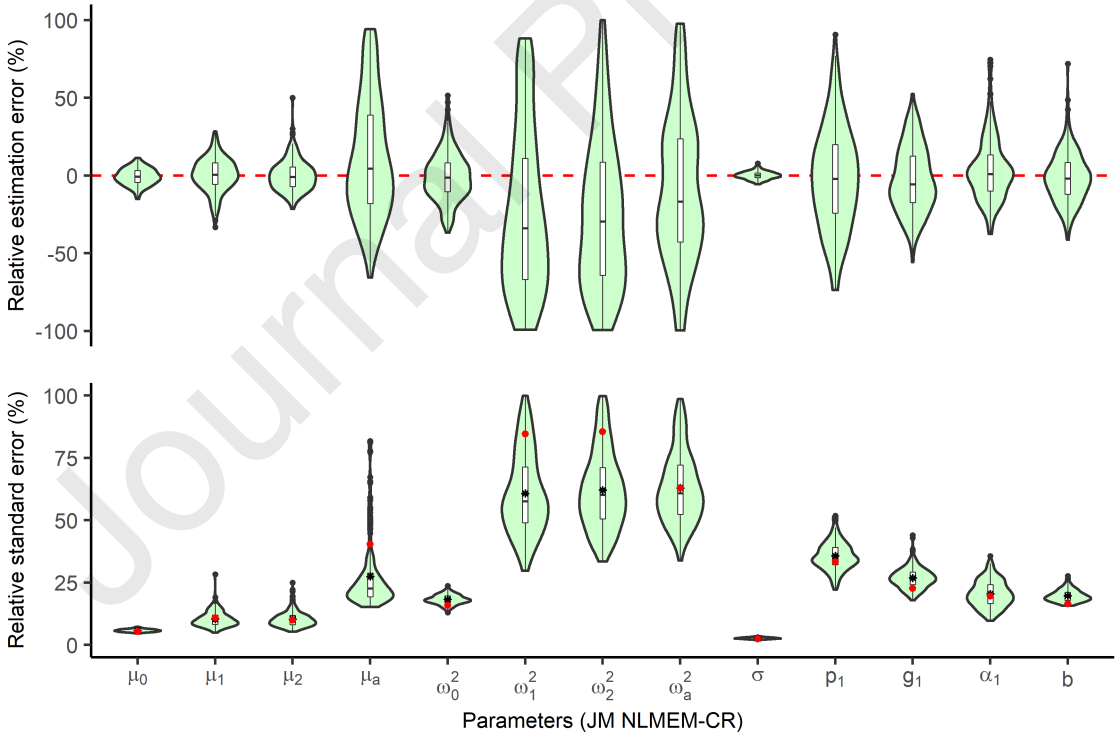


Figure 4: Distribution of the relative estimation errors (top) and relative standard errors (RSE) (bottom) for joint models with a nonlinear mixed-effects model and a competing risks model, when initial parameter estimates are set to the true values. Stars correspond to the mean of the RSE distribution. Red points correspond to the empirical RSE obtained over the 200 simulations.

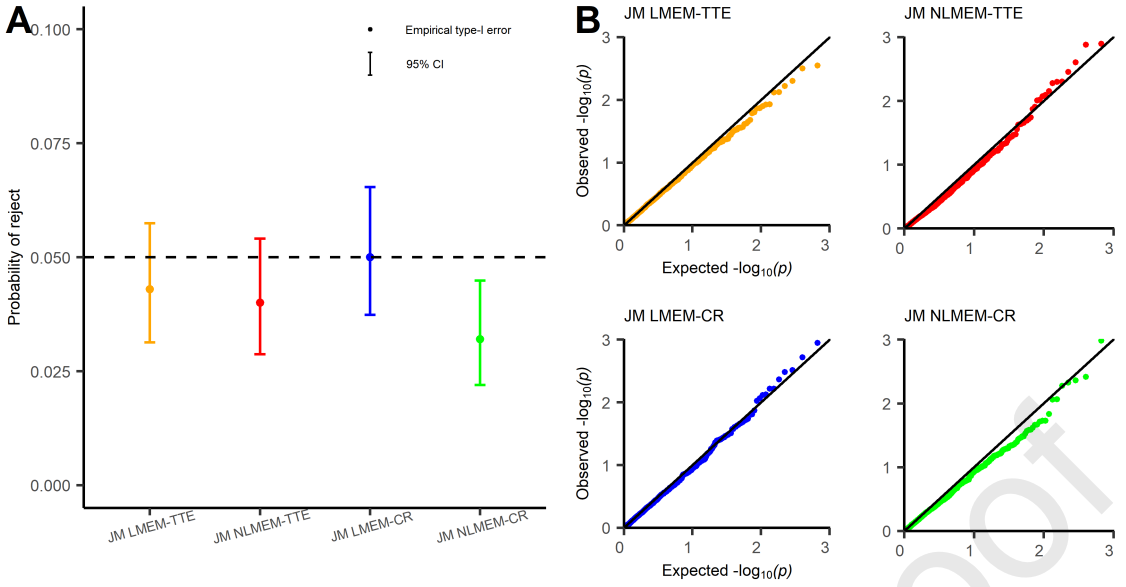


Figure 5: Type I error and its 95% CI for each joint model (A) and empirical $-\log_{10}$ p-values of the Wald statistics versus theorical ones (B)

95%CI = $[-0.045, -0.033]$, $p < 10^{-5}$). As a quality check, we also fitted the model using the *JM* package (*JointModel* function) and obtained similar parameter and RSE estimates (see Appendix J).

Parameters	Estimates	S.E.	R.S.E (%)
Longitudinal model			
Fixed effects			
μ_0	73.3	1.00	1.3
μ_1	0.57	0.33	57.9
Random effects			
ω_0^2	369	31.4	8.5
ω_1^2	16	1.99	12.4
Error model			
σ	17	0.17	1.0
Survival model			
h_0	3.1	0.21	6.8
α	-0.039	0.003	7.7

Table 3: Parameter estimates of the joint model for prothrombin application

3.2.2. *COVID-19 example*

Table 4 presents parameter estimates for the multivariate joint model applied to the COVID-19 example. Similar to the published work, the dynamic of blood neutrophil counts, arterial pH and CRP were found to be predictive of death in patients hospitalized for severe SARS-CoV-2 infection ($\alpha_1 = 0.14$ with 95%CI = $[0.08, 0.20]$, $p < 10^{-5}$, $\alpha_2 = -6.48$ with 95%CI = $[-11.93, -1.03]$, $p = 0.02$, $\alpha_3 = 0.49$ with 95%CI = $[0.32, 0.79]$, $p < 10^{-6}$). That is to say, increases in the blood neutrophil counts and CRP, and decrease of the arterial pH were associated with a higher risk of death while in hospital. Higher 4C-score at admission was also associated with a higher risk of death ($\beta = 0.31[0.17, 0.45]$, $p < 10-6$). These results are similar to those published in [31] when using the SAEM algorithm implemented in the Monolix software.

Parameter (unit)	Value	SE	RSE (%)
Longitudinal submodel			
Blood neutrophil counts			
μ_{0n} $(10^9.L^{-1})$	4.58	0.18	3.9
μ_{1n} $(10^9.L^{-1}.d^{-1})$	-0.15	0.0019	1.3
μ_{2n} $(10^9.L^{-1}.d^{-1})$	-0.17	0.012	6.9
μ_{an} $(10^9.L^{-1})$	7.1	0.97	13.7
ω_{0n} $(10^9.L^{-1})$	4.3	0.41	9.4
ω_{1n} $(10^9.L^{-1}.d^{-1})$	0.011	0.0027	24.3
ω_{2n} $(10^9.L^{-1}.d^{-1})$	0.003	0.0011	36.4
ω_{an} $(10^9.L^{-1})$	0.55	0.19	35.7
σ_{bn}	0.32	0.003	0.9
arterial pH			
μ_{0p}	7.44	0.0042	0.06
μ_{1p} (d^{-1})	0.001	0.00078	0.08
ω_{0p}	0.0015	0.00016	10.9
ω_{1p} (d^{-1})	1.5×10^{-5}	4.5×10^{-6}	29.3
σ_{ap}	0.055	0.00031	0.6
C-reactive protein			
μ_{0c} $(log(mg.L^{-1}))$	4.17	0.082	2.0
μ_{1c} $(log(mg.L^{-1}).d^{-1})$	-0.14	0.011	7.4
ω_{0c} $(log(mg.L^{-1}))$	0.85	0.10	12.1
ω_{1c} $(log(mg.L^{-1}).d^{-1})$	0.014	0.0025	18.1
σ_{ac} $(log(mg.L^{-1}))$	0.72	0.0075	1.0
Survival submodel			
Death			
h_0	0.0003	0.79	2.7×10^5
α_{1n} $(L.10^{-9})$	0.14	0.033	23.7
α_{1p}	-6.48	2.78	42.9
α_{1c} $(-log(mg.L^{-1}))$	0.55	0.12	21.4
β_1	0.31	0.070	22.5
Discharge			
b	12.1	1.08	8.9

Table 4: Parameter estimates of the multivariate joint model for COVID-19 application

4. Conclusion

In this work, we modified the code from the R package *saemix* to fit joint models. This tool allows to fit a wide range of parametric joint models with an explicit expression of the likelihood. As a non exhaustive list of examples, we applied it to linear joint models with a single event model, and to more complex joint models including NLMEM and competing risks. Multiple longitudinal models can be easily considered, if various biomarkers are available. Other error model functions (proportional or combined) can also be considered, as well as baseline covariates (not presented in this article but see Github for an example). In comparison with other packages available for joint model estimation where modeling is mainly limited to linear forms for the longitudinal response and pre-defined forms for the survival model [10, 11, 12, 18], *saemix* lets users fit very specific joint models by defining a personalized likelihood function.

In terms of standard error calculation, standard algorithms used in the framework of Gaussian responses based on linearization of the likelihood such as FO or FOCE [24] cannot be used in the framework of joint models with survival responses. The current

version of *saemix* offers several bootstrap approaches but has not yet implemented numerical or stochastic computation of the FIM for discrete data and survival data models. We opted to implement a recently developed stochastic algorithm [27] to estimate the FIM of the model. The advantage is that it integrates perfectly with the SAEM algorithm, and that it only involves first derivatives. Alternatively, the Monolix software uses Louis' decomposition principle combined with a stochastic algorithm to estimate the FIM of a joint model, but it is more computationally expensive because it involves second derivatives and it is run after the SAEM algorithm.

We showed through the simulation study good performances for both the estimation of the parameters in the joint model and for the computation of the SE. We however found a higher uncertainty in some estimates of random effects variances, for complex models (involving nonlinear modeling) and also a small under-estimation of relative SE for the same parameters. Of note, similar results are obtained using Monolix software, which highlights difficulties inherent to the proposed model [5]. For complex joint models involving NLMEM, increasing the number of chains of the algorithm is useful to reduce bias on longitudinal parameters. However, for JM NLMEM-CR, a moderate bias on a fixed-effects parameter (around 20%) was observed. This bias can be due to the model complexity coupled with the number of longitudinal observations. This bias resulted from a challenging model when coupled with a lower number of longitudinal observations. Indeed, observations were censored early in the JM NLMEM-CR scenario and this along with the variability made the parameter describing the amplitude of the biomarker evolution difficult to estimate. For all joint models considered, the link coefficients, which are of particular interest in a joint modeling context, were accurately and precisely estimated, as were their SE. Of note, in some studies joint modeling is used to correct for missing not at random values in the longitudinal process. Longitudinal parameters can be the main focus of interest in that case. We highlighted good estimation performances for longitudinal parameters, provided the number of chains and the design are adequate. Finally, we showed that type I error was controlled for each proposed model when assessing the significance of those coefficients (Wald test).

We applied our approach to data collected in liver cirrhosis patients. The joint model was easily implemented and showed that a decrease of the prothrombin over time is associated with a higher risk of death. Although the objective was not to compare the results with the existing *JM* package, very similar results were obtained in this case. However, more complex longitudinal and survival models can be explored with the *saemix* extension compared to currently available packages in R.

We provide code that is *ad hoc* but available and editable by any user. In the model definition, the technical part remains the writing the likelihood function. It has the advantage of being flexible in the sense that any parametric joint model can be considered, but the disadvantage of needing to be written explicitly so the user has to perfectly know the model expression. In the presence of survival data, the likelihood involves integrals that may have no close form depending on the model complexity. This issue can be overcome by using numerical integration, which is simple to implement but also makes parameter estimation much more time consuming. Other known limitations of the *saemix* package can be found in the documentation. These limitations also concern the present extension, in that that *saemix* currently doesn't handle left-censored data, inter-occasion variability or mixture models. Finally, further developments are required to optimize numerical integration and improve computation time. Studies comparing performances of parameter estimation and computation times with existing software should then be performed. As a first step, preliminary comparisons made with the JM package using default settings on the JM LMEM-TTE (the only model estimable using

JM) are presented here in Appendix K.

We conclude by acknowledging the limits of this work. In the simulation study, we assessed the performances of the developed algorithm only in a rich design. It could be interesting to report estimation performances when biomarker is less frequent measured. In such case, we could expect higher uncertainty on parameter estimates, as has been shown in the literature using other software [31]. Moreover, the code has not yet been extended to allow for correlation between random effects (independence between random effects is a strong assumption not met in many applications) and lacks advanced diagnostic tools. However, users have access to estimated individual parameters and individual predictions and hence, can build their own diagnostic plots such as observations vs predictions plots, normalised prediction distribution errors (NPDE) [33, 34], or individual residuals for the longitudinal part [34], and martingal or Cox-snell residuals for the survival part [35]. Finally, onvergence criteria could also be defined and integrated to stop iterations when parameter estimates stabilize. It is worth noting that the use of this code is mainly for advanced statisticians and for an experienced public in the use of joint models.

All the extended functions of the R package *saemix* are available on Github at the following address: `https://github.com/saemixdevelopment/saemixextension/tree/master/joint`.

Acknowledgments

The authors gratefully acknowledge Dr Maud Delattre (UMR MIA-Paris, AgroParisTech, INRAE, Université Paris-Saclay, 75005, Paris, France) for her kindness, availability and help with the FIM computation.

References

[1] Magdalena Murawska, Dimitris Rizopoulos, and Emmanuel Lesaffre. A Two-Stage Joint Model for Nonlinear Longitudinal Response and a Time-to-Event with Application in Transplantation Studies. Journal of Probability and Statistics, 2012:e194194, 2012.

[2] Dimitris Rizopoulos. Joint models for longitudinal and time-to-event data: with applications in R. Chapman & Hall / CRC biostatistics series. Boca Raton, Fla., 2012.

[3] Solène Desmée, France Mentré, Christine Veyrat-Follet, and Jérémie Guedj. Nonlinear Mixed-effect Models for Prostate-specific Antigen Kinetics and Link with Survival in the Context of Metastatic Prostate Cancer: A Comparison by Simulation of Two-stage and Joint Approaches. The AAPS Journal, 17(3):691–699, 2015.

[4] Marion Kerioui, François. Mercier, Julie. Bertrand, Coralie. Tardivon, René. Bruno, Jérémie. Guedj, and Solène. Desmée. Bayesian inference using Hamiltonian Monte-Carlo algorithm for nonlinear joint modeling in the context of cancer immunotherapy. Statistics in Medicine, 39(30):4853–4868, 2020.

[5] Alexandra Lavalley-Morelle, Jean-François Timsit, France Mentré, Jimmy Mullaert, and The OUTCOMEREA Network. Joint modeling under competing risks: Application to survival prediction in patients admitted in Intensive Care Unit for sepsis with daily Sequential Organ Failure Assessment score assessments. CPT: Pharmacometrics & Systems Pharmacology, 11(1472–1484), 2022.

[6] Michael S. Wulfsohn and Anastasios A. Tsiatis. A Joint Model for Survival and Longitudinal Data Measured with Error. Biometrics, 53(1):330–339, 1997.

[7] Jeffrey D. Long and James A. Mills. Joint modeling of multivariate longitudinal data and survival data in several observational studies of Huntington's disease. BMC Medical Research Methodology, 18(1):138, 2018.

[8] Jaeun Choi, Jianwen Cai, Donglin Zeng, and Andrew F. Olshan. Joint Analysis of Survival Time and Longitudinal Categorical Outcomes. Statistics in Biosciences, 7(1):19–47, 2015.

[9] R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, 2023.

[10] Pete Philipson, Ines Sousa, Peter J. Diggle, Paula Williamson, Ruwanthi Kolamunnage-Dona, Robin Henderson, and Graeme L. Hickey. joineR: Joint Modelling of Repeated Measurements and Time-to-Event Data, 2018.

[11] Dimitris Rizopoulos. JM: An R package for the joint modelling of longitudinal and time-to-event data. Journal of Statistical Software, 35(9):1–33, 2010.

[12] Graeme L. Hickey, Pete Philipson, and Andrea Jorgensen and. joinerml: a joint model and software package for time-to-event and multivariate longitudinal outcomes. BMC Medical Research Methodology, 18(1), 2018.

[13] Alberto Garcia-Hernandez and Dimitris Rizopoulos. %JM: A SAS Macro to Fit Jointly Generalized Mixed Models for Longitudinal Data and Time-to-Event Responses. Journal of Statistical Software, 84:1–29, 2018.

[14] Danjie Zhang, Ming-Hui Chen, Joseph G. Ibrahim, Mark E. Boye, and Wei Shen. JMFit: A SAS Macro for Joint Models of Longitudinal and Survival Data. Journal of Statistical Software, 71:1–24, 2016.

[15] Michael J. Crowther. STJM: Stata module to fit shared parameter joint models of longitudinal and survival data. Statistical Software Components, 2013.

[16] Dimitris Rizopoulos. The R package JMbayes for fitting joint models for longitudinal and time-to-event data using mcmc. Journal of Statistical Software, 72(7):1–45, 2016.

[17] Cécile Proust-Lima, Viviane Philipps, and Benoit Liquet. Estimation of extended mixed models using latent classes and latent processes: The R package lcmm. Journal of Statistical Software, 78(2):1–56, 2017.

[18] Virginie Rondeau, Yassin Mazroui, and Juan R. Gonzalez. frailtypack: An R package for the analysis of correlated survival data with frailty models using penalized likelihood estimation or parametrical estimation. Journal of Statistical Software, 47(4):1–28, 2012.

[19] Robert J. Bauer. NONMEM Tutorial Part I: Description of Commands and Options, With Simple Examples of Population Analysis. CPT: Pharmacometrics & Systems Pharmacology, 8(8):525–537, 2019.

[20] Pauline Traynard, Géraldine Ayral, Monika Twarogowska, and Jonathan Chauvin. Efficient Pharmacokinetic Modeling Workflow With the MonolixSuite: A Case Study of Remifentanil. CPT: Pharmacometrics & Systems Pharmacology, 9(4):198–210, 2020.

[21] Sreenath M. Krishnan, Lena E. Friberg, René Bruno, Ulrich Beyer, Jin Y. Jin, and Mats O. Karlsson. Multistate model for pharmacometric analyses of overall survival in HER2-negative breast cancer patients treated with docetaxel. CPT: Pharmacometrics & Systems Pharmacology, 10(10):1255–1266, 2021.

[22] Bernard Delyon, Marc Lavielle, and Eric Moulines. Convergence of a stochastic approximation version of EM algorithm. The Annals of Statistics, 27:94, 1999.

[23] Thomas A. Louis. Finding the Observed Information Matrix When Using the EM Algorithm. Journal of the Royal Statistical Society: Series B (Methodological), 44(2):226–233, 1982.

[24] Mary J. Lindstrom and Douglas M. Bates. Nonlinear Mixed Effects Models for Repeated Measures Data. Biometrics, 46(3):673–687, 1990.

[25] Marie-Karelle Riviere, Sebastian Ueckert, and France Mentré. An MCMC method for the evaluation of the Fisher information matrix for non-linear mixed effect models. Biostatistics, 17(4):737–750, 2016.

[26] Estelle Kuhn and Marc Lavielle. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics & Data Analysis, 49(4):1020–1038, 2005.

[27] Maud Delattre and Estelle Kuhn. Estimating Fisher Information Matrix in Latent Variable Models based on the Score Function, 2023.

[28] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological), 39(1):1–38, 1977.

[29] Vianney Debavelaere and Stéphanie Allassonnière. On the curved exponential family in the Stochastic Approximation Expectation Maximization Algorithm. ESAIM: Probability and Statistics, 25:408–432, 2021.

[30] John A. Nelder and Roger Mead. A Simplex Method for Function Minimization. The Computer Journal, 7(4):308–313, 1965.

[31] Alexandra Lavalley-Morelle, Nathan Peiffer-Smadja, Simon B. Gressens, Bérénice Souhail, Alexandre Lahens, Agathe Bounhiol, François-Xavier Lescure, France Mentré, and Jimmy Mullaert. Multivariate joint model under competing risks to predict death of hospitalized patients for SARS-CoV-2 infection. Biometrical Journal, n/a(n/a):2300049, July 2023.

[32] Stephen R Knight, Antonia Ho, Riinu Pius, Iain Buchan, Gail Carson, Thomas M Drake, Jake Dunning, Cameron J Fairfield, Carrol Gamble, Christopher A Green, Rishi Gupta, Sophie Halpin, Hayley E Hardwick, Karl A Holden, Peter W Horby, Clare Jackson, Kenneth A Mclean, Laura Merson, Jonathan S Nguyen-Van-Tam, Lisa Norman, Mahdad Noursadeghi, Piero L Olliaro, Mark G Pritchard, Clark D Russell, Catherine A Shaw, Aziz Sheikh, Tom Solomon,

Cathie Sudlow, Olivia V Swann, Lance CW Turtle, Peter JM Openshaw, J Kenneth Baillie, Malcolm G Semple, Annemarie B Docherty, Ewen M Harrison, J Kenneth Baillie, Malcolm G Semple, Peter JM Openshaw, Gail Carson, Beatrice Alex, Benjamin Bach, Wendy S Barclay, Debby Bogaert, Meera Chand, Graham S Cooke, Annemarie B Docherty, Jake Dunning, Ana da Silva Filipe, Tom Fletcher, Christopher A Green, Ewen M Harrison, Julian A Hiscox, Antonia Ying Wai Ho, Peter W Horby, Samreen Ijaz, Saye Khoo, Paul Klenerman, Andrew Law, Wei Shen Lim, Alexander J Mentzer, Laura Merson, Alison M Meynert, Mahdad Noursadeghi, Shona C Moore, Massimo Palmarini, William A Paxton, Georgios Pollakis, Nicholas Price, Andrew Rambaut, David L Robertson, Clark D Russell, Vanessa Sancho-Shimizu, Janet T Scott, Louise Sigfrid, Tom Solomon, Shiranee Sriskandan, David Stuart, Charlotte Summers, Richard S Tedder, Emma C Thomson, Ryan S Thwaites, Lance CW Turtle, Maria Zambon, Hayley Hardwick, Chloe Donohue, Jane Ewins, Wilna Oosthuyzen, Fiona Griffiths, Lisa Norman, Riinu Pius, Tom M Drake, Cameron J Fairfield, Stephen Knight, Kenneth A Mclean, Derek Murphy, Catherine A Shaw, Jo Dalton, Michelle Girvan, Egle Saviciute, Stephanie Roberts, Janet Harrison, Laura Marsh, Marie Connor, Sophie Halpin, Clare Jackson, Carrol Gamble, Gary Leeming, Andrew Law, Ross Hendry, James Scott-Brown, William Greenhalf, Victoria Shaw, Sarah McDonald, Katie A Ahmed, Jane A Armstrong, Milton Ashworth, Innocent G Asiimwe, Siddharth Bakshi, Samantha L Barlow, Laura Booth, Benjamin Brennan, Katie Bullock, Benjamin WA Catterall, Jordan J Clark, Emily A Clarke, Sarah Cole, Louise Cooper, Helen Cox, Christopher Davis, Oslem Dincarslan, Chris Dunn, Philip Dyer, Angela Elliott, Anthony Evans, Lewis WS Fisher, Terry Foster, Isabel Garcia-Dorival, Willliam Greenhalf, Philip Gunning, Catherine Hartley, Antonia Ho, Rebecca L Jensen, Christopher B Jones, Trevor R Jones, Shadia Khandaker, Katharine King, Robyn T Kiy, Chrysa Koukorava, Annette Lake, Suzannah Lant, Diane Latawiec, L Lavelle-Langham, Daniella Lefteri, Lauren Lett, Lucia A Livoti, Maria Mancini, Sarah McDonald, Laurence McEvoy, John McLauchlan, Soeren Metelmann, Nahida S Miah, Joanna Middleton, Joyce Mitchell, Shona C Moore, Ellen G Murphy, Rebekah Penrice-Randal, Jack Pilgrim, Tessa Prince, Will Reynolds, P Matthew Ridley, Debby Sales, Victoria E Shaw, Rebecca K Shears, Benjamin Small, Krishanthi S Subramaniam, Agnieska Szemiel, Aislynn Taggart, Jolanta Tanianis-Hughes, Jordan Thomas, Erwan Trochu, Libby van Tonder, Eve Wilcock, J Eunice Zhang, Kayode Adeniji, Daniel Agranoff, Ken Agwuh, Dhiraj Ail, Ana Alegria, Brian Angus, Abdul Ashish, Dougal Atkinson, Shahedal Bari, Gavin Barlow, Stella Barnass, Nicholas Barrett, Christopher Bassford, David Baxter, Michael Beadsworth, Jolanta Bernatoniene, John Berridge, Nicola Best, Pieter Bothma, David Brealey, Robin Brittain-Long, Naomi Bulteel, Tom Burden, Andrew Burtenshaw, Vikki Caruth, David Chadwick, Duncan Chambler, Nigel Chee, Jenny Child, Srikanth Chukkambotla, Tom Clark, Paul Collini, Catherine Cosgrove, Jason Cupitt, Maria-Teresa Cutino-Moguel, Paul Dark, Chris Dawson, Samir Dervisevic, Phil Donnison, Sam Douthwaite, Ingrid DuRand, Ahilanadan Dushianthan, Tristan Dyer, Cariad Evans, Chi Eziefula, Chrisopher Fegan, Adam Finn, Duncan Fullerton, Sanjeev Garg, Sanjeev Garg, Atul Garg, Jo Godden, Arthur Goldsmith, Clive Graham, Elaine Hardy, Stuart Hartshorn, Daniel Harvey, Peter Havalda, Daniel B Hawcutt, Maria Hobrok, Luke Hodgson, Anita Holme, Anil Hormis, Michael Jacobs, Susan Jain, Paul Jennings, Agilan Kaliappan, Vidya Kasipandian, Stephen Kegg, Michael Kelsey, Jason Kendall, Caroline Kerrison, Ian Kerslake, Oliver Koch, Gouri Koduri, George Koshy, Shondipon Laha, Susan Larkin, Tamas Leiner, Patrick Lillie, James Limb, Vanessa Linnett, Jeff Little, Michael MacMahon, Emily MacNaughton, Ravish Mankregod, Huw Masson, Elijah Matovu, Katherine McCullough, Ruth McEwen, Manjula Meda, Gary Mills, Jane Minton, Mariyam Mirfenderesky, Kavya Mohandas, Quen Mok, James Moon, Elinoor Moore, Patrick Morgan, Craig Morris, Katherine Mortimore, Samuel Moses, Mbiye Mpenge, Rohinton Mulla, Michael Murphy, Megan Nagel, Thapas Nagarajan, Mark Nelson, Igor Otahal, Mark Pais, Selva Panchatsharam, Hassan Paraiso, Brij Patel, Justin Pepperell, Mark Peters, Mandeep Phull, Stefania Pintus, Jagtur Singh Pooni, Frank Post, David Price, Rachel Prout, Nikolas Rae, Henrik Reschreiter, Tim Reynolds, Neil Richardson, Mark Roberts, Devender Roberts, Alistair Rose, Guy Rousseau, Brendan Ryan, Taranprit Saluja, Aarti Shah, Prad Shanmuga, Anil Sharma, Anna Shawcross, Jeremy Sizer, Richard Smith, Catherine Snelson, Nick Spittle, Nikki Staines, Tom Stambach, Richard Stewart, Pradeep Subudhi, Tamas Szakmany, Kate Tatham, Jo Thomas, Chris Thompson, Robert Thompson, Ascanio Tridente, Darell Tupper-Carey, Mary Twagira, Andrew Ustianowski, Nick Vallotton, Lisa Vincent-Smith, Shico Visuvanathan, Alan Vuylsteke, Sam Waddy, Rachel Wake, Andrew Walden, Ingeborg Welters, Tony Whitehouse, Paul Whittaker, Ashley Whittington, Meme Wijesinghe, Martin Williams, Lawrence Wilson, Sarah Wilson, Stephen Winchester, Martin Wiselka, Adam Wolverson, Daniel G Wooton, Andrew Workman, Bryan Yates, and Peter Young. Risk stratification of patients admitted to hospital with covid-19 using the ISARIC WHO Clinical Characterisation Protocol: development and validation of the 4C Mortality Score. British Medical Journal, 370:m3339, September 2020.

[33] Emmanuelle Comets, Karl Brendel, and France Mentré. Computing normalised prediction distribution errors to evaluate nonlinear mixed-effect models: the npde add-on package for R. Computer Methods and Programs in Biomedicine, 90(2):154–166, 2008.

[34] Emmanuelle Comets and France Mentré. Developing tools to evaluate non-linear mixed effect models: 20 years on the npde adventure. AAPS Journal, 23(4):75, 2021.

[35] David. Cox and E. Joyce Snell. A General Definition of Residuals. Journal of the Royal Statistical Society. Series B (Methodological), 30(2):248–275, 1968.

Appendix A. Derivation of Sufficient Statistics for a Gaussian mixed-effects model and for a joint model

Appendix A.1. Mixed-effects model with homoscedastic variance

We consider the mixed-effects model with an additive error defined as:

$$y_{ij} = m(t_{ij}, \psi_i) + \sigma\, \epsilon_{ij} \tag{A.1}$$

with ψ_i the individual parameters ($i = 1, ..N$), a function of fixed-effects μ_d and random effects $\eta_{d\,i}$ ($1 \leq d \leq D$). The vectors of random effects are assumed to be normally distributed with diagonal variance covariance matrix Ω. ϵ_{ij} is the residual error following a normal distribution with mean 0 and variance 1. For simplicity in the rest of this section, we will assume that the distribution of the individual parameters is normal:

$$\psi_{d,i} = \mu_d + \eta_{d,i}$$

but the equations below generalize to models where each component $\psi_{d,i}$ can be transformed to a linear function of μ_d, $\eta_{d,i}$, and optionally covariates $z_{cov,i}$ and associated coefficients $\beta_{d,cov}$.

The complete likelihood is:

$$\begin{aligned} l(y, \psi; \theta) &= \prod_{ij} p(y_{ij}|\psi_i; \theta) \prod_i p(\psi_i; \theta) \\ &= \prod_{ij} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_{ij} - m(t_{ij}, \psi_i))^2}{2\sigma^2}\right) \prod_i \frac{1}{(2\pi)^{D/2}|\Omega|^{1/2}} \exp\left(-\frac{1}{2}(\psi_i - \mu)^T\, \Omega^{-1}\, (\psi_i - \mu)\right) \\ &= \prod_{ij} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_{ij} - m(t_{ij}, \psi_i))^2}{2\sigma^2}\right) \prod_i \prod_d \frac{1}{(2\pi)^{1/2}\omega_d} \exp\left(-\frac{(\psi_{d\,i} - \mu_d)^2}{2\omega_d^2}\right) \end{aligned}$$

Taking the logarithm of previous equation, we can write:

$$\begin{aligned} \ln l(y, \psi; \theta) = &\sum_{i,j} -\ln\, \sqrt{2\pi}\sigma - \sum_{i,j} \frac{(y_{ij} - m(t_{ij}, \psi_i))^2}{2\, \sigma^2} \\ &+ N \sum_d -\ln\left((2\pi)^{1/2}\omega_d\right) - \sum_i \sum_d \frac{(\psi_{d\,i})^2}{2\omega_d^2} + \sum_i \sum_d \frac{(\mu_d\, \psi_{d\,i})}{\omega_d^2} - N \sum_d \frac{(\mu_d)^2}{2\omega_d^2} \end{aligned} \tag{A.2}$$

The terms in red depend only on the population parameters, while the terms in blue depend on the data and the individual and population parameters. In the framework of the exponential family, the red terms correspond to $\Psi(\theta)$ in equation 5, so that:

$$\Psi(\theta) = \sum_{i,j} \ln(\sqrt{2\pi}\sigma) + N \sum_d \ln(\sqrt{2\pi}\omega_d) + N \sum_d \frac{(\mu_d)^2}{2\omega_d^2}$$

The blue terms can now be factorised as the scalar product of two terms which conform to equation (5). We define:

$$S(y,\phi) = \begin{pmatrix} 1/2(y_{1,1} - m(t_{1,1},\psi_1))^2 \\ 1/2(y_{1,2} - m(t_{1,2},\psi_1))^2 \\ \dots \\ 1/2(y_{1,n_1} - m(t_{1,n_1},\psi_1))^2 \\ 1/2(y_{2,1} - m(t_{2,1},\psi_2))^2 \\ \dots \\ 1/2(y_{2,n_2} - m(t_{2,n_2},\psi_2))^2 \\ \dots \\ 1/2(y_{N,1} - m(t_{N,1},\psi_N))^2 \\ \dots \\ 1/2(y_{N,n_N} - m(t_{N,n_N},\psi_N))^2 \\ \\ \psi_{1\,1} \\ \dots \\ \psi_{1\,N} \\ \dots \\ \psi_{D\,1} \\ \dots \\ \psi_{D\,N} \\ \\ 1/2(\psi_{1\,1})^2 \\ \dots \\ 1/2(\psi_{1\,N})^2 \\ \dots \\ 1/2(\psi_{D\,1})^2 \\ \dots \\ 1/2(\psi_{D\,N})^2 \end{pmatrix}$$

composed of 3 stacked vectors

$$S_y = \left(\tfrac{1}{2}(y_{i,j} - m(t_{i,j},\psi_i))^2\right)_{i=1,..,N,j=1,..,n_i} \qquad S_\psi = \left(\psi_{d\,i}\right)_{i=1,..,N,d=1,..,D} \qquad S_{\psi^2} = \left(\tfrac{1}{2}(\psi_{d\,i})^2\right)_{i=1,..,N,d=1,..,D}$$

We then define the function $\phi(\theta)$ as a similar stacked vector composed of 3 parts of the same sizes as S_y, S_ψ and S_{ψ^2} respectively:

$$\phi_1(\theta) = \begin{pmatrix} -\frac{1}{\sigma^2} \\ \dots \\ -\frac{1}{\sigma^2} \end{pmatrix}_{\sum_i n_i \text{times}} \quad \phi_2(\theta) = \begin{pmatrix} \frac{\mu_1}{\omega_1^2} \\ \dots \\ \frac{\mu_1}{\omega_1^2} \\ \dots \\ \frac{\mu_D}{\omega_D^2} \\ \dots \\ \frac{\mu_D}{\omega_D^2} \end{pmatrix}_{N\text{times}} \quad \phi_3(\theta) = \begin{pmatrix} -\frac{1}{\omega_1^2} \\ \dots \\ -\frac{1}{\omega_1^2} \\ \dots \\ -\frac{1}{\omega_D^2} \\ \dots \\ -\frac{1}{\omega_D^2} \end{pmatrix}_{N\text{times}}$$

We can see that

$$< S_y, \phi_1 >= -\sum_{i,j} \frac{(y_{ij} - m(x_{ij}, \psi_i))^2}{2\,\sigma^2}$$

$$< S_\psi, \phi_2 >= \sum_d \frac{(\mu_d\ \psi_d)}{\omega_d^2}$$

and

$$< S_{\psi^2}, \phi_3 >= -\sum_d \frac{(\psi_d)^2}{2\omega_d^2}$$

so that we have isolated the scalar product $< S(y, \psi), \phi(\theta) >$ through the definition of the three sufficient statistics S_y, S_ψ and S_{ψ^2}. In turn, these statistics are sufficient because we can write the population parameters as a function of them only:

$$\mu_d = \frac{1}{N} \sum_i \psi_{d\,i}$$

$$\omega_d^2 = \sqrt{\frac{1}{N} \sum_i (\psi_{d\,i})^2}$$

$$\sigma^2 = \sqrt{\frac{1}{n_{obs}} \sum_{ij} (y_{ij} - m(t_{ij}, \psi_i))^2}$$

Appendix A.2. Joint model with linear mixed-effects model and single event

We consider the joint model defined as:

$$\begin{aligned} y_{ij} &= m(t_{ij}, \psi_i) + \sigma\epsilon_{ij} \\ &= \psi_{0i} + \psi_{1i}\, t_{ij} + \sigma\epsilon_{ij} \\ h_i(t, \psi_i) &= h_0 \, \exp(\alpha \, m(t, \psi_i)) \end{aligned} \tag{A.3}$$

For longitudinal parameters, we assume same distributions as in equation A.1. $h_i(t, \psi)$ is the instantaneous risk of event for individual i at time t with h_0 the baseline risk and α the link coefficient. We introduce $S_i(t, \psi_i) = \exp\left(-\int_0^t h_i(s, \psi_i)\, \mathrm{d}s\right)$.

In the first K_1 iterations of the algorithm, where artificial variability is added on parameters h_0 and α, the vectors μ and η contains 2 additional parameters referring to h_0 and α. Hence, $\mu = (\mu_0, \mu_1, \mu_{h_0}, \mu_\alpha)$ and $\eta = (\eta_{0\,i}, \eta_{1\,i}, \eta_{h_0\,i}, \eta_{\alpha\,i})$. In that case, the joint likelihood of the complete data (y, T, δ, ψ) can be written as:

$$\begin{aligned} l(y_i, T_i, \delta_i, \psi_i) &= \prod_{ij} p(y_{ij}|\psi_i; \theta) \prod_i p(T_i, \delta_i|\psi_i; \theta) \prod_i p(\psi_i; \theta) \\ &= \prod_{ij} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_{ij} - m(t_{ij}, \psi_i))^2}{2\sigma^2}\right) \prod_i h_i(T_i, \psi_i)^{\delta_i} S_i(T_i, \psi_i) \prod_i \prod_d \frac{1}{(2\pi)^{1/2}\omega_d} \exp\left(-\frac{(\psi_{d\,i} - \mu_d)^2}{2\omega_d^2}\right) \end{aligned}$$

and the log-likelihood as:

$$\begin{aligned} \ln l(y, \psi; \theta) = &\sum_{i,j} -\ln\sqrt{2\pi}\sigma - \sum_{i,j} \frac{(y_{ij} - m(t_{ij}, \psi_i))^2}{2\,\sigma^2} \\ &+\delta_i \left(\log(\psi_{i\,h_0}) + \psi_{i\,\alpha}\, m(T_i, \psi_i)\right) - \frac{\psi_{i\,h_0}}{\psi_{i\,\alpha}\,\psi_{i\,1}} \left(\exp\left(\psi_{i\,\alpha}\, m(T_i, \psi_i)\right) - \exp\left(\psi_{i\,\alpha}\,\psi_{i\,0}\right)\right) \\ &+ N \sum_d -\ln\left((2\pi)^{D/2}\omega_d\right) - \sum_i \sum_d \frac{(\psi_{d\,i})^2}{2\omega_d^2} + \sum_i \sum_d \frac{(\mu_d\,\psi_{d\,i})}{\omega_d^2} - N \sum_d \frac{(\mu_d)^2}{2\omega_d^2} \end{aligned}$$

The log-likelihood again decomposes in the general form:

$$\log l(y, \psi; \theta) = -\Psi(\theta) + < S(y, \psi), \phi(\theta) > + A(T, \delta, \psi) \tag{A.4}$$

$A(T, \delta, \psi)$ depends only the complete survival data (T, δ, ψ) and is ignored in the computation of sufficient statistics, while the red and blue terms are the same as in equation A.2 and factorise similarly.

After the K_1 first iterations, the variability on h_0 and α is set to 0. The vector μ and η are now only composed of longitudinal parameters and are derived through the sufficient statistics. An optimization algorithm is used for h_0 and α. Note, This approach is also used to obtain estimates of the residual error parameters when using a combined error model for the continuous outcome, in which case the complete log-likelihood is strictly speaking no longer in the curved exponential family.

Appendix B. Code for dataset simulation

The following code is used to simulate the datasets in R. It does not require the *saemix* code or its extensions to run.

```
############### SIMULATION OF DATASETS #################
##### 6 models: LMEM      NLMEM      LMEM/NLMEM - TTE      LMEM/NLMEM - CR


##########    LMEM    ##########

set.seed(1996)
N = 100

longi = data.frame(id=NA,obs=NA,time=NA)

# Values used for simulation
b0=4
b1 = 0.2
omega_b0 = sqrt(4)
omega_b1 = sqrt(0.1)
sigma_a = 1

for (i in 1:100){
  r1 = rnorm(N,mean=b0 ,sd=omega_b0) # vector of random intercepts
  r2 = rnorm(N,mean=b1 ,sd=omega_b1) # vector of random slopes
  tt = 0:30 # design for longitudinal observations
  longi = data.frame(id = as.vector(sapply(1:N,function(x) rep(x,length(tt)))),
                     time = rep(tt,N))
  longi$obs = r1[longi$id] + r2[longi$id]*longi$time +
              rnorm(nrow(longi),sd=sigma_a)

  write.table(longi,paste0("../data",i,".txt"),row.names = F)
}


##########    NLMEM    ##########

set.seed(1996)
N = 100
longi = data.frame(id=NA,obs=NA,time=NA)

b0<-4
b1<- 0.15
b2 <- 0.2
```

```r
a = 10
omega_b0<-sqrt(4)
omega_b1<-sqrt(0.1)
omega_b2<-sqrt(0.1)
omega_a<-sqrt(0.5)
sigma_a<-1

for (i in 1:100){
  r1 = rnorm(N,mean=b0 ,sd=omega_b0)
  r2 = b1*exp(rnorm(N,mean=0 ,sd=omega_b1))
  r3 = b2*exp(rnorm(N,mean=0, sd=omega_b2))
  r4 = a*exp(rnorm(N,mean=0, sd=omega_a))
  tt = 0:30
  longi = data.frame(id = as.vector(sapply(1:N,function(x) rep(x,length(tt)))),
                     time = rep(tt,N))
  longi$obs = r1[longi$id] + r4[longi$id] *
              (exp(-r2[longi$id]*longi$time)-exp(-r3[longi$id]*longi$time)) +
              rnorm(nrow(longi),sd=sigma_a)

  write.table(longi,paste0("../data",i,".txt"),row.names = F)
}



########## LMEM - TTE ##########

set.seed(1996)
N=100

longi = data.frame(id=NA,obs=NA,time=NA)

b0=4
b1 = 0.2
omega_b0 = sqrt(4)
omega_b1 = sqrt(0.1)
sigma_a = 1
h0=0.005
alpha=0.2

for (i in 1:100){
  r1 = rnorm(N,mean=b0 ,sd=omega_b0)
  r2 = rnorm(N,mean=b1 ,sd=omega_b1)
  tt = 0:30
```

```
  longi = data.frame(id = as.vector(sapply(1:N,function(x) rep(x,length(tt)))),
                     time = rep(tt,N))
  longi$obs = r1[longi$id] + r2[longi$id]*longi$time +
              rnorm(nrow(longi),sd=sigma_a)

  # function that gives an event time given a random intercept r1 and a random
  # slope r2
  gettimes = function(r1,r2){
    x = rexp(1)
    H = function(tt) h0*exp(alpha*r1)/alpha/r2*(exp(alpha*r2*tt)-1) - x
    z = try(uniroot(H,interval=c(0,30))$root,silent=T)
    ifelse(class(z)=="try-error",Inf,z)
  }
  #survival data with a censoring time = D30
  tte.data = data.frame(id=1:N,time = mapply(gettimes,rint,reff), obs = 1)
  tte.data$obs[tte.data$time>30]=0
  tte.data$time[tte.data$time>30]=30
  # removal of post-event longitudinal data
  longi = longi[longi$time<=sapply(longi$id,
                                   function(x) tte.data$time[tte.data$id==x]),]

  longi$ytype=1
  tte.data$ytype=2

  alldata_saem = rbind(longi,tte.data)

  write.table(alldata_saem,paste0(".../data",i,".txt"),row.names = F)
}


##########   NLMEM - TTE   ##########

set.seed(1996)
N=100

longi = data.frame(id=NA,obs=NA,time=NA)

b0<-4
b1<- 0.15
b2 <- 0.2
a = 10
omega_b0<-sqrt(4)
omega_b1<-sqrt(0.1)
```

```
omega_b2<-sqrt(0.1)
omega_a<-sqrt(0.5)
sigma_a<-1
h0=0.005
alpha=0.3

for (i in 1:100){
  r1 = rnorm(N,mean=b0 ,sd=omega_b0)
  r2 = b1*exp(rnorm(N,mean=0, sd=omega_b1))
  r3 = b2*exp(rnorm(N,mean=0, sd=omega_b2))
  r4 = a*exp(rnorm(N,mean=0, sd=omega_a))
  tt = 0:30
  longi = data.frame(id = as.vector(sapply(1:N,function(x) rep(x,length(tt)))),
                     time = rep(tt,N))
  longi$obs = r1[longi$id] + r4[longi$id] *
              (exp(-r2[longi$id]*(longi$time))-exp(-r3[longi$id]*(longi$time)))+
              rnorm(nrow(longi),sd=sigma_a)

  gettimes = function(r1,r2,r3,r4){
    x = rexp(1)
    t = seq(0,100,length.out=1000)
    pas = t[2]-t[1]
    h = h0*exp(alpha*(r1+r4*(exp(-r2*(t))-exp(-r3*(t)))))
    H = cumsum(h)*pas
    z=(which(H>x)[1])*pas
    z=ifelse(is.na(z)==T,Inf,z)
  }
  tte.data = data.frame(id=1:N,time = mapply(gettimes,r1,r2,r3,r4), obs = 1)
  tte.data$obs[tte.data$time>30]=0
  tte.data$time[tte.data$time>30]=30

  longi = longi[longi$time<=sapply(longi$id,
                                   function(x) tte.data$time[tte.data$id==x]),]

  longi$ytype=1
  tte.data$ytype=2

  alldata_saem = rbind(longi,tte.data)

  write.table(alldata_saem,paste0(".../data",i,".txt"),row.names = F)
}
```

```r
##########    LMEM - CR   ##########

set.seed(1996)
N = 100

b0.pop<-4
b1.pop<- 0.2
omega_b0<-sqrt(4)
omega_b1<-sqrt(0.1)
sigma_a<-1
p1=0.15
g1 = 0.1
alpha1 = 0.2

for (i in 1:100){
  rint = rnorm(N,mean=b0.pop ,sd=omega_b0)
  reff = rnorm(N,mean=b1.pop ,sd=omega_b1)
  tt = 0:30
  longi = data.frame(id = as.vector(sapply(1:N,function(x) rep(x,length(tt)))),
                     time = rep(tt,N))
  longi$obs = rint[longi$id] + reff[longi$id]*longi$time +
              rnorm(nrow(longi),sd=sigma_a)

  gettimes = function(r1,r2){
    t = seq(0,1000,length.out=10000)
    pas = t[2]-t[1]
    h = (p1*g1*exp(-g1*t)/(1-p1*(1-exp(-g1*t))))*exp(alpha1*(r1+r2*t))
    F1 = 1-exp(-cumsum(h)*pas)
    F2 = (1-rev(F1)[1])*(1-exp(-t/15))

    p1 = rev(F1)[1]
    p2 = 1-p1

    u = runif(1)
    event = ifelse(u<p1,1,2)

    if(event==1) z=(which(F1/p1>runif(1))[1])*pas
    if(event==2) z=(which(F2/p2>runif(1))[1])*pas
    c(z,event)
  }

  a=mapply(gettimes,rint,reff)
```

```
  tte.data = data.frame(id=1:N,time = a[1,], obs = a[2,])
  tte.data$obs[tte.data$time>30]=0
  tte.data$time[tte.data$time>30]=30

  longi = longi[longi$time<=sapply(longi$id,
                                   function(x) tte.data$time[tte.data$id==x]),]
  longi$ytype=1
  tte.data$ytype=2

  alldata_saem = rbind(longi,tte.data)

  write.table(alldata_saem,paste0("../data",i,".txt"),row.names = F)

}


##########   NLMEM - CR  ##########

set.seed(1996)
N=100

longi = data.frame(id=NA,obs=NA,time=NA)
b0<-4
b1<- 0.15
b2 <- 0.2
a = 10
omega_b0<-sqrt(4)
omega_b1<-sqrt(0.1)
omega_b2<-sqrt(0.1)
omega_a<-sqrt(0.5)
sigma_a<-1
p1=0.15
g1 = 0.1
alpha1 = 0.3

for (i in 1:100){
  r1 = rnorm(N,mean=b0 ,sd=omega_b0)
  r2 = b1*exp(rnorm(N,mean=0 ,sd=omega_b1))
  r3 = b2*exp(rnorm(N,mean=0, sd=omega_b2))
  r4 = a*exp(rnorm(N,mean=0, sd=omega_a))

  tt = 0:30
  longi = data.frame(id = as.vector(sapply(1:N,function(x) rep(x,length(tt)))),
```

```
                    time = rep(tt,N))
  longi$obs = r1[longi$id] + r4[longi$id] *
              (exp(-r2[longi$id]*(longi$time))-exp(-r3[longi$id]*(longi$time)))+
              rnorm(nrow(longi),sd=sigma_a)

  gettimes = function(r1,r2,r3,r4){
    t = seq(0,1000,length.out=10000)
    pas = t[2]-t[1]
    h = (p1*g1*exp(-g1*t)/(1-p1*(1-exp(-g1*t))))*
        exp(alpha1*(r1+r4*(exp(-r2*(t))-exp(-r3*(t)))))
    F1 = 1-exp(-cumsum(h)*pas)
    F2 = (1-rev(F1)[1])*(1-exp(-t/15))

    p1 = rev(F1)[1]
    p2 = 1-p1

    u = runif(1)
    event = ifelse(u<p1,1,2)

    if(event==1) z=(which(F1/p1>runif(1))[1])*pas
    if(event==2) z=(which(F2/p2>runif(1))[1])*pas
    c(z,event)
  }
  aa=mapply(gettimes,r1,r2,r3,r4)

  tte.data = data.frame(id=1:N,time = aa[1,], obs = aa[2,])
  tte.data$obs[tte.data$time>30]=0
  tte.data$time[tte.data$time>30]=30
  longi = longi[longi$time<=sapply(longi$id,
                                   function(x) tte.data$time[tte.data$id==x]),]

  longi$ytype=1
  tte.data$ytype=2

  alldata_saem = rbind(longi,tte.data)

  write.table(alldata_saem,paste0("../data",i,".txt"),row.names = F)
}
```

Appendix C. Alternative initial parameter values (using separate submodel fits)

Appendix C.1. Choosing initial parameter values

For each of the joint models, we fitted separately the longitudinal submodel of the first simulated data set (initializing fixed effects to 1 and their variances to 100% of their values.) We also fitted Kaplan-Meier or cumulative incidence functions to choose plausible initial estimates for survival submodel parameters. For a time-to-event model with a single event, h_0 represents the inverse of the time constant. For competing risks models, p_1 represents the asymptotic proportion of event 1, g_1 the inverse of the time constant for event 1, and b the time constant for event 2.

	JM LMEM-TTE	JM NLMEM-TTE	JM LMEM-CR	JM NLMEM-CR
Fixed effects				
μ_0	4	4	4	4
μ_1	0.20	0.10	0.30	0.20
μ_2	-	0.20	-	0.20
μ_a	-	7	-	6
Random effects				
ω_0^2	4	4	4	4
ω_1^2	0.10	0.10	0.09	0.30
ω_2^2	-	0.10	-	0.30
ω_a^2	-	0.50	-	0.20
Error model				
σ	1	1	1	1
Survival model				
h_0	0.025	0.05	-	-
p_1	-	-	0.50	0.40
g_1	-	-	0.10	0.10
α	0	0	0	0
b	-	-	10	15

Table C.5: Initial parameter values for the joint models under the alternative setting

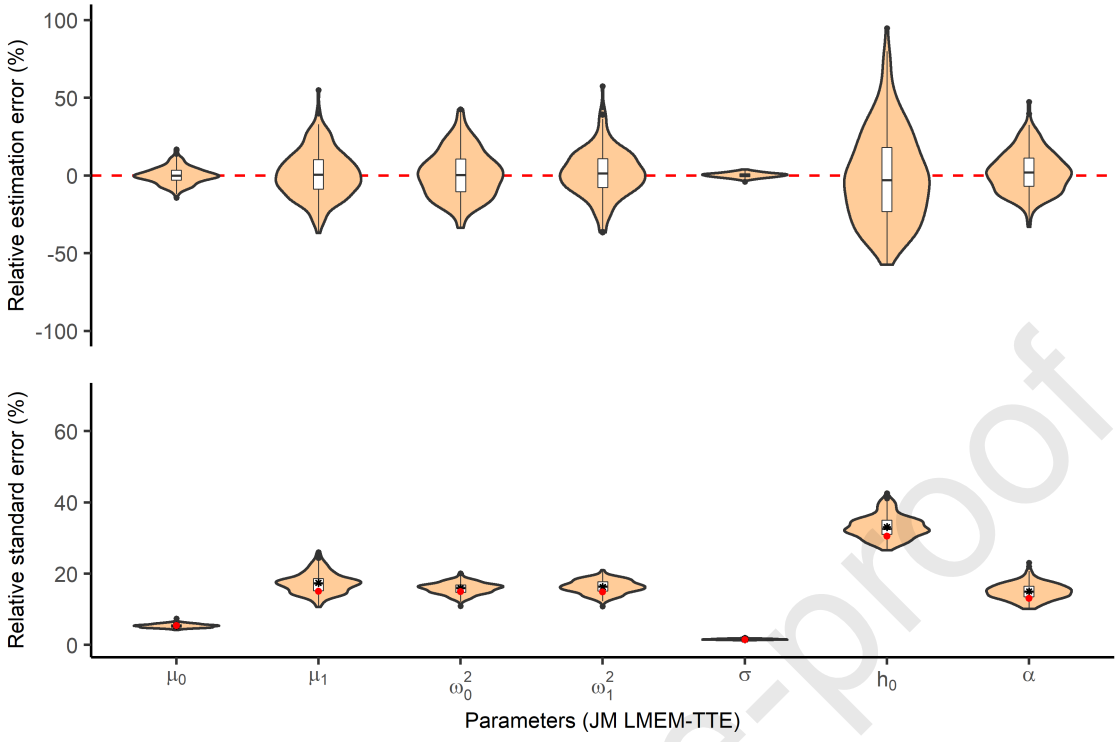
Appendix C.2. Relative Bias (RB) and Relative Root Mean Square Error (RRMSE) for parameters of the four joint models with alternative initial parameter estimates

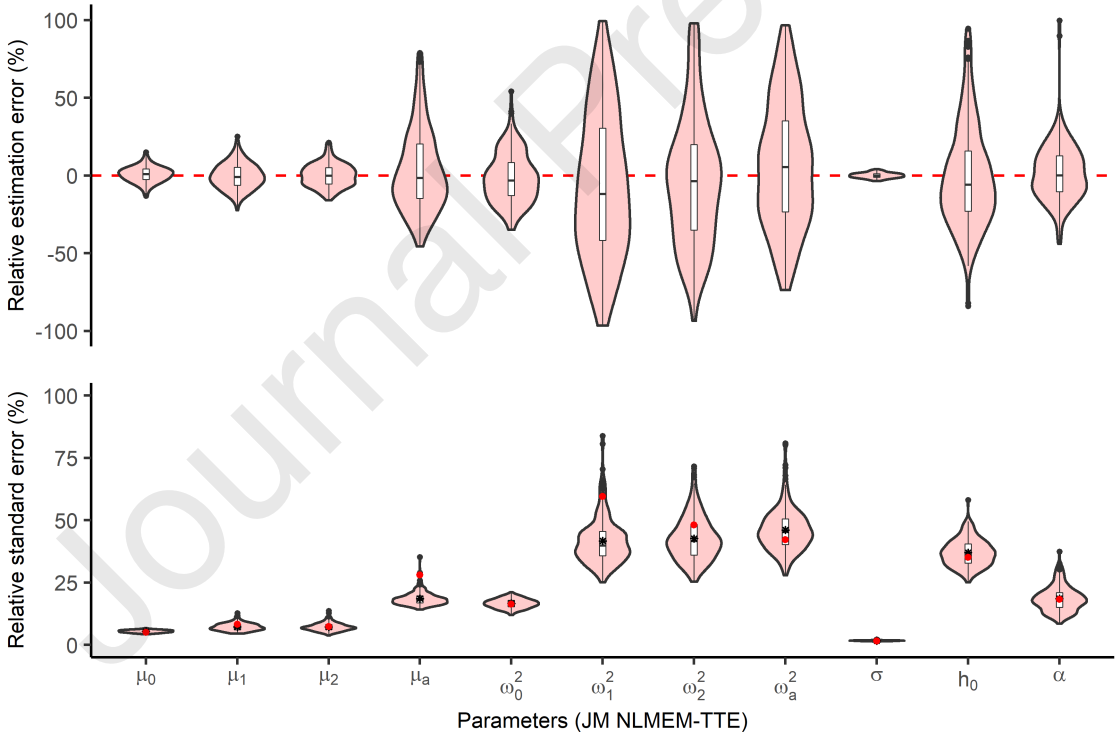
	JM LMEM-TTE		**JM NLMEM-TTE**		**JM LMEM-CR**		**JM NLMEM-CR**	
	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)
Longitudinal model								
Fixed effects								
μ_0	0.10	5.4	0.70	5.1	-0.25	5.6	-0.83	5.4
μ_1	0.94	15.2	-0.44	8.3	-0.33	18.8	0.10	10.9
μ_2	-	-	0.19	7.4	-	-	0.61	10.3
μ_a	-	-	4.8	29.7	-	-	16.2	50.1
Random effects								
ω_0^2	0.95	15.1	-1.7	16.1	-2.2	15.1	-0.48	15.7
ω_1^2	1.5	15.1	2.6	61.0	-3.1	18.0	5.5	89.3
ω_2^2	-	-	1.9	48.9	-	-	2.6	86.1
ω_a^2	-	-	12.2	48.8	-	-	4.3	66.3
Error model								
σ	0.24	1.5	0.04	1.6	0.16	2.2	0.07	2.5
Survival model								
h_0	-0.20	30.4	0.73	35.4	-	-	-	-
p_1	-	-	-	-	-4.1	35.9	-1.8	32.7
g_1	-	-	-	-	1.8	18.6	-2.0	22.3
α	2.4	13.5	1.8	18.6	5.7	21.4	4.1	20.8
b	-	-	-	-	1.2	20.0	-1.3	16.2

Table C.6: Relative Bias (RB) and Relative Root Mean Square Error (RRMSE) obtained on 200 simulations, for the four joint models when using the initial estimates from table C.5

The coverage rates of all the link coefficients and their 95% confidence intervals were 0.975 [0.943,0.992], 0.955 [0.916,0.979], 0.965 [0.929,0.986] and 0.925 [0.879,0.957] for JM-LMEM-TTE, JM-NLMEM-TTE, JM-LMEM-CR and JM-NLMEM-CR respectively.

Appendix C.3. Distribution of relative estimation errors and RSE





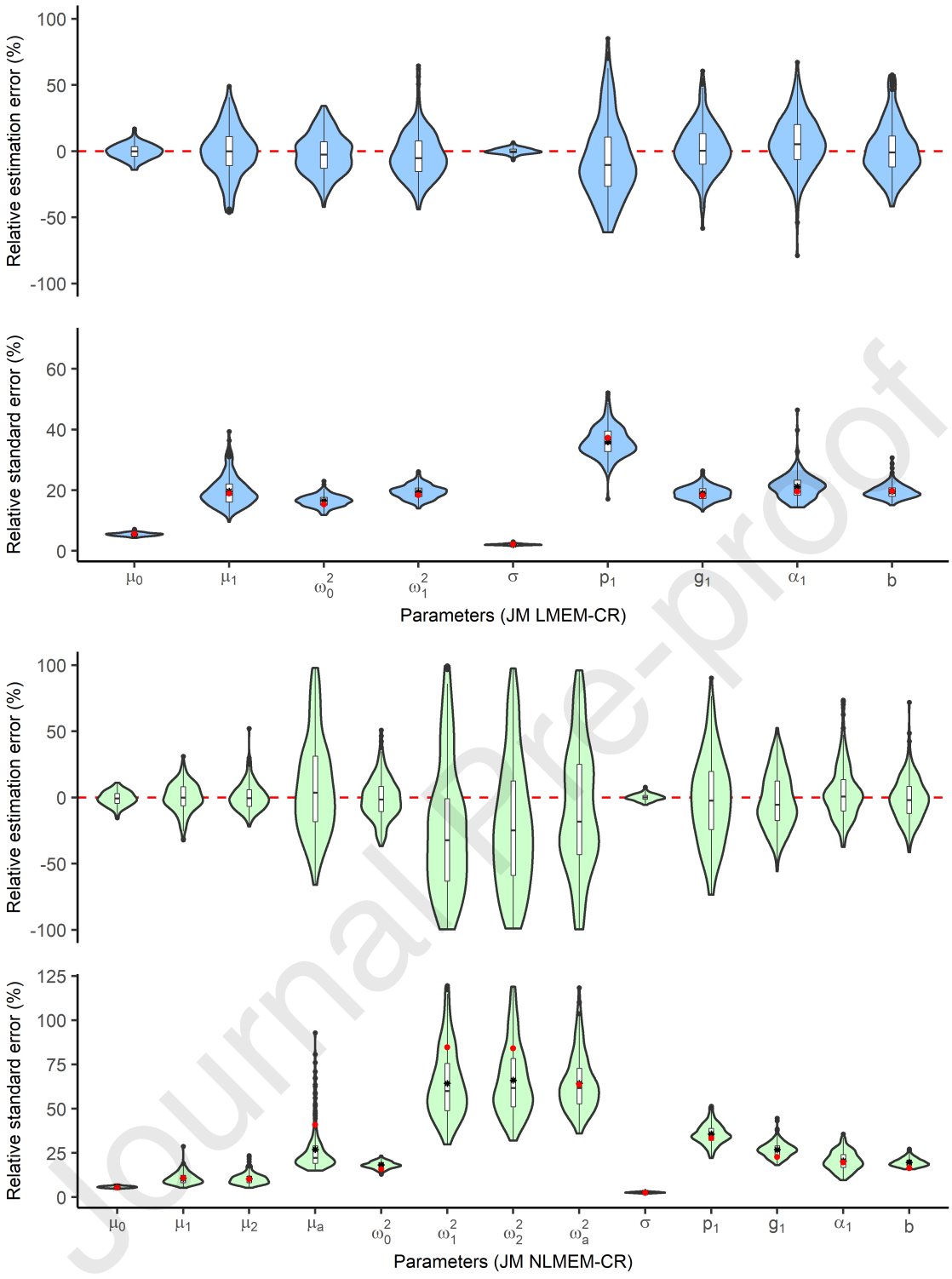


Figure C.6: Distribution of the relative estimation errors (top) and RSE (bottom) for all the joint models under the alternative scenario of initial parameters. Stars correspond to the mean of the RSE distribution. Red points correspond to the empirical RSE obtained over the 200 simulations.

Appendix D. R Markdown document for prothrombin example

This Rmarkdown document helps user to buid and fit joint models using the code extension available on Github. This document relies on the examples presented in the article.

Prothrombin example

We consider the data available in package JM: the prothro data set. 488 liver cirrhosis patients are followed at most 12 days with prothrombin measurements. Status (dead/censored) at the end of the follow-up is available for each patient. The objective is to assess the link between the individual prothrombin evolution with the risk of death.

```
library(JM)
```

```
## Warning: package 'JM' was built under R version 4.0.5
```

```
## Loading required package: MASS
```

```
## Loading required package: nlme
```

```
## Warning: package 'nlme' was built under R version 4.0.5
```

```
## Loading required package: splines
```

```
## Loading required package: survival
```

```
## Warning: package 'survival' was built under R version 4.0.5
```

```
library(pracma)
```

```
## Warning: package 'pracma' was built under R version 4.0.5
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(Cairo)
library(viridis)
```

```
## Warning: package 'viridis' was built under R version 4.0.5
```

```
## Loading required package: viridisLite
```

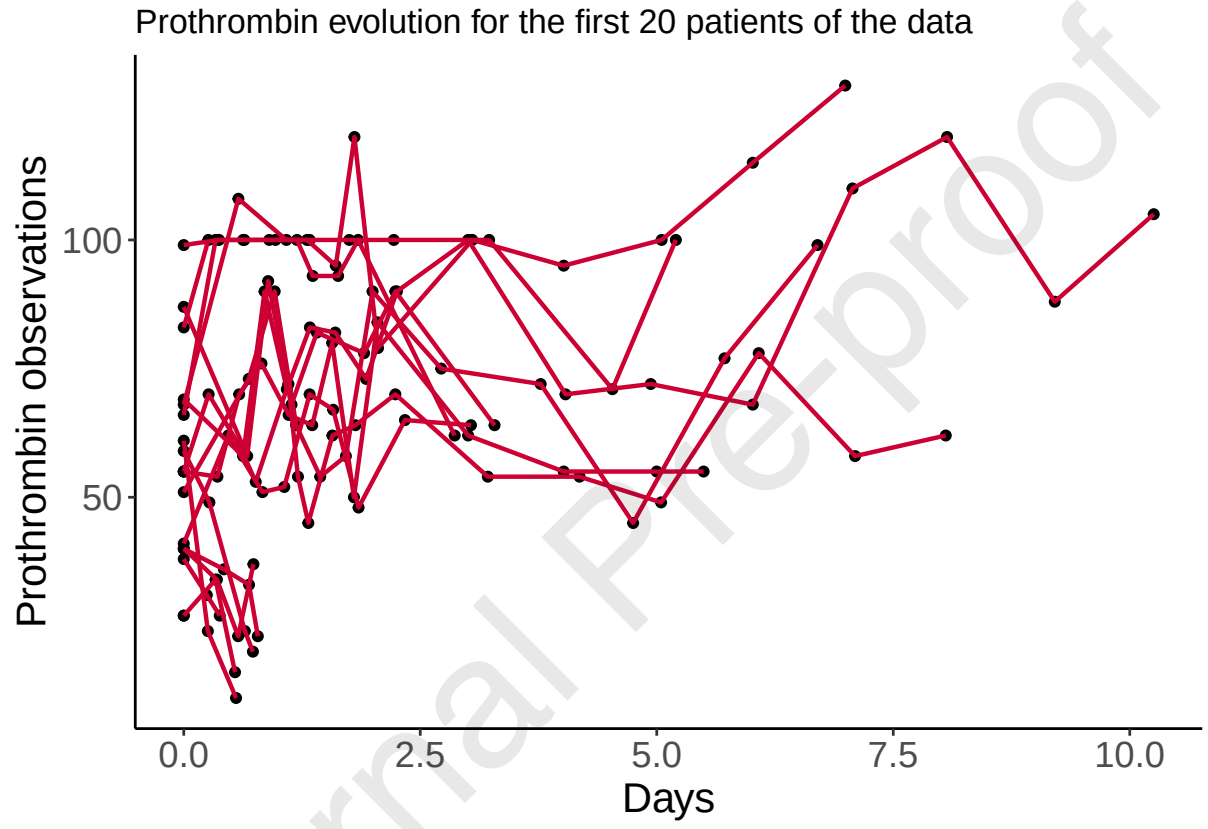
```
## Warning: package 'viridisLite' was built under R version 4.0.5
```

```
library(rlang)
```

```
## Warning: package 'rlang' was built under R version 4.0.5
```

```
data("prothro")
data("prothros")

gp = ggplot(data=prothro[which(prothro$id %in% 1:20),], aes(x=time, y=pro, group = id))+
  geom_point(lwd=1.5)+geom_line(col="#CC0033",lwd=0.8)+theme_classic()+
  ylab("Prothrombin observations")+xlab("Days")+
  theme(axis.text = element_text(size=14),axis.title = element_text(size=16))+
  ggtitle(label = "Prothrombin evolution for the first 20 patients of the data")
gp
```




```
table(prothros$death)
```

```
##
##   0   1
## 196 292
```

Joint model fit using new saemix.multi() function

The extended code uses the same main functions as saemix package does. We therefore refer the user to the saemix documentation previously published for the detail of each function (see Comets et al. JSS, 2017). Briefly, the main function is the saemix.multi() function used to estimate the population parameters of the (joint) model. This function requires two mandatory arguments referring to (1) the model (saemixModel object) and the data (saemixData object). The third argument is optional and concerns the algorithm settings.

```
saemixDir <- "C:/Users/AlexandraLAVALLEY/Documents/GitHub/saemixextension"
workDir <- file.path(saemixDir, "joint")
progDir<-file.path(saemixDir, "R")
setwd(workDir)

source(file.path(progDir,"aaa_generics.R"))
```

Loading functions from Github

```
## Creating a new generic function for 'psi' in the global environment

## Creating a new generic function for 'eta' in the global environment
```

```
source(file.path(progDir,"SaemixData.R"))
source(file.path(progDir,"SaemixRes.R"))
source(file.path(progDir,"SaemixModel.R"))
source(file.path(progDir,"SaemixObject.R"))
source(file.path(progDir,"func_plots.R"))

source(file.path(workDir,"multi_aux.R"))
source(file.path(workDir,"multi_initializeMainAlgo.R"))
source(file.path(workDir,"multi_estep.R"))
source(file.path(workDir,"multi_mstep.R"))
source(file.path(workDir,"multi_main.R"))
source(file.path(workDir,"multi_map.R"))
source(file.path(workDir,"compute_LL_multi.R"))
```

Formatting data The function saemixData() requires a mandatory argument which is the name of the dataset. The dataset has to be formatted in order to obtain an id column corresponding to the patient id, a time column corresponding to the sampling times, an observation column corresponding to the response observations and a ytype column corresponding to the distinct response types. Optional columns can be added if user the wants to model covariates for example. See saemix documentation for more details. In this example, prothrombin measurements correspond to response 1 (ytype = 1) and survival measurements correspond to response 2 (ytype = 2). For the survival reponse (ytype = 2), observation is 1 in case of event (death) and 0 otherwise.

```
d1 = prothro[,c(1,2,3)]
d1$ytype=1
colnames(d1)[2] = "obs"
d2 = prothros[,c(1,3,2)]
d2$ytype = 2
colnames(d2)[2] = "obs"
colnames(d2)[3] = "time"
data_joint = rbind(d1,d2)
# To see the data for patient 1
data_joint[data_joint$id==1,]
```

```
##       id obs      time ytype
## 1      1  38 0.0000000     1
```

```
## 2      1  31 0.2436754     1
## 3      1  27 0.3805717     1
## 11000  1   1 0.4134268     2
```

The user is encouraged to specify optional arguments of the saemixData() function: the id variable (name.group argument), the predictor variables (name.predictors argument) with at least the sampling times, the observation variable (name.reponse argument) and the response type variable (name.ytype argument).

```
saemix.data<-saemixData(name.data=data_joint, name.group=c("id"),
                        name.predictors=c("time","obs"),
                        name.response="obs",name.ytype = "ytype")
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset data_joint
##     Structured data: obs ~ time + obs | id
##     X variable for graphs: time ()
```

Joint model with a linear mixed-effects model and a survival model with constant baseline hazard The saemixModel() function requires two mandatory arguments. The first one is a R function describing the joint model involving the structural model for longitudinal observations and the likelihood contribution for the survival observations. The second one is a matrix with a number of columns equal to the number of parameters, and one (when no covariates) or several row (when covariates enter the model) giving the initial estimates of fixed-effects. The user is encouraged to specify optional arguments of the saemixModel() function: the response type with the modeltype argument ("structural" for longitudinal observations and "likelihood" for survival ones), the distribution of each parameter with the transform.par argument (0 = normal, 1 = log-normal, 2 = probit and 3 = logit), the fixed or estimated parameters with the fixed.estim argument (0 = to be fixed to the initial estimate, 1 = to be estimated), if random effects are added with the covariance.model argument (square matrix of size equal to the number of parameters giving the variance-covariance matrix of the model), the initialization of random effect variances with the omega.init argument (square matrix of size equal to the number of parameters giving the initialization of the variance-covariance matrix of the model), the error model with the error.model argument (valid types are "constant", "proportional", "combined"). Futher arguments can be considered and found in the package description (see documentation).

In the following we start with a simple case: a joint model with a linear mixed-effects and a survival model involving constant baseline risk. The joint model writes:

$$\begin{aligned} y_{ij} =& m_l(t_{ij},\psi_i) + g[m_l(t_{ij},\psi_i),\sigma]\epsilon_{ij} \\ =& \psi_{i0} + \psi_{i1} \times t_{ij} + \sigma\epsilon_{ij} \\ h_i(t,\psi_i) =& h_0 \, \exp(\alpha \, m(t,\psi_i)) \end{aligned} \tag{1}$$

We then define the model to be entered in the function. This function must have 3 arguments named psi (assumed to be a matrix with the number of columns equal to the number of parameters in the model (excluding error parameters), so here 4 for μ_0, μ_1, h_0 and α), id (assumed to be a vector of indices matching observation number with subject index) and xidep (assumed to be a matrix with as many columns as

predictors + 1 for the type of response, so here 3 for time, observations and response type). The three arguments passed to the function will be generated automatically from the model and data object within the saemix code. The function must return a vector composed of predictions for longitudinal responses and likelihood contributions for the survival responses. The length of the vector is equal to the number of rows in the predictor xidep.

```
JMmodel<-function(psi,id,xidep) {
  ytype<-xidep$ytype  # type of response (1: continuous, 2: event)
  b0 <- psi[id,1]
  b1 <- psi[id,2]
  h0 <- psi[id,3]
  alpha <- psi[id,4]

  ypred <- b0+b1*xidep$time # predictions for the longitudinal part

  T<-xidep$time[ytype==2] # vector of times (survival response)
  Nj <- length(T)
  ev = xidep$obs[ytype==2] # vector of observations (survival response)
  cens<-which(ev==0)       # with censored ones
  ind <- which(ev==1)      # and event ones

  # Creating vectors of the same length of T to compute likelihood of the survival part
  #(so removing duplicates)
  b0b = b0[ytype==2] # to have vectors of the same length as T
  b1b = b1[ytype==2]
  h0b = h0[ytype==2]
  alphab = alpha[ytype==2]

  haz <- h0b*exp(alphab*(b0b+b1b*T)) # instantaneous hazard
  # cumulative hazard (explicit expression in that case)
  H <- (h0b/(alphab*b1b))*exp((b0b+b1b*T)*alphab)-(h0b/(alphab*b1b))*exp(alphab*b0b)

  logpdf <- rep(0,Nj)
  logpdf[cens] <- -H[cens] # likelihood contributions for censored observations
  logpdf[ind] <- -H[ind] + log(haz[ind]) # likelihood contributions for event observations

  ypred[ytype==2] = logpdf
  return(ypred)
}
```

```
#### initializing parameters

param<-c(73,1.25,0.6,0.0001)
omega.sim<-c(18, 3, 0.05, 0.01)
sigma.sim <- 17

### saemix Model

saemix.model<-saemixModel(model=JMmodel,description="JM LMEM-TTE constant baseline hazard",
                      modeltype=c("structural","likelihood"),
                      psi0=matrix(param,ncol=4,byrow=TRUE,
                                  dimnames=list(NULL, c("b0","b1","h0","alpha"))),
                      transform.par=c(0,0,1,0), covariance.model=diag(c(1,1,0,0)),
                      fixed.estim = c(1,1,1,1),error.model = "constant",
```

```
                        omega.init = diag(omega.sim))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  JM LMEM-TTE constant baseline hazard
##   Model type:  structural likelihood
## function(psi,id,xidep) {
##   ytype<-xidep$ytype  # type of response (1: continuous, 2: event)
##   b0 <- psi[id,1]
##   b1 <- psi[id,2]
##   h0 <- psi[id,3]
##   alpha <- psi[id,4]
##
##   ypred <- b0+b1*xidep$time # predictions for the longitudinal part
##
##   T<-xidep$time[ytype==2] # vector of times (survival response)
##   Nj <- length(T)
##   ev = xidep$obs[ytype==2] # vector of observations (survival response)
##   cens<-which(ev==0)       # with censored ones
##   ind <- which(ev==1)      # and event ones
##
##   # Creating vectors of the same length of T to compute likelihood of the survival part
##   #(so removing duplicates)
##   b0b = b0[ytype==2] # to have vectors of the same length as T
##   b1b = b1[ytype==2]
##   h0b = h0[ytype==2]
##   alphab = alpha[ytype==2]
##
##   haz <- h0b*exp(alphab*(b0b+b1b*T)) # instantaneous hazard
##   # cumulative hazard (explicit expression in that case)
##   H <- (h0b/(alphab*b1b))*exp((b0b+b1b*T)*alphab)-(h0b/(alphab*b1b))*exp(alphab*b0b)
##
##   logpdf <- rep(0,Nj)
##   logpdf[cens] <- -H[cens] # likelihood contributions for censored observations
##   logpdf[ind] <- -H[ind] + log(haz[ind]) # likelihood contributions for event observations
##
##   ypred[ytype==2] = logpdf
##   return(ypred)
## }
##   Nb of parameters: 4
##       parameter names:  b0 b1 h0 alpha
##       distribution:
##      Parameter Distribution Estimated
## [1,] b0        normal       Estimated
## [2,] b1        normal       Estimated
## [3,] h0        log-normal   Estimated
## [4,] alpha     normal       Estimated
##   Variance-covariance matrix:
##       b0 b1 h0 alpha
## b0     1  0  0     0
```

```
## b1     0  1  0     0
## h0     0  0  0     0
## alpha  0  0  0     0
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##              b0   b1  h0 alpha
## Pop.CondInit 73 1.25 0.6 1e-04
```

In the following we specify some algorithm settings. The option fim = T is specified to obtain standard errors of parameter estimates. ll.is is specified to obtain the loglikelihood at the MLE, the AIC and BIC. Graphs are not currently adapted so please specify save.graphs = F. We run the algorithm using the saemix.multi() function.

```
saemix.options<-saemixControl(seed=12345, map=T, fim=T, ll.is=TRUE, save.graphs = F)
# please, specify save.graphs=F (currently not extended)
yfit <- saemix.multi(saemix.model, saemix.data, saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset data_joint
##     Structured data: obs ~ time + obs | id
##     X variable for graphs: time ()
## Dataset characteristics:
##     number of subjects:     488
##     number of observations: 3456
##     average/min/max nb obs: 7.08  /  2  /  18
## First 10 lines of data:
##       id      time obs obs.1 mdv cens occ ytype
## 1      1 0.0000000  38    38   0    0   1     1
## 2      1 0.2436754  31    31   0    0   1     1
## 3      1 0.3805717  27    27   0    0   1     1
## 11000  1 0.4134268   1     1   0    0   1     2
## 4      2 0.0000000  51    51   0    0   1     1
## 5      2 0.6872194  73    73   0    0   1     1
## 6      2 0.9610119  90    90   0    0   1     1
## 7      2 1.1882598  64    64   0    0   1     1
## 8      2 1.4428869  54    54   0    0   1     1
## 9      2 1.7139415  58    58   0    0   1     1
## -----------------------------------
## ----          Model            ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  JM LMEM-TTE constant baseline hazard
##   Model type:  structural likelihood
## function(psi,id,xidep) {
##   ytype<-xidep$ytype  # type of response (1: continuous, 2: event)
##   b0 <- psi[id,1]
##   b1 <- psi[id,2]
```

```
##   h0 <- psi[id,3]
##   alpha <- psi[id,4]
##
##   ypred <- b0+b1*xidep$time # predictions for the longitudinal part
##
##   T<-xidep$time[ytype==2] # vector of times (survival response)
##   Nj <- length(T)
##   ev = xidep$obs[ytype==2] # vector of observations (survival response)
##   cens<-which(ev==0)       # with censored ones
##   ind <- which(ev==1)      # and event ones
##
##   # Creating vectors of the same length of T to compute likelihood of the survival part
##   #(so removing duplicates)
##   b0b = b0[ytype==2] # to have vectors of the same length as T
##   b1b = b1[ytype==2]
##   h0b = h0[ytype==2]
##   alphab = alpha[ytype==2]
##
##   haz <- h0b*exp(alphab*(b0b+b1b*T)) # instantaneous hazard
##   # cumulative hazard (explicit expression in that case)
##   H <- (h0b/(alphab*b1b))*exp((b0b+b1b*T)*alphab)-(h0b/(alphab*b1b))*exp(alphab*b0b)
##
##   logpdf <- rep(0,Nj)
##   logpdf[cens] <- -H[cens] # likelihood contributions for censored observations
##   logpdf[ind] <- -H[ind] + log(haz[ind]) # likelihood contributions for event observations
##
##   ypred[ytype==2] = logpdf
##   return(ypred)
## }
## <bytecode: 0x00000000246acf60>
##   Nb of parameters: 4
##       parameter names:  b0 b1 h0 alpha
##       distribution:
##      Parameter Distribution Estimated
## [1,] b0        normal       Estimated
## [2,] b1        normal       Estimated
## [3,] h0        log-normal   Estimated
## [4,] alpha     normal       Estimated
##   Variance-covariance matrix:
##       b0 b1 h0 alpha
## b0     1  0  0     0
## b1     0  1  0     0
## h0     0  0  0     0
## alpha  0  0  0     0
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##              b0   b1  h0 alpha
## Pop.CondInit 73 1.25 0.6 1e-04
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
```

```
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  1
##     Seed:  12345
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  TRUE
##         save the graphs to files:  FALSE
##         directory where results should be saved:  newdir
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## ----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate
## [1,] b0        73.338
## [2,] b1         0.570
## [3,] h0         3.094
## [4,] alpha     -0.039
## [5,] a.1       17.233
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate
## b0 omega2.b0 369
## b1 omega2.b1  16
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.b0 omega2.b1
## omega2.b0 1         0
## omega2.b1 0         1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 28050.73
##       AIC = 28064.73 28064.73
##       BIC = 28094.07 28094.07
## ----------------------------------------------------
```

We obtain the summary of the fit with the parameter estimates, and likelihood value at MLE (with AIC and BIC). The standard error estimates are obtained using the following script.

```
yfit@results@fim  # variance covariance matrix (inverse of the FIM)
```

```
##                 b0            b1            h0         alpha     omega2.b0
## [1,]  1.0004032819 -5.975768e-02  0.0232522122 -3.188424e-04   2.569130851
## [2,] -0.0597576780  1.087968e-01 -0.0035186342  7.009299e-05  -0.088465304
```

```
## [3,]  0.0232522122 -3.518634e-03  0.0441352091 -6.058584e-04  -0.371007774
## [4,] -0.0003188424  7.009299e-05 -0.0006058584  9.059682e-06   0.002326825
## [5,]  2.5691308512 -8.846530e-02 -0.3710077744  2.326825e-03 988.446907983
## [6,]  0.0110253172  1.692256e-01 -0.0105168727  4.847003e-05   2.419921193
## [7,] -0.0280793906 -4.166982e-03  0.0005563514 -2.434968e-05  -0.147787916
##          omega2.b1           a.1
## [1,]  1.102532e-02 -2.807939e-02
## [2,]  1.692256e-01 -4.166982e-03
## [3,] -1.051687e-02  5.563514e-04
## [4,]  4.847003e-05 -2.434968e-05
## [5,]  2.419921e+00 -1.477879e-01
## [6,]  3.970566e+00 -7.040594e-02
## [7,] -7.040594e-02  2.981143e-02
```

```
sqrt(diag(yfit@results@fim)) # standard errors of parameters estimates
```

```
## [1]  1.00020162  0.32984363  0.21008381  0.00300993 31.43957551  1.99262797
## [7]  0.17265985
```

```
# Formatting results in a data frame
d = data.frame(par=c(yfit@results@name.fixed,yfit@results@name.random,"sigma_a1"),
               est = c(yfit@results@fixed.effects,diag(yfit@results@omega)[1:2],
                       yfit@results@respar[c(1)]),
               se = sqrt(diag(yfit@results@fim)))
print(d)
```

```
##         par          est          se
## 1        b0  73.33846825  1.00020162
## 2        b1   0.57019224  0.32984363
## 3        h0   3.09438337  0.21008381
## 4     alpha  -0.03867724  0.00300993
## 5 omega2.b0 369.32691471 31.43957551
## 6 omega2.b1  15.75845002  1.99262797
## 7  sigma_a1  17.23307846  0.17265985
```

COVID-19 example (multivariate joint model with competing risks)

We consider the joint model defined in the article involving three biomarkers and two competing risks:

$$
\begin{aligned}
y_{ij1} &= m_1(t_{ij1}, \psi_{i1}) + \sigma_1\, m_1(t_{ij1}, \psi_{i1})\epsilon_{ij1} \\
y_{ij2} &= m_2(t_{ij2}, \psi_{i2}) + \sigma_2\epsilon_{ij2} \\
y_{ij3} &= m_3(t_{ij3}, \psi_{i3}) + \sigma_3\epsilon_{ij3}
\end{aligned} \tag{2}
$$

$$
h_1(t, \psi_i, Score_i; \theta) = h_0 \exp\left[\sum_{k=1}^{3}\Big(\alpha_k \times \big(m_k(t, \psi_{ik}) - med_k\big)\Big) + \beta \cdot Score_i\right]
$$

$$
h_2(t, \psi_i, Score_i; \theta) = \frac{1}{b}\frac{(1 - F_1(\infty))\exp(-t/b)}{1 - (1 - F_1(\infty))\,(1 - \exp(-t/b))}
$$

with:

$$
\begin{aligned}
m_1(t_{ij1}, \psi_{i1}) &= \psi_{i01} + \psi_{ia1} \times [\exp(\psi_{i11}\, t_{ij1}) - \exp(\psi_{i21}\, t_{ij1})] \\
m_2(t_{ij2}, \psi_{i2}) &= \psi_{i02} + \psi_{i12} \times t_{ij2} \\
m_3(t_{ij3}, \psi_{i3}) &= \psi_{i03} + \psi_{i13} \times t_{ij3}
\end{aligned} \tag{3}
$$

```
data_joint = read.table("W:/saemix/dev/riscov/dataJM.txt", header = T)
dataJM<-saemixData(name.data=data_joint, name.group=c("id"), name.predictors=c("time","obs"),
                   name.response="obs", name.ytype = "ytype", name.covariates = c("score4C"))
```

Creating the saemix data object

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset data_joint
##     Structured data: obs ~ time + obs | id
##     X variable for graphs: time ()
##     covariates: score4C (-)
```

Creating the saemix model object The model can be implemented as follows.

```
JMmodel<-function(psi,id,xidep) {
    ytype<-xidep$ytype
    N = unique(id)

    b01 <- psi[id,1]
    b11 <- psi[id,2]
    b21 <- psi[id,3]
    a1 <- psi[id,4]
    b02 <- psi[id,5]
    b12 <- psi[id,6]
    b03 <- psi[id,7]
    b13 <- psi[id,8]
    h1 <- psi[id,9]
    alpha1 <- psi[id,10]
    alpha2 <- psi[id,11]
    alpha3 <- psi[id,12]
    b <- psi[id,13]
    cov <- psi[id,14]

    T2 = xidep[ytype==2,1]  # vector of times for biom 2
    T3 = xidep[ytype==3,1]  # vector of times for biom 3
    T<-xidep[ytype==4,1]    # vector of times partie survie
    ev = xidep$obs[ytype==4]
    Nj <- length(T)
    cens<-which(ev==0)  # index of censored observations
```

```
ind1 <- which(ev==1) # index of event 1
ind2 <- which(ev==2) # index of event 2

schem = sapply(N, function(i) length(which(id <= i)))
# schem is a vector giving the number of measurements for each patients
# defined for having unique individual parameters (see what follows)

b01b <- b01[schem]
b11b <- b11[schem]
b21b = b21[schem]
a1b = a1[schem]
b02b <- b02[schem]
b12b <- b12[schem]
b03b = b03[schem]
b13b = b13[schem]
h1b <- h1[schem]
alpha1b = alpha1[schem]
alpha2b = alpha2[schem]
alpha3b = alpha3[schem]
bb = b[schem]
covb = cov[schem]

f=function(x) seq(0,x,length.out=100)
tab = mapply(f,T)
tab = t(tab)
pas = tab[,2]-tab[,1]
# used for numerical integration because no explicit form of the cumulative hazard

f2=function(x) seq(0,x,length.out=1000)
tab2 = replicate(Nj,f2(1000))
tab2 = t(tab2)
pas2 = tab2[,2]-tab2[,1]
# used to compute the value F1(inf) (used in the competing event hazard)
# proxy: F1(inf) approximated by F1(1000)

haz1 = h1b*exp(alpha1b*(b01b+a1b*(exp(b11b*tab)-exp(b21b*tab))-5.78)
               +alpha2b*(b02b+b12b*tab-7.42)
               +alpha3b*(b03b+b13b*tab-2.89)+covb)
H1 = apply(haz1,1,sum)*pas # cumulative hazard from 0 to T
hazt1 = haz1[,100] # cumulative hazard for event 1 at time T

haz1b = h1b*exp(alpha1b*(b01b+a1b*(exp(b11b*tab2)-exp(b21b*tab2))-5.78)
                +alpha2b*(b02b+b12b*tab2-7.42)
                +alpha3b*(b03b+b13b*tab2-2.89)+covb)
H1b = apply(haz1b,1,sum)*pas2
F1 = 1-exp(-H1b) # F1(1000)

hazt2 = 1/bb*(1-F1)*exp(-T/bb)/(1-(1-F1)*(1-exp(-T/bb)))
H2 = -log(1-(1-F1)*(1-exp(-T/bb))) # cumulative hazard for event 2 at time T

logpdf <- rep(0,Nj)
logpdf[cens] <- log(exp(-H1[cens])+exp(-H2[cens])-1)
# likelihood contributions for censored observations
```

```
    logpdf[ind1] <- -H1[ind1] + log(hazt1[ind1])
    # likelihood contributions for event 1 observations
    logpdf[ind2] <- -H2[ind2] + log(hazt2[ind2])
    # likelihood contributions for event 2 observations

    ypred = rep(NA,length(xidep[,1]))

    ypred[ytype==1] = b01[ytype==1]+a1[ytype==1]*(exp(b11[ytype==1]*xidep[ytype==1,1])
                                                  -exp(b21[ytype==1]*xidep[ytype==1,1]))
    ypred[ytype==2] = b02[ytype==2]+b12[ytype==2]*xidep[ytype==2,1]
    ypred[ytype==3] = b03[ytype==3]+b13[ytype==3]*xidep[ytype==3,1]

    ypred[ytype==4] = logpdf

    return(ypred)
}

# Initial guesses for parameter estimates
param_longi<-c(4.5,-0.14,-0.16,6.1,7.4,0.0008,4.2,-0.15)
omega_longi <- c(4,0.01,0.003,0.7,0.001,0.00001,0.8,0.01)
param_surv<-c(0.0009,0.25,-9,0.76,16,0)
omega_surv <- c(0.01,0.01,0.01,0.01,0.01,0.01)
param_cov <-  c(0,0,0,0,0,0,0,0,0,0,0,0,0,0.3)
param <- matrix(data = c(param_longi, param_surv, param_cov), ncol = 14, byrow = T,
                dimnames=list(NULL, c( "b01", "b11", "b21", "a1","b02", "b12", "b03", "b13",
                                       "h1", "alpha1", "alpha2", "alpha3", "b", "cov")))
omega <- c(omega_longi, omega_surv)

jointTTE<-saemixModel(model=JMmodel,description="JM lin+competing risks",
                      modeltype=c("structural","structural","structural","likelihood"),
                      psi0=param,transform.par=c(0,0,0,1,0,0,0,0,1,0,0,0,1,0),
                      covariance.model=diag(c(1,1,1,1,1,1,1,1,0,0,0,0,0,0)),
                      omega.init = diag(omega), fixed.estim = c(1,1,1,1,1,1,1,1,1,1,1,1,1,0),
                      covariate.model = c(0,0,0,0,0,0,0,0,0,0,0,0,0,1),
                      error.model = c("proportional","constant","constant","likelihood"))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  JM lin+competing risks
##   Model type:  structural structural structural likelihood
## function(psi,id,xidep) {
##     ytype<-xidep$ytype
##     N = unique(id)
##
##     b01 <- psi[id,1]
##     b11 <- psi[id,2]
##     b21 <- psi[id,3]
##     a1 <- psi[id,4]
##     b02 <- psi[id,5]
##     b12 <- psi[id,6]
```

```
##     b03 <- psi[id,7]
##     b13 <- psi[id,8]
##     h1 <- psi[id,9]
##     alpha1 <- psi[id,10]
##     alpha2 <- psi[id,11]
##     alpha3 <- psi[id,12]
##     b <- psi[id,13]
##     cov <- psi[id,14]
##
##     T2 = xidep[ytype==2,1]  # vector of times for biom 2
##     T3 = xidep[ytype==3,1]  # vector of times for biom 3
##     T<-xidep[ytype==4,1]    # vector of times partie survie
##     ev = xidep$obs[ytype==4]
##     Nj <- length(T)
##     cens<-which(ev==0)  # index of censored observations
##     ind1 <- which(ev==1) # index of event 1
##     ind2 <- which(ev==2) # index of event 2
##
##     schem = sapply(N, function(i) length(which(id <= i)))
##     # schem is a vector giving the number of measurements for each patients
##     # defined for having unique individual parameters (see what follows)
##
##     b01b <- b01[schem]
##     b11b <- b11[schem]
##     b21b = b21[schem]
##     a1b = a1[schem]
##     b02b <- b02[schem]
##     b12b <- b12[schem]
##     b03b = b03[schem]
##     b13b = b13[schem]
##     h1b <- h1[schem]
##     alpha1b = alpha1[schem]
##     alpha2b = alpha2[schem]
##     alpha3b = alpha3[schem]
##     bb = b[schem]
##     covb = cov[schem]
##
##     f=function(x) seq(0,x,length.out=100)
##     tab = mapply(f,T)
##     tab = t(tab)
##     pas = tab[,2]-tab[,1]
##     # used for numerical integration because no explicit form of the cumulative hazard
##
##     f2=function(x) seq(0,x,length.out=1000)
##     tab2 = replicate(Nj,f2(1000))
##     tab2 = t(tab2)
##     pas2 = tab2[,2]-tab2[,1]
##     # used to compute the value F1(inf) (used in the competing event hazard)
##     # proxy: F1(inf) approximated by F1(1000)
##
##     haz1 = h1b*exp(alpha1b*(b01b+a1b*(exp(b11b*tab)-exp(b21b*tab))-5.78)
##                    +alpha2b*(b02b+b12b*tab-7.42)
##                    +alpha3b*(b03b+b13b*tab-2.89)+covb)
##     H1 = apply(haz1,1,sum)*pas # cumulative hazard from 0 to T
```

```
##     hazt1 = haz1[,100] # cumulative hazard for event 1 at time T
##
##     haz1b = h1b*exp(alpha1b*(b01b+a1b*(exp(b11b*tab2)-exp(b21b*tab2))-5.78)
##                     +alpha2b*(b02b+b12b*tab2-7.42)
##                     +alpha3b*(b03b+b13b*tab2-2.89)+covb)
##     H1b = apply(haz1b,1,sum)*pas2
##     F1 = 1-exp(-H1b) # F1(1000)
##
##     hazt2 = 1/bb*(1-F1)*exp(-T/bb)/(1-(1-F1)*(1-exp(-T/bb)))
##     H2 = -log(1-(1-F1)*(1-exp(-T/bb))) # cumulative hazard for event 2 at time T
##
##     logpdf <- rep(0,Nj)
##     logpdf[cens] <- log(exp(-H1[cens])+exp(-H2[cens])-1)
##     # likelihood contributions for censored observations
##     logpdf[ind1] <- -H1[ind1] + log(hazt1[ind1])
##     # likelihood contributions for event 1 observations
##     logpdf[ind2] <- -H2[ind2] + log(hazt2[ind2])
##     # likelihood contributions for event 2 observations
##
##     ypred = rep(NA,length(xidep[,1]))
##
##     ypred[ytype==1] = b01[ytype==1]+a1[ytype==1]*(exp(b11[ytype==1]*xidep[ytype==1,1])
##                                                   -exp(b21[ytype==1]*xidep[ytype==1,1]))
##     ypred[ytype==2] = b02[ytype==2]+b12[ytype==2]*xidep[ytype==2,1]
##     ypred[ytype==3] = b03[ytype==3]+b13[ytype==3]*xidep[ytype==3,1]
##
##     ypred[ytype==4] = logpdf
##
##     return(ypred)
## }
##   Nb of parameters: 14
##       parameter names:  b01 b11 b21 a1 b02 b12 b03 b13 h1 alpha1 alpha2 alpha3 b cov
##       distribution:
##       Parameter Distribution Estimated
##  [1,] b01       normal       Estimated
##  [2,] b11       normal       Estimated
##  [3,] b21       normal       Estimated
##  [4,] a1        log-normal   Estimated
##  [5,] b02       normal       Estimated
##  [6,] b12       normal       Estimated
##  [7,] b03       normal       Estimated
##  [8,] b13       normal       Estimated
##  [9,] h1        log-normal   Estimated
## [10,] alpha1    normal       Estimated
## [11,] alpha2    normal       Estimated
## [12,] alpha3    normal       Estimated
## [13,] b         log-normal   Estimated
## [14,] cov       normal       Fixed
##   Variance-covariance matrix:
##        b01 b11 b21 a1 b02 b12 b03 b13 h1 alpha1 alpha2 alpha3 b cov
## b01      1   0   0  0   0   0   0   0  0      0      0      0 0   0
## b11      0   1   0  0   0   0   0   0  0      0      0      0 0   0
## b21      0   0   1  0   0   0   0   0  0      0      0      0 0   0
## a1       0   0   0  1   0   0   0   0  0      0      0      0 0   0
```

```
## b02      0   0   0  0   1   0   0   0  0      0      0      0 0   0
## b12      0   0   0  0   0   1   0   0  0      0      0      0 0   0
## b03      0   0   0  0   0   0   1   0  0      0      0      0 0   0
## b13      0   0   0  0   0   0   0   1  0      0      0      0 0   0
## h1       0   0   0  0   0   0   0   0  0      0      0      0 0   0
## alpha1   0   0   0  0   0   0   0   0  0      0      0      0 0   0
## alpha2   0   0   0  0   0   0   0   0  0      0      0      0 0   0
## alpha3   0   0   0  0   0   0   0   0  0      0      0      0 0   0
## b        0   0   0  0   0   0   0   0  0      0      0      0 0   0
## cov      0   0   0  0   0   0   0   0  0      0      0      0 0   0
##   Error model: proportional , initial values: b.1=1 a.1=1 a.1=1
##   Error model: proportional , initial values: b.1=1 a.1=1 a.1=1
##   Error model: proportional , initial values: b.1=1 a.1=1 a.1=1
##   Covariate model:
##      b01 b11 b21 a1 b02 b12 b03 b13 h1 alpha1 alpha2 alpha3 b cov
## [1,]   0   0   0  0   0   0   0   0  0      0      0      0 0   1
##     Initial values
##              b01   b11   b21  a1 b02   b12 b03   b13    h1 alpha1 alpha2 alpha3
## Pop.CondInit 4.5 -0.14 -0.16 6.1 7.4 8e-04 4.2 -0.15 9e-04   0.25     -9   0.76
## Cov.CondInit 0.0  0.00  0.00 0.0 0.0 0e+00 0.0  0.00 0e+00   0.00      0   0.00
##               b cov
## Pop.CondInit 16 0.0
## Cov.CondInit  0 0.3
```

Appendix E. Simulation study results for LMEM and NLMEM (no joint models)

Appendix E.1. Relative bias and relative root mean square errors obtained on 200 simulations, for the LMEM and NLMEM

	LMEM		**NLMEM**	
	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)
Longitudinal model				
Fixed effects				
μ_0	-0.19	5.2	0.07	4.89
μ_1	1.06	17.2	-0.34	6.05
μ_2	-	-	-1.25	5.88
μ_a	-	-	9.56	25.2
Random effects				
ω_0^2	-0.86	14.2	-1.09	15.94
ω_1^2	-0.99	14.3	-9.86	37.3
ω_2^2	-	-	-1.80	38.1
ω_a^2	-	-	-13.8	36.8
Error model				
σ	0.07	1.40	0.11	1.33

Table E.7: Relative Bias (RB) and Relative Root Mean Square Error (RRMSE) obtained on 200 simulations, for the LMEM and NLMEM (when initial parameter estimates are the true values). Note: the number of chains of the algorithm was set to 3 for both models.

Appendix E.2. Distribution of relative estimation errors and RSE for LMEM and NLMEM

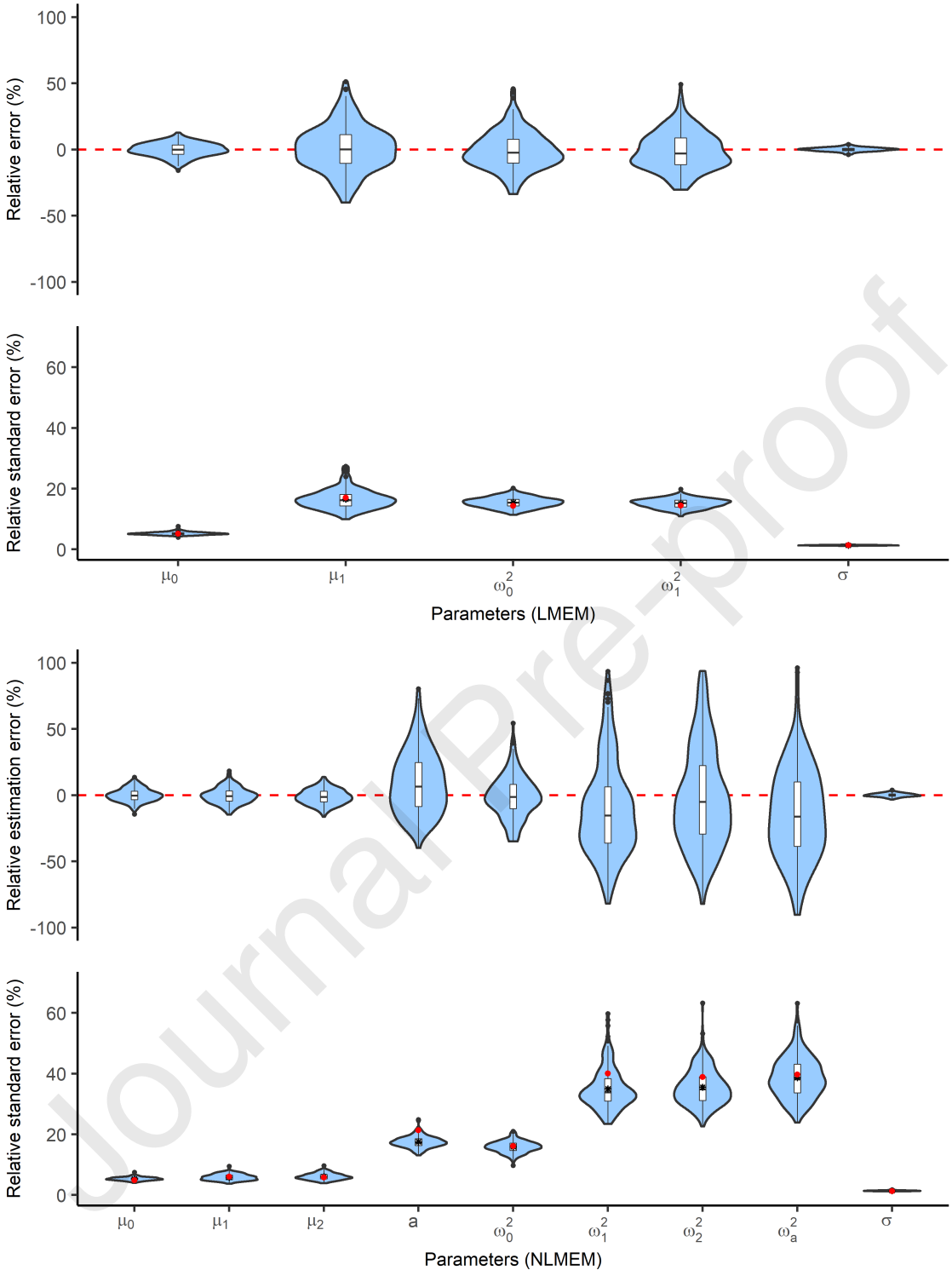


Figure E.7: Distribution of the relative estimation errors (top) and RSE (bottom) for LMEM and NLMEM, obtained on 200 simulations (when initial parameter estimates are the true values). Stars correspond to the mean distribution of the RSE. Red points correspond to the empirical RSE obtained over all the simulations. Note: the number of chains of the algorithm was set to 3 for both models.

Appendix F. Simulated longitudinal data for the joint models

Figure F.8 shows the data for the first simulated dataset in the four scenarios with a joint model.

Table F.8 gives the total and individual number of observations in the joint scenarios for these datasets. The scenarios with competing risks (JM-LMEM-CR, JM-NLMEM-CR) have around 50% fewer longitudinal observations in total than the scenarios with a single event (JM-LMEM-TTE, JM-NLMEM-TTE), but the median number of observations is decreased by 70%.

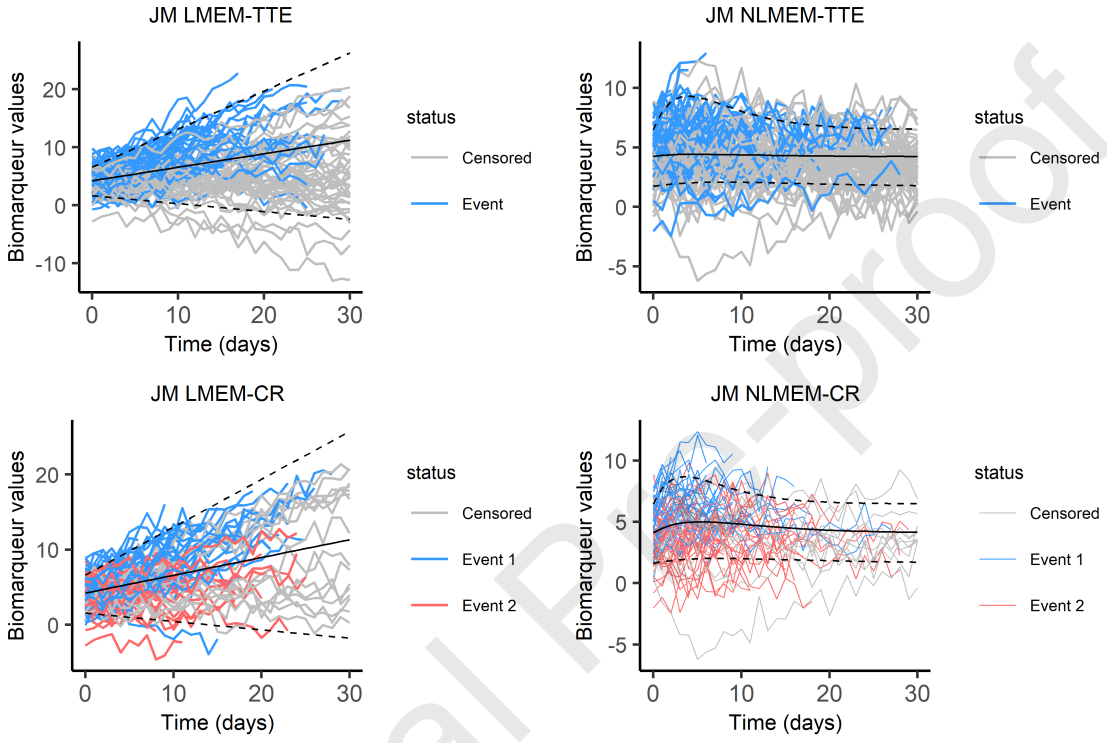


Figure F.8: Longitudinal data of the first data set simulated for each joint model (N=100 subjects). The solid black lines correspond to the median of the predicted curves for 1000 individual parameter vectors ψ_{ik} simulated within the population distribution. Dashed lines represent the corresponding 10th and 90th percentiles.

	JM LMEM-TTE	JM-NLMEM-TTE	JM LMEM-CR	JM-NLMEM-CR
Total number of longitudinal observations	2235	2209	1453	1204
Median [Q1-Q3] number of longitudinal observations per individual	26 [15-31]	31 [12-31]	13 [5-23]	9 [5-17]

Table F.8: Description of longitudinal observations in the first simulated data set of each of the four scenarios.

Appendix G. Calibration of the number of chains for the SAEM algorithm

	3 chains				10 chains			
	JM NLMEM-TTE		**JM NLMEM-CR**		**JM NLMEM-TTE**		**JM NLMEM-CR**	
	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)	RB (%)	RRMSE (%)
Longitudinal model								
Fixed effects								
μ_0	0.80	5.1	-0.74	5.4	0.70	5.1	-0.81	5.4
μ_1	1.6	8.5	1.3	11.8	-0.33	8.3	0.46	10.8
μ_2	-1.9	7.4	-0.92	10.7	0.10	7.4	0.13	10.0
μ_a	24.1	46.2	33.5	70.0	5.5	29.7	19.7	52.1
Random effects								
ω_0^2	-1.7	16.1	-0.76	15.6	-1.7	16.1	-0.49	15.7
ω_1^2	-19.6	56.6	-6.7	78.9	1.1	60.6	0.59	84.8
ω_2^2	-19.3	47.0	-0.69	87.5	0.90	50.0	-0.52	84.8
ω_a^2	4.5	48.6	-15.6	57.0	12.6	50.2	4.9	66.0
Error model								
σ	0.10	1.6	0.02	2.5	0.04	1.6	0.09	2.5
Survival model								
h_0	0.70	35.3	-	-	0.74	35.4	-	-
p_1	-	-	-1.7	32.7	-	-	-1.8	32.7
g_1	-	-	-2.0	22.3	-	-	-2.0	22.3
α	1.8	18.6	4.1	20.8	1.7	18.6	4.1	20.8
b	-	-	-1.3	16.2	-	-	-1.3	16.2

Table G.9: Relative Bias (RB) and Relative Root Mean Square Error (RRMSE) obtained on 200 simulations, for the joint models involving NLMEM, for different number of chains. In this table initial parameters were set to the true values.

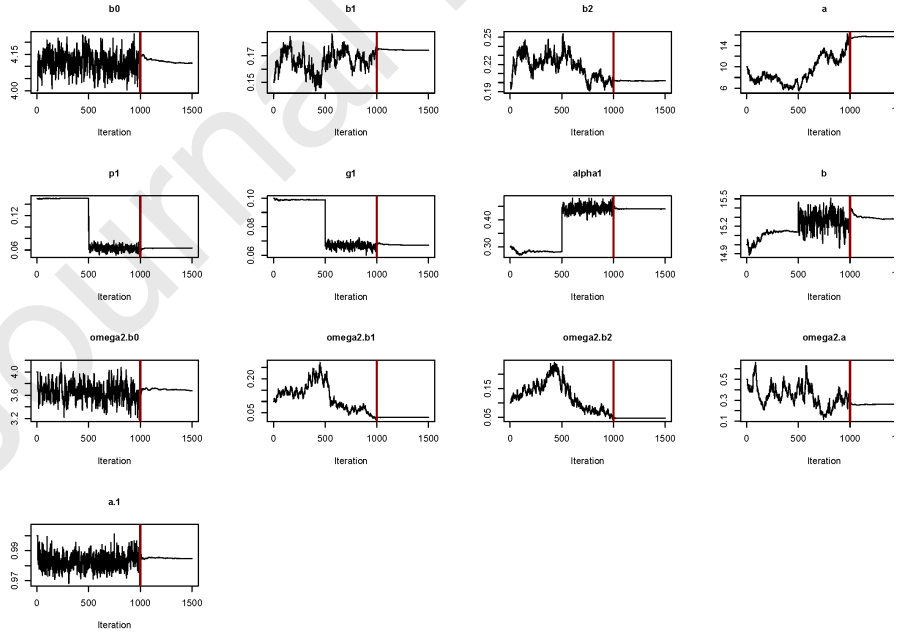


Figure G.9: Convergence graph showing the evolution of the parameter estimates across the iterations of the algorithm (when initial estimates were set to the true values, and the number of chains for the algorithm was set to 3). Results are shown for the first simulated dataset for JM-NLMEM-CR.

Appendix H. Comparison of convergence plots for different initial conditions

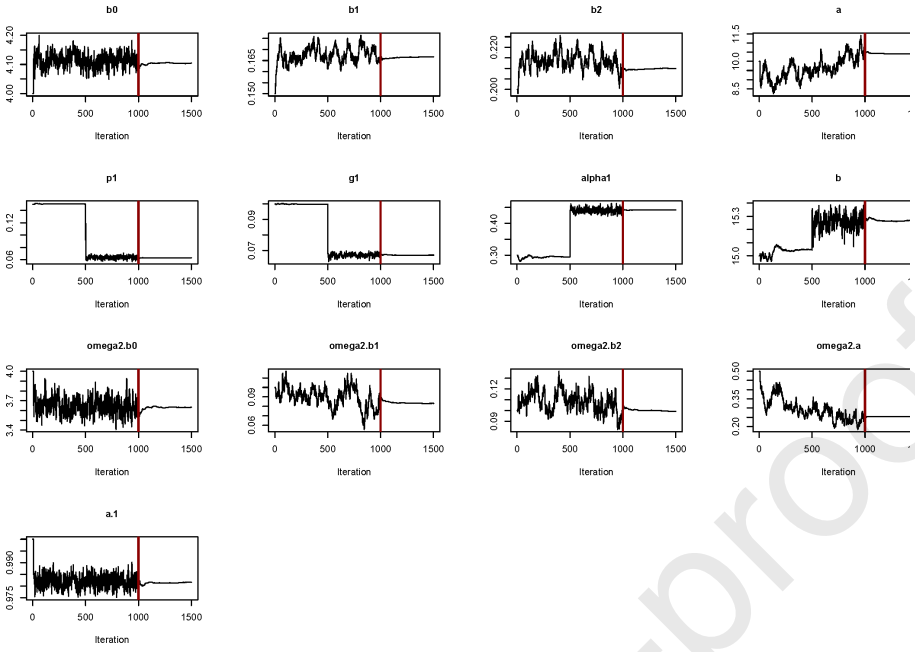


Figure H.10: Convergence graph showing the evolution of the parameter estimates across the iterations of the algorithm (when initial estimates were set to the true values, and the number of chains for the algorithm was set to 10). Results are shown for the first simulated dataset for JM-NLMEM-CR.

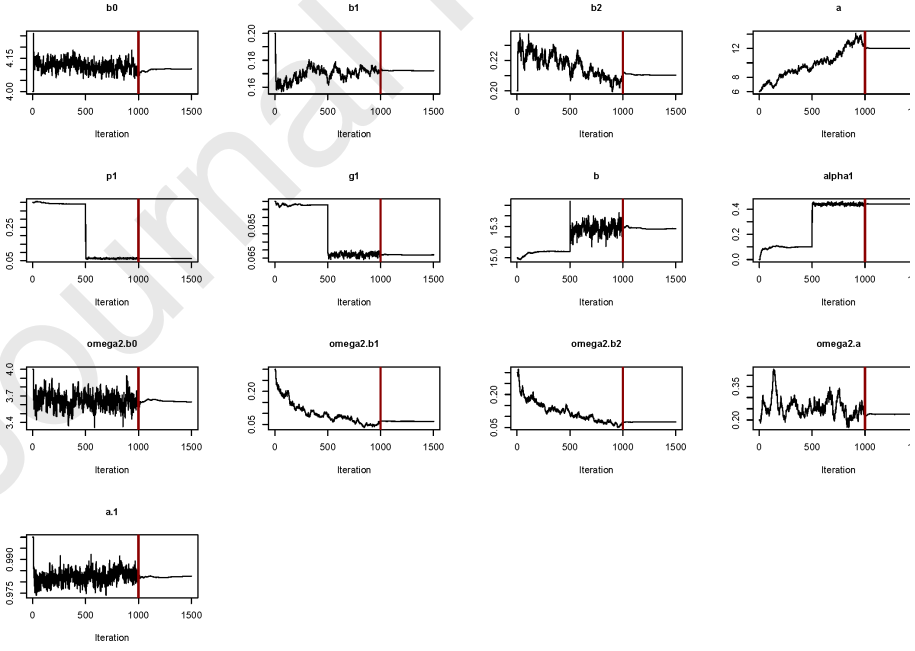


Figure H.11: Convergence graph showing the evolution of the parameter estimates across the iterations of the algorithm (when initial estimates were set using the two separate submodel fits). Results are shown for the first simulated dataset using JM-NLMEM-CR.

Appendix I. Computation times

We fitted models on a shared computing center including 16 CPUs clocked at 3.8GHz (Intel Xeon Gold 5222) and 320GB of RAM memory. The number of chains of the algorithm was set to 3 for both models, in order to allow comparisons.

The results are reported in Figure I.12.

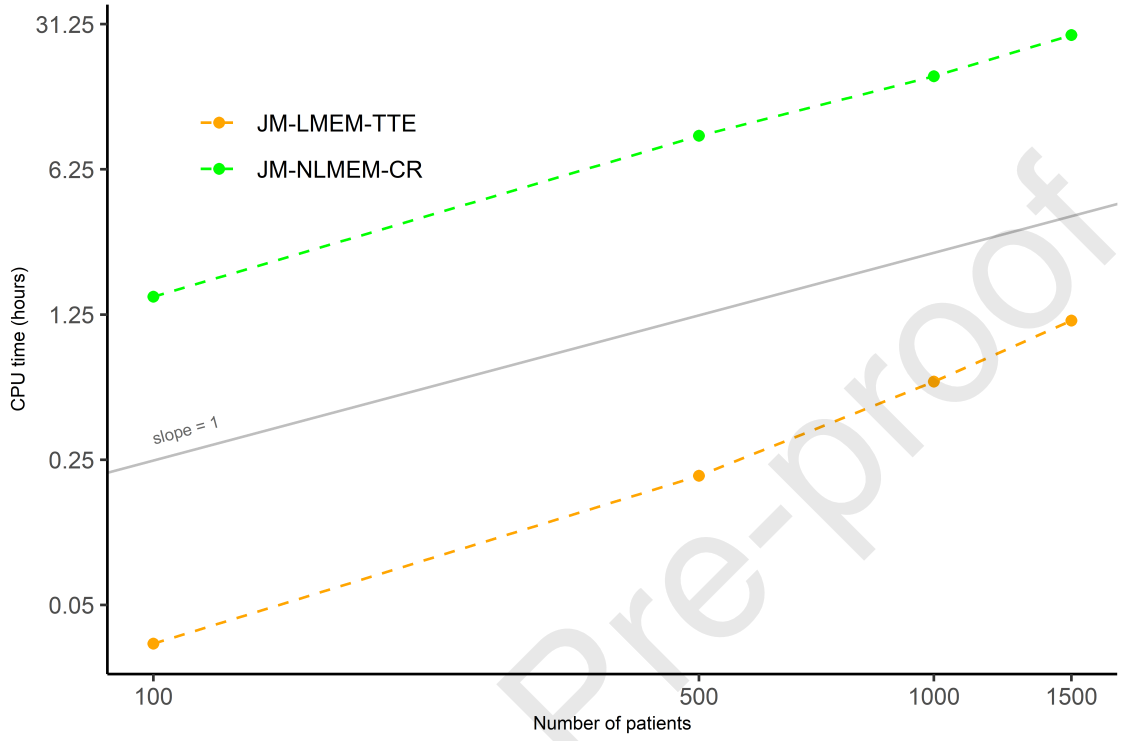


Figure I.12: Time spent to estimate parameters and SE for the joint model with a linear mixed-effects model and a single event (orange), and for the joint model with a nonlinear mixed-effects model and two competing risks (green), depending on the number of subjects included. Note that the figure is in log-log scale. A slope equal to 1 corresponds to a linear growth in computation time with respect to the number of subjects.

Appendix J. Prothrombin example - Fit using *JM* package

Parameters	Estimates	S.E.	R.S.E (%)
Longitudinal model			
Fixed effects			
μ_0	73.5	0.87	1.2
μ_1	0.74	0.28	37.8
Random effects			
ω_0^2	385	0.04	0.01
ω_1^2	15	0.37	2.5
Error model			
σ	17.2	0.015	0.09
Survival model			
h_0	3.01	0.24	8.0
α	-0.038	0.003	7.9

Table J.10: Parameter estimates on the real data application

Appendix K. Estimating JM LMEM-TTE using JM package

We considered the same datasets simulated under the JM-LMEM-TTE model and estimated model parameters using the *JM* package in R (initial points set to the true ones, use of an EM algorithm with a convergence criterion that stops iterations when convergence is achieved). We provide the results as violin plots in Figure K.13.

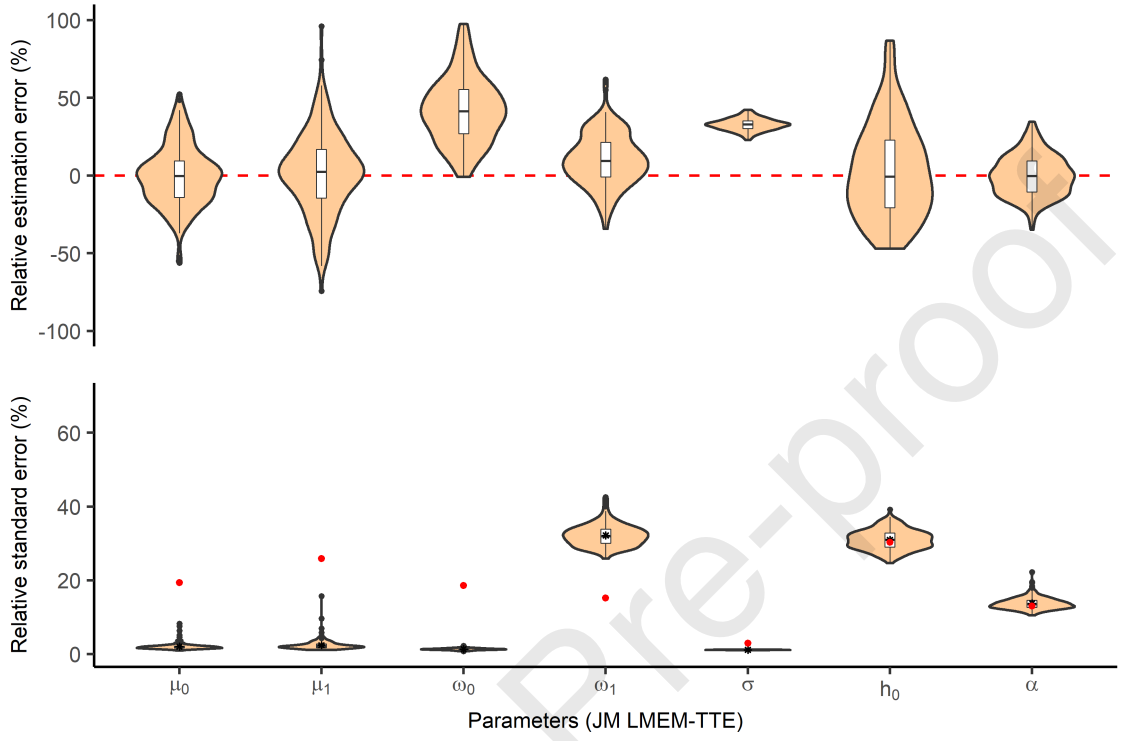


Figure K.13: Distribution of the relative estimation errors (top) and RSE (bottom) for the parameters of the joint model with linear mixed-effects model and time-to-event data (when $\sigma = 1$). Estimation was performed using the JointModel function of the package JM. Stars correspond to the mean of the RSE distribution. Red points correspond to the empirical RSE obtained over the 200 simulations.

A relative bias around 30% was observed for parameter ω_0 and σ. Moreover, SE seemed under-estimated for fixed effects, ω_0 and over-estimated for ω_1. Survival parameter were well estimated as was their SE, while variability for the baseline risk parameter was overestimated. Of note, if we increased the residual error term in the simulations, that is to say simulating data using $\sigma = 3$ (instead of $\sigma = 1$), we obtained better results regarding the bias of σ and ω_0 (see Figure K.14):

Caveat: as we are not experts of JM, we could have missed some settings to optimize the fit. Caution is advised regarding those results, as it is possible that they could be improved changing algorithm settings.

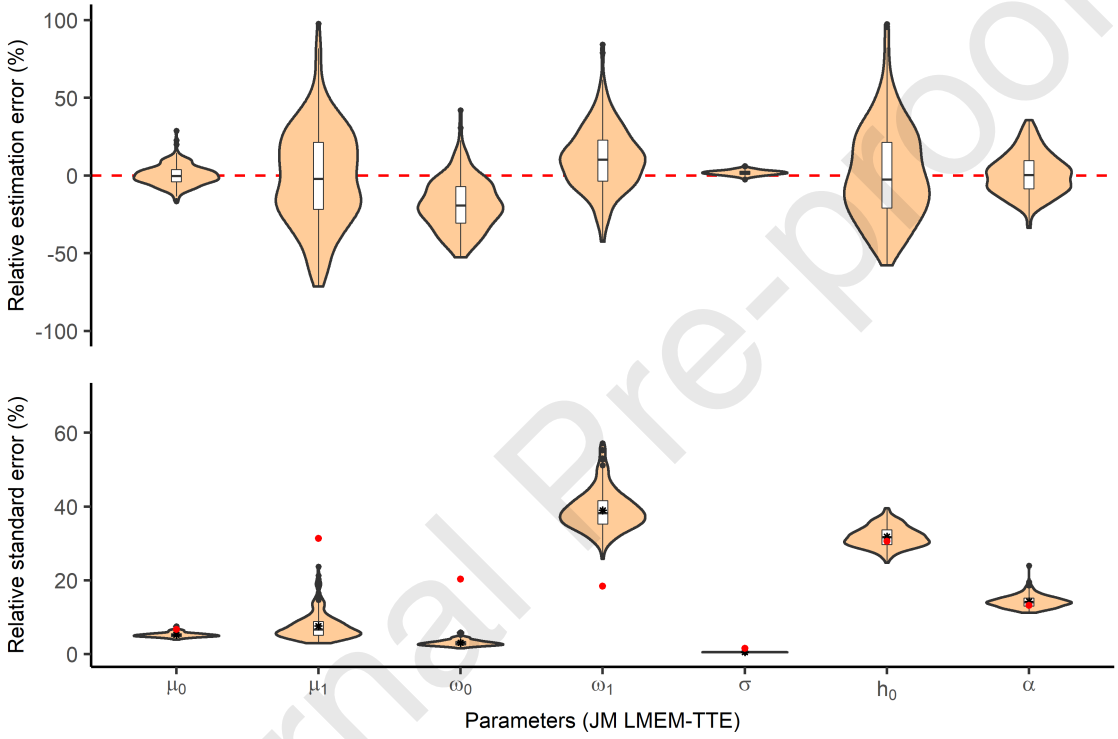


Figure K.14: Distribution of the relative estimation errors (top) and RSE (bottom) for the parameters of the joint model with linear mixed-effects model and time-to-event data (when $\sigma = 3$). Estimation was performed using the JointModel function of the package JM. Stars correspond to the mean of the RSE distribution. Red points correspond to the empirical RSE obtained over the 200 simulations.

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this article.