
JMSTATE, A FLEXIBLE PYTHON PACKAGE FOR MULTI-STATE JOINT MODELING

A PREPRINT

Félix Laplante

MaIAGE
Université Paris-Saclay, INRAE

felixlaplante0@proton.me

Christophe Ambroise

Laboratoire de Mathématiques et Modélisation d'Evry
Université Paris-Saclay, CNRS, Univ Evry

christophe.ambroise@univ-evry.fr

Estelle Kuhn

MaIAGE
Université Paris-Saclay, INRAE

estelle.kuhn@inrae.fr

Sarah Lemler

Mathématiques et Informatique pour la Complexité et les Systèmes
Université Paris-Saclay, CentraleSupélec

sarah.lemler@centralesupelec.fr

October 9, 2025

ABSTRACT

Classical joint modeling approaches often rely on competing risks or recurrent event formulations to account for complex real-world processes involving evolving longitudinal markers and discrete event occurrences. However, these frameworks typically capture only limited aspects of the underlying event dynamics.

Multi-state joint models offer a more flexible alternative by representing full event histories through a network of possible transitions, including recurrent cycles and terminal absorptions, all potentially influenced by longitudinal covariates.

In this paper, we propose a general framework that unifies longitudinal biomarker modeling with multi-state event processes defined on arbitrary directed graphs. Our approach accommodates both Markovian and semi-Markovian transition structures, and extends classical joint models by coupling nonlinear mixed-effects longitudinal submodels with multi-state survival processes via shared latent structures.

We derive the full likelihood and develop scalable inference procedures based on stochastic gradient descent. Furthermore, we introduce a dynamic prediction framework, enabling individualized risk assessments along complex state-transition trajectories.

To facilitate reproducibility and dissemination, we provide an open-source Python library `jmstate` implementing the proposed methodology, available on [PyPI](#). Simulation experiments and a biomedical case study demonstrate the flexibility and performance of the framework in representing complex longitudinal and multi-state event dynamics. The full Python notebooks used to reproduce the experiments as well as the source code of this paper are available on [GitLab](#).

Keywords joint modeling • multi-state processes • longitudinal data • survival analysis • stochastic gradient descent • dynamic prediction • Python library

1 Introduction

Joint modeling of longitudinal and time-to-event data has become an essential tool of modern biostatistics [Papageorgiou et al., 2019], particularly for dynamic prediction in clinical applications. Classical joint models typically couple a longitudinal biomarker process with a single time-to-event outcome [Rizopoulos, 2012], allowing for the integration of biological knowledge and individual heterogeneity via shared latent structures. However, many real-world processes involve multiple possible outcomes, intermediate stages, or recurrent events, which cannot be fully captured by a single-event framework [Król et al., 2017]. In such settings, multi-state models provide a more flexible approach [Ferrer et al., 2016].

Multi-state models represent an overall joint process of clinical course as a series of discrete stages or health states that occur sequentially [Lovblom et al., 2024]. In biostatistics, these models are widely used for survival and reliability analysis, allowing for a richer and more accurate representation by capturing alternative paths to an event of interest, intermediate events, and progressive disease. Key components of multi-state models include transition intensity functions, which denote the instantaneous risk of moving from one state to another, and transition probability functions, which describe the probability of transition over longer intervals. Often, these models assume the Markov property, where future transitions depend only on the current state, simplifying the transition intensity functions [Asanjarani et al., 2022].

However, semi-Markov processes extend this by allowing future probability transitions to depend on the sojourn time, with the clock resetting after each transition.

The link between multi-state models and joint models arises when a multi-state process is integrated as a component within a broader joint modeling framework. While multi-state models, such as those described in the “sequential state framework” primarily focus on movements between discrete states, joint models operate under a “parallel trajectory framework” that combines a longitudinal process with a time-to-event process.

The model proposed by Ferrer et al. [Ferrer, Rondeau, Dignam, Pickles, Jacqmin-Gadda, and Proust-Lima, 2016] exemplifies this link by presenting a joint model for a longitudinal process (e.g., Prostate-Specific Antigen measurements) and a multi-state process (e.g., clinical progressions in prostate cancer). These two sub-models are interconnected by shared random effects, allowing the model to account for the correlation between the continuous longitudinal biomarker trajectory and the discrete transitions between health states.

Several software packages have been developed in R and Python to fit joint models that combine longitudinal and time-to-event data. The Jmbayes package [Rizopoulos, 2016] offers a Bayesian implementation of joint models using Markov chain Monte Carlo (MCMC) methods. Its successor, Jmbayes2 [Rizopoulos et al., 2024], extends this framework to accommodate more complex data structures, such as multiple longitudinal outcomes, nonlinear trajectories, and competing risks. Another popular R package, joiner [Philipson et al., 2018], enables joint modeling of repeated measurements and survival outcomes, based on the methodology introduced by Williamson et al. [2008]. In the Python ecosystem, the flash package [Nguyen et al., 2024] provides a scalable joint modeling approach tailored to high-dimensional settings, exploiting sparse association patterns between longitudinal and survival processes.

Despite the richness of these tools, none currently supports joint models in the multi-state framework with nonlinear longitudinal modeling. In this work, we build on the approach of Ferrer et al. [Ferrer, Rondeau, Dignam, Pickles, Jacqmin-Gadda, and Proust-Lima, 2016] and introduce `jmstate`, a Python package for joint modeling of nonlinear longitudinal and multi-state time-to-event data. The underlying model of `jmstate` incorporates a linear mixed-effects submodel for the longitudinal biomarker and a multi-state process for the event history, linked through shared random effects as in [Ferrer, Rondeau, Dignam, Pickles, Jacqmin-Gadda, and Proust-Lima, 2016].

The model we propose operates on an arbitrary directed graph, enabling representation of complex state transitions and recurrent events. We derive the full likelihood for the nonlinear joint model, introduce an efficient stochastic approximation inference method, and develop dynamic prediction tools. To illustrate its practical relevance, we conduct both a simulation study and a synthetic biomedical case study using the `jmstate` PyPI package.

This paper is organized as follows. Section 2 presents background on joint modeling frameworks and multi-state processes. Section 3 introduces the proposed multi-state joint model, detailing its likelihood formulation and inference strategies. Section 4 develops dynamic prediction tools. Section 5 details efficient inference methods based on stochastic approximation. Section 6 provides implementation details. Section 7 exemplifies the use of `jmstate` through a simulation study while Section 8 illustrates the use of `jmstate` on data from the PAQUID cohort. Finally, Section 9 concludes with a discussion of the main strengths, limitations, and future directions of this work.

2 Background

2.1 Notations

Each individual i , where $i = 1, \dots, n$, is observed at times t_{ij} , where $j = 1, \dots, n_i$. n_i is the number of observations for an individual i . The biomarker values at these times are denoted by $Y_{ij} \in \mathbb{R}^d$, where d is the number of biomarkers. Each individual also has a set of covariates $X_i \in \mathbb{R}^k$, which may include demographic or clinical characteristics.

2.2 Standard Joint Modeling Framework

Joint modeling frameworks are essential to link longitudinal and time-to-event data for more accurate dynamic predictions. The standard framework [Rizopoulos, 2012, 2011] consists of:

- **Mixed-Effects for Longitudinal Data.** The longitudinal process Y_{ij} for subject i at time t_{ij} is typically modeled as:

$$Y_{ij} = h(t_{ij}, \psi_i) + \epsilon_{ij}, \quad \psi_i = f(\gamma, X_i, b_i),$$

where γ are the population parameters, b_i are individual random effects, and ϵ_{ij} is a random measurement error term. It is usually assumed to be Gaussian with zero mean. The functions h and f can be nonlinear, capturing complex within-subject trajectories and between-subject variability. In practice, h is often a linear predictor of the biomarker values, while f may represent a nonlinear function of the random effects.

Both h and f may be nonlinear, allowing flexible modeling of complex biomarker dynamics. Typical choices include:

- $h(t, \psi_i) = \psi_{i1} + \psi_{i2}t + \psi_{i3}t^2$ (polynomial growth),
- $h(t, \psi_i) = \psi_{i1} \exp(-\psi_{i2}t)$ (exponential decay),
- $f(\gamma, X_i, b_i) = \gamma X_i + b_i$ [Davidian and Giltinan, 2003], $\gamma + b_i$, or $\exp(\gamma + b_i)$ for log-normally distributed parameters.

- **Cox-Based Survival Modeling.** The survival process is described by a hazard function incorporating the current (and possibly historical) longitudinal process and baseline covariates:

$$\lambda_i(t | X_i, \mathcal{H}_i(t)) = \lambda_0(t) \exp(\alpha g(t, \psi_i) + \beta X_i),$$

where $\mathcal{H}_i(t)$ is the information that would be provided by knowing the "true trajectory" of the longitudinal biomarker process, that is, $\mathcal{H}_i(t) = \{(u, h(u, \psi_i)) : 0 \leq u \leq t\}$ and $g(\cdot)$ links the biomarker trajectory to the hazard, with α et β as association parameters. The baseline hazard $\lambda_0(t)$ may be parametric or semiparametric with a spline basis representation [Andrinopoulou et al., 2018].

- **Prior distribution.** A prior has to be specified for the latent variables b_i . Here, we assume it to be Gaussian with zero mean: $\mathcal{N}(0, Q)$ but other prior distributions can be considered.

In joint modeling frameworks, the longitudinal process is linked to the time-to-event component through a function g of the underlying trajectory. Several choices for g have been proposed, depending on the hypothesized biological mechanism. Current examples include [Papageorgiou et al., 2019, Mauff, 2023]:

- $g(t, \psi_i) = h(t, \psi_i)$: direct effect of the current biomarker level,
- $g(t, \psi_i) = \frac{\partial}{\partial t} h(t, \psi_i)$: effect of the biomarker slope (rate of change),
- $g(t, \psi_i) = \int_0^t h(w, \psi_i) dw$: cumulative exposure.

This joint framework not only captures the dynamics of longitudinal biomarkers but also leverages their predictive power to inform survival outcomes, enabling dynamic, individualized risk predictions [Rizopoulos, 2011, Ibrahim et al., 2010].

Moreover, in order to derive an expression for the marginal (joint) likelihood, joint modeling typically requires the following independence assumptions [Rizopoulos, 2012]:

Assumption 1. *The latent variables $(b_i)_i$ are independent across individuals.*

Assumption 2. *The observed markers $(Y_{ij})_{ij}$ are independent conditionally on $(b_i)_i$.*

Assumption 3. *The occurrence times $(T_i^*)_i$ are independent conditionally on $(b_i)_i$.*

Assumption 4. *The occurrence times $(T_i^*)_i$ and the observed markers $(Y_{ij})_{ij}$ are mutually independent conditionally on $(b_i)_i$.*

Assumption 5. *The censoring times $(C_i)_i$ are independent and noninformative: conditional on $(b_i)_i$ (and possibly covariates), $(C_i)_i$ is independent of both the event times $(T_i^*)_i$ and the longitudinal measurements $(Y_{ij})_{ij}$.*

These assumptions facilitate the derivation of the marginal likelihood by allowing the decomposition of the joint distribution into manageable components. They ensure that the latent variables capture all relevant dependencies between the longitudinal and survival processes, enabling efficient estimation and inference.

2.3 Multi-State Markov Processes

A *multi-state stochastic process* is defined as a process $S(t)$ for $t \geq 0$, where $S(t)$ can take a finite number of values (states), often labeled $1, 2, \dots, p$. Quantities of interest typically include the probability of being in a certain state at a given time and the distribution of first passage times.

A *Markov process* (or continuous-time Markov chain) is a specific class of multi-state models in which future transitions between states *depend only upon the current state*. This *Markov property* means the process is *memoryless*. A key consequence is that the duration spent in any state follows an *exponential distribution*, implying a constant hazard rate for leaving that state [Jackson, 2011, Putter et al., 2007].

However, the Markov assumption can be unrealistic in many real-world applications. For example, in the study of human sleep stages, sojourn times often do not follow an exponential distribution [Dong et al., 2008, Roever et al., 2010], and in the case of chronic diseases such as AIDS, the risk of disease progression can depend on the time elapsed since infection [Andersen and Keiding, 1999]. To address these limitations, *semi-Markov processes* (SMPs) were introduced, which allow for arbitrary distributions of sojourn times while retaining the Markov property for the embedded discrete-time chain. This flexibility makes SMPs suitable for modeling complex disease progression and patient recovery scenarios [Putter et al., 2007].

2.4 Multi-State Semi-Markov Processes

Multi-state semi-Markov processes (MSMPs) offer a natural generalization by allowing the distribution of sojourn times to be arbitrary.

In an MSMP, the process is defined by a directed graph $G = (V, E)$, where V represents the set of states and E the set of possible transitions. The transition intensities $\lambda_{k'|k}(t | t_0)$ from state k to state k' at time t , given entry time t_0 . The sojourn time in state k is not restricted to an exponential distribution, allowing for more realistic modeling of the time spent in each state.

MSMPs have been widely applied in a range of disciplines. In reliability, they are used to modeling degradation and repair processes [Limnios and Oprisan, 2001]. In biomedical studies, they have proven useful for analyzing illness-death models or disease progression with non-exponential transitions [Commenges et al., 2006, Fiocco et al., 2008]. Applications in finance also exist, where semi-Markovian dynamics can model credit rating migrations or economic regimes [Luciano and Vasiliev, 2006]. In these contexts, MSMPs retain the Markov property in the embedded jump chain while offering increased realism through flexible dwell time modeling.

From a methodological standpoint, MSMPs extend estimation strategies developed for Markov models, such as maximum likelihood or Bayesian inference, and can accommodate interval-censored or misclassified data [Frydman, 2005]. This makes them a powerful and general tool for multi-state event history analysis.

While both joint models and multi-state models offer powerful tools, they primarily address different facets of disease progression and have complementary strengths. Multi-state models excel at understanding the sequence and timing of discrete events, while joint models are adept at modeling continuous biomarker trajectories and their associations with event outcomes for dynamic prediction. The complexity of real-world diseases often necessitates a more integrated approach that can leverage the strengths of both frameworks. For instance, in prostate cancer, multiple types of relapse may occur successively (a multi-state process), and their risk is influenced by the dynamics of longitudinal biomarkers like PSA.

2.5 Joint Models and Multi-State Processes

Previous research has initiated such integration, with models like the one proposed by Ferrer et al. [Ferrer, Rondeau, Dignam, Pickles, Jacqmin-Gadda, and Proust-Lima, 2016], which combines a **linear** mixed sub-model for longitudinal data with a multi-state sub-model using shared random effects. This model enables the simultaneous analysis of *repeated measurements of a biomarker* (the longitudinal process, e.g., PSA levels) and the *times of transitions between multiple health states* (the multi-state process). It explicitly accounts for the link between these two correlated processes and uses information from the biomarker dynamics to explain or predict clinical progression events. The multi-state sub-model assumes a *non-homogeneous Markov process* for the clinical progression, meaning that the future of the

process depends only on the current state (Markov property) and the time elapsed since study entry (non-homogeneous property).

Although Ferrer et al. [Ferrer et al., 2016] note that their joint modeling framework could be adapted to semi-Markov processes in other contexts, their primary application to prostate cancer progression assumes a non-homogeneous Markov multi-state process. Consequently, the model does not explicitly incorporate the effect of time spent *within* a particular state on the instantaneous risk of transitioning out of that state, beyond the overall follow-up time from study entry.

While this prior work has managed to combine multi-state and joint models in particular applications, these models typically rely on a linear mixed model for the biomarker and impose a specific directed acyclic graph (DAG) structure through their likelihood decomposition, which restricts the types of dependency structures they can represent. As a result, limitations remain in their generalizability, flexibility, and the scalability of inference. In what follows, we propose a more general framework defined on arbitrary directed graphs (not restricted to acyclic forms) that accommodates both Markov and semi-Markov assumptions and allows nonlinear longitudinal submodels.

3 Extending Multi-State Joint Models

Joint models traditionally focus on a single time-to-event outcome, which limits their ability to represent complex disease trajectories involving multiple intermediate and recurrent events. To address this limitation, we extend the classical nonlinear joint modeling framework to incorporate multi-state processes, enabling the joint analysis of longitudinal biomarkers and transitions between multiple health states. This is achieved by introducing a directed graph structure $G = (V, E)$, which encodes all permissible transitions and supports both Markov and semi-Markov assumptions.

3.1 Graph Structure and Example

Let us consider the illustrative state-transition diagram shown in Figure 1, representing a 4-state process.

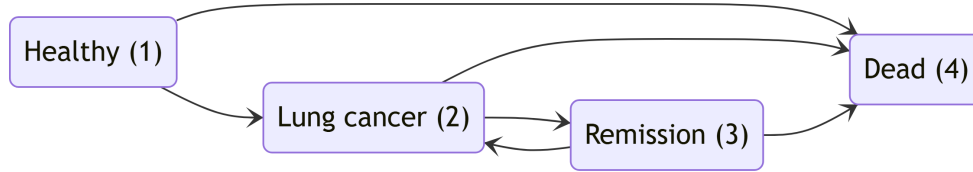


Figure 1: Illustration of the 4-state transition graph $G = (V, E)$, including absorbing state 4 (Dead).

This graph can be encoded by an adjacency matrix $A = [A_{kk'}]_{kk'}$ where $A_{kk'} = 1$ if a transition from state k to k' is possible, and 0 otherwise. For the graph in Figure 1, the adjacency matrix is:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Under certain assumptions borrowed from the semi-Markov literature, competing and recurrent risks mentioned in the introduction, as well as the classical joint model, can be represented using such graphs as illustrated in Figure 2.

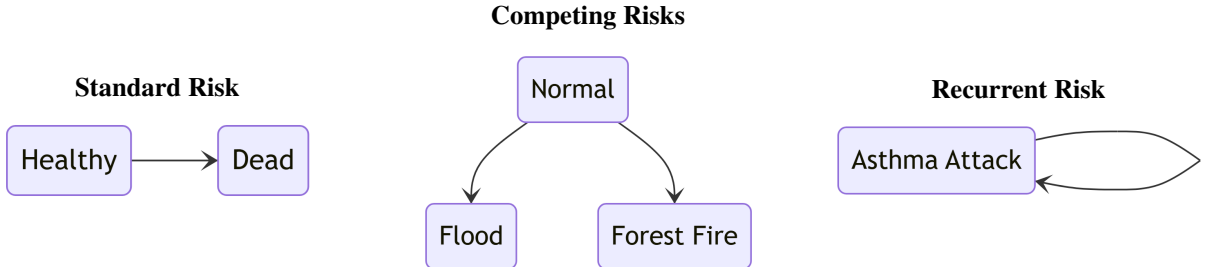


Figure 2: Illustration of classical joint modeling approaches: (left) standard risk, (middle) competing risks, and (right) recurrent risk.

Such joint modeling offers a flexible framework that extends beyond classical approaches and is particularly useful in medical applications.

3.2 Individual Trajectories and the Markov Assumption

Each individual i follows a true trajectory:

$$\mathcal{T}_i^* = ((T_{i0}, S_{i0}), (T_{i1}, S_{i1}), (T_{i2}, S_{i2}), \dots),$$

where T_{il} denotes the l -th transition time and $S_{il} \in V$ the corresponding state. Between transitions, the state $S_i(t)$ is assumed constant.

Only a finite portion of the trajectory is observed, up to a right-censoring time C_i :

$$(\mathcal{T}_i, C_i) = \left(((T_{i0}, S_{i0}), \dots, (T_{im_i}, S_{im_i})), C_i \right),$$

where $m_i(t) = \max\{l : T_{il} \leq t\}$, and $m_i = m_i(C_i)$ denotes the last observed transition index.

We assume that the process satisfies the (time-inhomogeneous) Markov property:

$$\mathcal{L}((T_{i(l+1)}, S_{i(l+1)}) \mid \{(T_{il'}, S_{il'})\}_{l' \leq l}) = \mathcal{L}((T_{i(l+1)}, S_{i(l+1)}) \mid (T_{il}, S_{il})). \quad (*)$$

Remark 1. *The transition times are not restricted to be non-negative.*

3.3 Transition Intensities and Survival Functions

Let $(k, k') \in V^2$. The instantaneous risk of transition from state k to state k' at time t , given entry at time t_0 , is defined as:

$$\lambda_i^{k'|k}(t \mid t_0) := \lim_{\delta \rightarrow 0^+} \frac{\mathbb{P}(T_{i(l+1)} \leq t + \delta, S_{i(l+1)} = k' \mid T_{i(l+1)} > t, T_{il} = t_0, S_{il} = k)}{\delta}.$$

Summing over all admissible transitions from k , we get the total instantaneous risk of leaving state k :

$$\sum_{s: (k, s) \in E} \lambda_i^{s|k}(t \mid t_0) = \lim_{\delta \rightarrow 0^+} \frac{\mathbb{P}(T_{i(l+1)} \leq t + \delta \mid T_{i(l+1)} > t, T_{il} = t_0, S_{il} = k)}{\delta}.$$

The cumulative intensity of the transition from k to k' is defined by:

$$\Lambda_i^{k'|k}(t \mid t_0) = \int_{t_0}^t \lambda_i^{k'|k}(w \mid t_0) dw,$$

which leads to the corresponding survival function:

$$\mathbb{P}(T_{i(l+1)} > t \mid T_{il}, S_{il}) = \exp \left(- \sum_{s: (S_{il}, s) \in E} \Lambda_i^{s|S_{il}}(t \mid T_{il}) \right).$$

Conditional transition probabilities are given by:

$$\mathbb{P}(S_{i(l+1)} \mid T_{i(l+1)}, T_{il}, S_{il}) = \frac{\lambda_i^{S_{i(l+1)}|S_{il}}(T_{i(l+1)} \mid T_{il})}{\sum_{s: (S_{il}, s) \in E} \lambda_i^{s|S_{il}}(T_{i(l+1)} \mid T_{il})}.$$

Combining both, the joint density is:

$$p(T_{i(l+1)}, S_{i(l+1)} \mid T_{il}, S_{il}) = \lambda_i^{S_{i(l+1)}|S_{il}}(T_{i(l+1)} \mid T_{il}) \exp \left(- \sum_{s: (S_{il}, s) \in E} \Lambda_i^{s|S_{il}}(T_{i(l+1)} \mid T_{il}) \right).$$

Furthermore, conditioning on the event $T_{i(j+1)} \geq t$ for some known t can also easily be achieved by using the Chasles relation:

$$\forall u \geq t,$$

$$\mathbb{P}(T_{i(l+1)} > u \mid T_{il}, S_{il}, T_{i(l+1)} > t) = \exp \left(- \sum_{s: (S_{il}, s) \in E} \int_t^u \lambda_i^{s|S_{il}}(w \mid T_{il}) dw \right).$$

3.4 Multi-State Joint Model Specification

The proposed multi-state joint model with Gaussian prior and homoscedastic Gaussian noise is specified by:

$$\begin{cases} Y_{ij} = h(t_{ij}, \psi_i) + \epsilon_{ij}, \\ \psi_i = f(\gamma, X_i, b_i), \quad b_i \sim \mathcal{N}(0, Q), \quad \epsilon_{ij} \sim \mathcal{N}(0, R), \\ \lambda_i^{k'|k}(t \mid X_i, T_{im_i(t)}, \mathcal{H}_i(t)) = \lambda_0^{k'|k}(t \mid T_{im_i(t)}) \cdot \exp(\alpha^{k'|k} g^{k'|k}(t, X_i, \psi_i) + \beta^{k'|k} X_i), \end{cases}$$

along with the assumptions 1, 2, and 5 as well as the following assumptions, analogous to assumptions 3 and 4:

Assumption 6. *Conditionally on b_i , the trajectories $\mathcal{T}_i^*$ are independent and satisfy the semi-Markov property (*).*

Assumption 7. *Conditionally on b_i , the biomarker process Y_{ij} is independent of the transition process $\mathcal{T}_i^*$.*

3.5 Clock Reset vs. Clock Forward

Two baseline hazard specifications are commonly used:

$$\lambda_0^{k'|k}(t \mid T_{im_i(t)}) = \begin{cases} \lambda_0^{k'|k}(t - T_{im_i(t)}) & \text{(clock reset),} \\ \lambda_0^{k'|k}(t) & \text{(clock forward).} \end{cases}$$

The *clock reset* model resets the baseline hazard at each state entry and is appropriate when the transition risk depends on time spent in the state (Asanjarani et al. [2022]). The *clock forward* model assumes that risk accumulates with global time, ignoring the state entry time.

3.6 Initial State

We assume that the initial state S_{i0} is observed. However, the framework could also incorporate a multinomial model for unobserved initial states (Yiu et al. [2018]).

This flexible framework enables inference on complex event histories, leveraging the full trajectory of biomarkers to refine predictions of transitions and survival [de Wreede et al., 2010].

3.7 Multi-State Simulation

In nonlinear joint models with known design and known parameters parameters $\theta = (\gamma, Q, R, \alpha, \beta)$, the occurrence times of events conditionally on the latent variables b_i and the covariates X_i can be easily sampled using the inverse cumulative distribution function transform method. This may be achieved through bisection or other root-finding algorithms.

The simulation of the transition process $\mathcal{T}_i^*$ conditionally on b_i , X_i , and (T_{i0}, S_{i0}) can be achieved drawing on the semi-Markov property 6 by considering one transition at a time, according to Algorithm 1. The algorithm is similar to Gillespie's algorithm [Gillespie, 2007], and can also include a survival condition $T_{i1} \geq t_i^{\text{surv}}$ that uses the Chasles relation 3.3.

Algorithm 1 Simulation of trajectory i

Input: subject-specific censoring time $C_i \in \mathbb{R} \cup \{+\infty\}$; covariates X_i ; latent variables b_i ; initial time-state pair (T_{i0}, S_{i0}) ; optional survival condition $T_{i1} \geq t_i^{\text{surv}}$, defaults to $t_i^{\text{surv}} = -\infty$.

```

1: Initialize  $\ell_i \leftarrow 1, \mathcal{T}_i \leftarrow ((T_{i0}, S_{i0}))$ .

2: while  $T_{i(\ell_i-1)} < C_i$  and  $\{s : (S_{i(\ell_i-1)}, s) \in E\} \neq \emptyset$  do
3:   for each  $s$  with  $(S_{i(\ell_i-1)}, s) \in E$  do
4:     Draw  $T_{i\ell_i}^s$  such that
       
$$-\log \mathbb{P}(T_{i\ell_i}^s > t) = \int_{T_{i(\ell_i-1)} \vee t_i^{\text{surv}}}^t \lambda_i^{s|S_{i(\ell_i-1)}}(w \mid \theta, X_i, T_{i(\ell_i-1)}, b_i) dw,$$

       
$$\forall t \geq T_{i(\ell_i-1)} \vee u.$$

5:   end for

6:   Set  $T_{i\ell_i} \leftarrow \min_m T_{i\ell_i}^s$ , and  $S_{i\ell_i} \leftarrow \arg \min_m T_{i\ell_i}^s$ .
7:   Append  $(T_{i\ell_i}, S_{i\ell_i})$  to  $\mathcal{T}_i$ .
8:    $\ell_i \leftarrow \ell_i + 1$ .
9: end while

10: if  $T_{i(\ell_i-1)} > C_i$  then
11:   Remove the last pair:  $\mathcal{T}_i \leftarrow \mathcal{T}_i[: -1]$ .
12: end if

```

The proof of the algorithm is provided in Appendix A.

4 Dynamic Prediction

We now focus on constructing random variables of interest for multi-state prediction for a new individual i , such that these random variables may be feasibly simulated with Algorithm 1.

Let τ_i be a stopping time adapted to the filtration $\mathcal{F}_{in} = \sigma(T_{i0}, S_{i0}, \dots, T_{in}, S_{in})$, and let $\kappa_i = \{\inf n \in \mathbb{N} : \mathbb{P}(\tau_i = +\infty \mid \mathcal{F}_{in}) = 1\}$ be a second stopping time.

Assume that:

$$\mathbb{P}(\min(\tau_i, \kappa_i) < +\infty) = 1.$$

Here, τ_i represents the time at which an event occurs for an individual i , while κ_i is the time at which it becomes certain that the event will not occur. For instance, if the event of interest is patient remission, but the patient dies without remission, then $\tau_i = +\infty$, while κ_i would be finite. As such, it is almost surely possible to simulate the trajectory of an individual i up to the time of the event or the time of censoring.

Let ξ be a function taking values in some space $\mathbb{Y}$:

$$\begin{aligned} \xi : (\mathbb{N} \cup \{+\infty\})^2 \times (\mathbb{R} \times V)^{\mathbb{N}} &\longrightarrow \mathbb{Y} \\ \tau_i, \kappa_i, \mathcal{T}_i^* &\longmapsto \xi(\tau_i, \kappa_i, \mathcal{T}_i^*) = \xi(\tau_i, \kappa_i, (T_{il}, S_{il})_{l \leq \min(\tau_i, \kappa_i)}) \end{aligned}$$

This function therefore only depends on the beginning of the trajectory and not on its entirety. Thus, during simulations, if τ_i is finite or almost surely infinite after a certain index, there is no need to simulate the entire trajectory. Applications include, for instance, the distributions of states at any time, transition times, return times, sojourn times, hitting times and many other quantities. This flexibility is illustrated by Examples 1 and 2.

Example 1 (State at time t). Let $G = (V, E)$ be a directed acyclic graph, $t \in \mathbb{R}$ be a fixed time, and $\mathbb{Y} = V$ be the set of possible states. Let $\tau_i = \inf\{n \in \mathbb{N} : T_{in} \geq t \text{ or } S_{in} \text{ is absorbing}\}$. Clearly, $\tau_i \leq \text{depth}(G) < +\infty$. Then, take $\xi_t(\tau_i, \kappa_i, \mathcal{T}_i^*) = S_{i\tau_i}$, which corresponds to the state of the individual i at time t .

Example 2 (Hitting time). Let $G = (V, E)$ be a directed acyclic graph, $\mathbb{A} \subset V$ a non-empty subset of states, and $\mathbb{Y} = \mathbb{R} \cup \{+\infty\}$. For any two non-empty sets $\mathbb{A}, \mathbb{B} \subset V$, we note $\mathbb{A} \rightsquigarrow \mathbb{B}$ if there exists a path from $\mathbb{A}$ to $\mathbb{B}$ in G . Let $\tau_i = \inf\{n \in \mathbb{N} : S_{in} \in \mathbb{A}\}$, with the convention $\inf \emptyset = +\infty$. Then, $\kappa_i = \inf\{n \in \mathbb{N} : \{S_{in}\} \not\rightsquigarrow \mathbb{A}\}$ and $\min(\tau_i, \kappa_i) \leq \text{depth}(G) < +\infty$. Finally, $\xi_{\mathbb{A}}(\tau_i, \kappa_i, \mathcal{T}_i^*) = \mathbb{1}_{\tau_i < +\infty} T_{i\tau_i} + \mathbb{1}_{\tau_i = +\infty} \tau_i$ represents the hitting time for the set $\mathbb{A}$.

5 Likelihood and Statistical Inference

The estimation of model parameters $\theta = (\gamma, Q, R, \alpha, \beta)$ relies on two likelihood formulations: the complete-data likelihood $\mathcal{L}_{\text{full}}$, which includes latent variables, and the marginal likelihood $\mathcal{L}_{\text{marginal}}$, obtained by integrating them out:

$$\mathcal{L}_{\text{marginal}}(\theta; X, Y, \mathcal{T}, C) = \int \mathcal{L}_{\text{full}}(\theta; X, Y, \mathcal{T}, C, b) db,$$

where:

$$\mathcal{L}_{\text{full}}(\theta; X, Y, \mathcal{T}, C, b) = \prod_{i=1}^n p(b_i | \theta) p(\mathcal{T}_i, C_i | \theta, X_i, b_i) \prod_{j=1}^{m_i} p(Y_{ij} | \theta, X_i, b_i).$$

This expression is very similar to that obtained by Rizopoulos [2012] and is subdivided into three terms:

Prior likelihood:

$$p(b_i | \theta) = \frac{(2\pi)^{-q/2}}{\det(Q)^{1/2}} \exp\left(-\frac{1}{2} b_i^T Q^{-1} b_i\right),$$

Longitudinal likelihood:

$$p(Y_{ij} | \theta, X_i, b_i) = \frac{(2\pi)^{-d/2}}{\det(R)^{1/2}} \exp\left(-\frac{1}{2} (Y_{ij} - h(t_{ij}, \psi_i))^T R^{-1} (Y_{ij} - h(t_{ij}, \psi_i))\right),$$

with $\psi_i = f(\gamma, X_i, b_i)$.

Semi-Markov likelihood:

$$p(\mathcal{T}_i, C_i | \theta, X_i, b_i) = \prod_{l=0}^{m_i-1} p((T_{i(l+1)}, S_{i(l+1)}) | \theta, X_i, (T_{il}, S_{il}), b_i) \exp\left(-\sum_{s: (S_{im_i}, s) \in E} \Lambda_i^{k'|S_{im_i}}(C_i | \theta, X_i, T_{im_i}, b_i)\right). \quad (1)$$

The proof of the expression of the Semi-Markov likelihood (1) is provided in Appendix B.

It is to be noted that this integral is typically intractable in closed form due to the nonlinear nature of the model, necessitating numerical approximation methods such as Monte Carlo integration or Laplace approximation.

5.1 Stochastic Gradient Estimation via Fisher Identity

Several optimization methods adapted from the nonlinear mixed-effects literature can be applied, including Stochastic EM [Kuhn and Lavielle, 2004], Laplace approximation [Wolfinger, 1993], Gauss-Hermite quadrature, stochastic gradient ascent with Robbins–Monro updates [Robbins and Monro, 1951], and MCMC-based approximations such as Metropolis–Hastings and Hamiltonian Monte Carlo. These strategies are implemented in software such as **JMBayes** [Rizopoulos et al., 2024, Rizopoulos, 2020]. Another approach requiring mild regularity assumptions on the log marginal likelihood, but without the need for the model to belong in the exponential family, is to consider a stochastic gradient ascent scheme [Caillebotte et al., 2025, Baey et al., 2023] using Fisher’s identity and following the Robbins-Monro procedure [Robbins and Monro, 1951].

Under interchangeability of integral and differentiation, setting $x = (X, Y, \mathcal{T}, C)$ for convenience, the Fisher identity writes:

$$\nabla_{\theta} \log \mathcal{L}_{\text{marginal}}(\theta; x) = \mathbb{E}_{b \sim p(\cdot | x, \theta)} (\nabla_{\theta} \log \mathcal{L}_{\text{full}}(\theta; x, b)).$$

This expectation is approximated using Monte Carlo samples from the posterior $p(\cdot | x, \theta)$, avoiding the need to evaluate the intractable marginal likelihood $\mathcal{L}_{\text{marginal}}(\theta; x)$. This formulation naturally supports stochastic gradient ascent with MCMC-based posterior sampling and is amenable to minibatch parallelization across individuals, ensuring the convergence to a critical point. The update rule is as follows:

Algorithm 2 Stochastic gradient ascent model inference

Input: data x ; initial parameter θ_0 ; step sizes $(\eta_n)_{n \geq 0}$ with $\sum_n \eta_n = +\infty$, $\sum_n \eta_n^2 < +\infty$; batch size K ; sampler p_n^{MCMC} ; stopping criterion.

```

1: while not converged do
2:   Draw  $b_k \sim p(\cdot \mid x, \theta_n) \approx p_n^{\text{MCMC}}$ .
3:    $\theta_{n+1} \leftarrow \theta_n + \frac{\eta_n}{K} \sum_{k=1}^K \nabla_{\theta} \log \mathcal{L}(\theta_n; x, b_k)$ .
4: end while

```

Preconditioning matrices may also be used, such as the Fisher Information Matrix in a natural gradient descent framework, or even specialized *optimizers* such as Adam [Kingma and Ba, 2014], NAdam [Dozat, 2016] or Adagrad [Duchi et al., 2011]. In particular, the Adam optimizer will be later used in our numerical experiments.

6 Implementation Details

The nonlinear joint model is fully implemented in Python, with PyTorch and NumPy as its only major dependencies. Notably, the stochastic gradient scheme is carried out following Algorithm 2 and heavily relies on PyTorch’s automatic differentiation and broadcasting capabilities [Paszke et al., 2017].

6.0.0.1 Posterior Sampling and Parameter Estimation. A Metropolis-Hastings sampler is used to simultaneously draw posterior samples of the random effects in an efficient way. Adaptive step sizes are implemented at the individual level via the Robbins-Monro procedure. We favored Metropolis-Hastings over alternatives such as Hamiltonian Monte Carlo or Metropolis-within-Gibbs for its simplicity. Autocorrelation can be monitored and mitigated by subsampling. The acceptance rate is targeted at 0.234, which is asymptotically optimal for Gaussian proposals in high dimensions [Gelman et al., 1997]. Although our posteriors are not Gaussian, this rule is simple and effective. Moreover, as the number of longitudinal measurements increases ($m_i \rightarrow +\infty$), posterior distributions converge to a multivariate normal [Rizopoulos et al. [2008]].

6.0.0.2 Numerical integration. Cumulative hazards are computed using Gauss–Legendre quadrature [Lether, 1978], which integrates exactly all polynomials up to degree $2n - 1$ with n nodes:

$$\int_a^b f(t) dt \approx \sum_{i=1}^n f(t_i) w_i.$$

We set $n = 32$ by default. Since nodes are fixed, integration points are cached, using a hash table, outside the backpropagation graph, reducing cost and memory. The positivity of the quadrature weights ensures stable computation and preserves the positivity of integrals.

6.0.0.3 Simulation of event times. Event times are generated by inverting the survival function via a bisection search. An exponential random variable is first drawn to determine the event threshold. Starting from the initial and censoring times, the cumulative hazard is iteratively approximated with Gauss–Legendre quadrature until the threshold is reached. This procedure naturally incorporates censoring and yields the simulated event time.

6.0.0.4 Covariance matrix parameterization. Covariance matrices, needed in both prior and longitudinal likelihoods, are expressed through the precision matrix $P = \Sigma^{-1}$, which avoids explicit inverses. We adopt a *log-Cholesky* parametrization $P = LL^T$, where the diagonal entries of L are parameterized as $L_{ii} = e^{\tilde{L}_{ii}}$. Storing $\tilde{L}_{ii}$ in log-scale ensures uniqueness, stability, and efficient computation of $\log \det P = 2 \operatorname{Tr}(\tilde{L})$. This parametrization, available in full, diagonal, or scalar form, is well suited for automatic differentiation.

7 Simulation Study

To study the robustness of the estimator, we propose a simulation example and compute an estimate of the *Root Mean Square Error* (RMSE) for each parameter, based on different datasets generated from a common underlying distribution.

In parallel, we illustrate the usage of the `jmstate` package through concise code snippets, structured as follows:

- Specification of the true underlying model.
- Simulation of longitudinal and event data.
- Stochastic optimization of the parameters with corresponding diagnostic plots.

7.1 Model Specification

The model considered involves three states (See Figure 3), labeled 0 (healthy), 1 (sick), and 2 (terminal stage). The allowed transitions are $0 \rightarrow 1$, $0 \rightarrow 2$, and $1 \rightarrow 2$, corresponding to a biological modeling of the biomarker evolution, where a too rapid decrease of the biomarker or a too low value leads to the degradation of the patient’s health status.

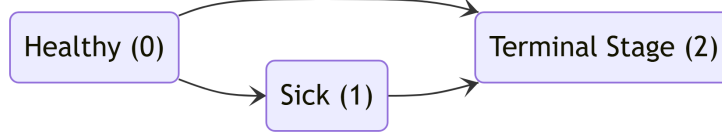


Figure 3: State transition diagram of the multi-state model: transitions are allowed from Healthy (0) to Sick (1), from Healthy (0) to Terminal Stage (2), and from Sick (1) to Terminal Stage (2).

The regression function is a piecewise affine approximation of a bi-exponential curve, with an inflexion point at a fixed known time $\tau = 6$ which may correspond to a drug administration time, causing a rupture in the longitudinal behavior while remaining continuous:

$$h(t, \psi) = \psi_1 + \psi_2 t + \mathbb{1}_{t > \tau} (\psi_3 - \psi_2)(t - \tau),$$

which can be written in Python, after necessary imports, as:

```

import torch

TAU = 6.0

def reg(t: torch.Tensor, psi: torch.Tensor):
    b, w1, w2 = psi.chunk(3, dim=-1) # Extract relevant terms

    # psi has shape (n_chains, n_individuals, n_repetitions)
    # output has shape (n_chains, n_individuals, n_repetitions, 1)
    return (b + w1 * t + (w2 - w1) * (t > TAU) * (t - TAU)).unsqueeze(-1)

```

Note this python function must accept tensors psi with two or three dimensions and return a tensor of dimension $\dim(\psi) + 1$, with a shape whose last dimension matches the dimension of the biomarker (here, $\mathbb{R}^1$ so it is equal to 1).

For the link function, the concatenation of h as well as its partial derivative with respect to time is considered and shared across all possible transitions: $g = (h, \frac{\partial h}{\partial t})$, which is efficiently implemented:

```

def link(t: torch.Tensor, psi: torch.Tensor):
    b, w1, w2 = psi.chunk(3, dim=-1)

    diff = (w2 - w1) * (t > TAU)
    val = b + w1 * t + diff * (t - TAU)
    der = w1 + diff
    return torch.cat([val.unsqueeze(-1), der.unsqueeze(-1)], dim=-1)

```

Additionally, normally distributed covariates X and exponential baseline hazards were considered with a *clock reset* specification. The model design is therefore declared as follows:

```

from jmstate.functions import Exponential, gamma_plus_b
from jmstate.typedefs import ModelDesign

# Survival model specification
surv = {
    (0, 1): (Exponential(0.1), link),
    (0, 2): (Exponential(0.01), link),
    (1, 2): (Exponential(0.2), link),
}

```

```
# Model design gathers regression, link and hazard functions
model_design = ModelDesign(gamma_plus_b, reg, surv)
```

Furthermore, longitudinal measurements were taken at $m_i \leq 20$ time points, and censoring times were drawn from $C_i \sim \mathcal{U}([10, 15])$. Longitudinal observations were also truncated at the censoring times.

The model is therefore parameterized in $\mathbb{R}^{16}$ and the true parameters may be defined as follows:

```
from jmstate.typedefs import ModelParams
from jmstate.utils import repr_from_cov

# Gaussian means
gamma = torch.tensor([2.5, -1.3, 0.2])

# Covariance matrices
Q = torch.diag(torch.tensor([0.6, 0.2, 0.3]))
R = torch.tensor([[1.7]])

# Link parameters
alphas = {
    (0, 1): torch.tensor([-0.5, -3.0]),
    (0, 2): torch.tensor([-1.0, -5.0]),
    (1, 2): torch.tensor([0.0, -1.2]),
}

# Covariate parameters
betas = {
    (0, 1): torch.tensor([-1.3]),
    (0, 2): torch.tensor([-0.9]),
    (1, 2): torch.tensor([-0.7]),
}

# Instance declaration
real_params = ModelParams(
    gamma,
    repr_from_cov(Q, method="diag"),
    repr_from_cov(R, method="ball"),
    alphas,
    betas,
)
```

Note the argument method, which can take values "full", "diag", or "ball". It represents the chosen parameterisation of the covariance matrices, either unconstrained (full), diagonal (diag) or scalar (ball).

7.2 Data Generation

With the parameter values defined above, as well as the choice of the regression and link functions, we are now able to declare the true base model and use it to simulate longitudinal processes and transition trajectories, with the help of the following helper function:

```
from jmstate import MultiStateJointModel
from jmstate.typedefs import ModelData, SampleData
from torch.distributions import MultivariateNormal

# Declares the true underlying model
real_model = MultiStateJointModel(model_design, real_params)
```

```

def gen_data(n: int, m: int):
    # Censoring times
    c = torch.rand(n, 1) * 5 + 10

    # Covariates
    x = torch.randn(n, 1)

    # Latent and noise distributions
    Q_dist = MultivariateNormal(torch.zeros(Q.size(0)), Q)
    R_dist = MultivariateNormal(torch.zeros(R.size(0)), R)

    # Individual effects
    b = Q_dist.sample((n,))
    psi = model_design.individual_effects_fn(gamma, x, b)

    # Generates random evaluations points with a minimum distance
    a = torch.zeros((n, 1))
    b = torch.full((n, 1), 15)
    t = random_far_apart(n, m, a, b, 0.7 * b / m)

    # Defines initial state for individuals
    trajectories_init = [[(0.0, 0)]] * n

    # Samples trajectories
    sample_data = SampleData(x, trajectories_init, psi)
    trajectories = real_model.sample_trajectories(sample_data, c)

    # Samples longitudinal values
    y = model_design.regression_fn(t, psi)
    y += R_dist.sample(y.shape[:2])

    # Censors longitudinal measurements exceeding censoring times
    y[t > c] = torch.nan

    return x, t, y, trajectories, c

# Generate data
data = ModelData(*gen_data(1000, 20))

```

A short summary of the longitudinal process as well as the trajectories is given by the Figure 4 below. The function `build_buckets` was used to determine the number of transitions observed in the dataset. Given a list of trajectories, it returns a dictionary of transitions as keys and Buckets (NamedTuple) as values, containing the individual indices and times of the transitions.

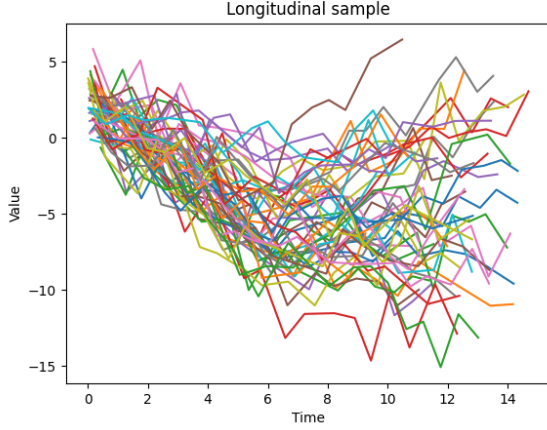


Figure 4: Simulated data: on the left, a sample of longitudinal measurements from 50 individuals; on the right, observed transitions between states for the full population of 1000 individuals.

7.3 Estimator Convergence

To illustrate the convergence of the estimator, we examine the optimization process for a particular run with $n = 1000$ (see Figure 5), as well as RMSE values computed from $n_{\text{runs}} = 100$ independent runs on different simulated datasets, all generated from the same underlying distribution. The results of these simulations are shown in Table 1.

To fit the model, we first have to define initial parameters, which can be particularly important for the optimization process if the model is not well-behaved. In the present case, without prior knowledge of the true parameters, we zero the values and use the identity matrix for both covariance matrices:

```
# Declares initial parameters; zero mean and unit variance
init_params = ModelParams(
    torch.zeros_like(gamma),
    repr_from_cov(torch.eye(Q.size(0)), method="diag"),
    repr_from_cov(torch.eye(R.size(0)), method="ball"),
    {k: torch.zeros_like(v) for k, v in alphas.items()},
    {k: torch.zeros_like(v) for k, v in betas.items()},
)
```

To run the optimization process, we can make use of the Fit class that provides a simple interface for choosing the optimizer (torch.optim.Adam by default) and set its hyperparameters through the kwargs argument. Additionally, a stochastic stopping condition based on the first and second moments of the parameters' consecutive differences can be easily employed with the StopParams class.

Formally:

$$m_i^{(t)} \leftarrow \beta_i m_i^{(t-1)} + (1 - \beta_i)(\theta^{(t)} - \theta^{(t-1)})^i, \quad \hat{m}_i^{(t)} := \frac{m_i^{(t)}}{1 - \beta_i^t},$$

and the optimization process is terminated when:

$$|\hat{m}_1^{(t)}| \leq 10^{-6} + 10^{-1} \sqrt{\hat{m}_2^{(t)}}.$$

At last, the LogParamsHistory class can be used to log the parameters' history, allowing to monitor the convergence of the parameters.

This is all done automatically by executing the do method of the model which runs the Markov Chain Monte Carlo (MCMC) algorithm to sample from the posterior distribution of the parameters based on the inputted data, a list of Job factories (such as fitting, logging, stopping, or even prediction once the model is fitted). It optionally returns metrics collected by the Job factories. The hyperparameters such as the target acceptance rate, the number of chains, the warmup period or the subsample parameter are automatically set based on the job_factories provided but can be overridden:

```

from jmstate.jobs import Fit, LogParamsHistory, ParamStop

# Declares initial model
model = MultiStateJointModel(model_design, init_params)

# Runs optimization process
metrics = model.do( # Metrics object
    data,
    job_factories=[
        Fit(lr=0.5, fused=True),
        ParamStop(rtol=0.1),
        LogParamsHistory(), # Returns a list of ModelParams
    ],
    max_iterations=500,
)

```

Although not demonstrated here, the model also supports gradient clipping, scheduling, stacking multiple other stopping criteria based on likelihood or gradient magnitude, penalized likelihood, as well as covariate selection. For more details, please refer to the official documentation of the `jmstate` package.

The following figure shows the evolution of the parameters during the optimization process and Table 1 summarizes the accuracy of the inferred parameters.

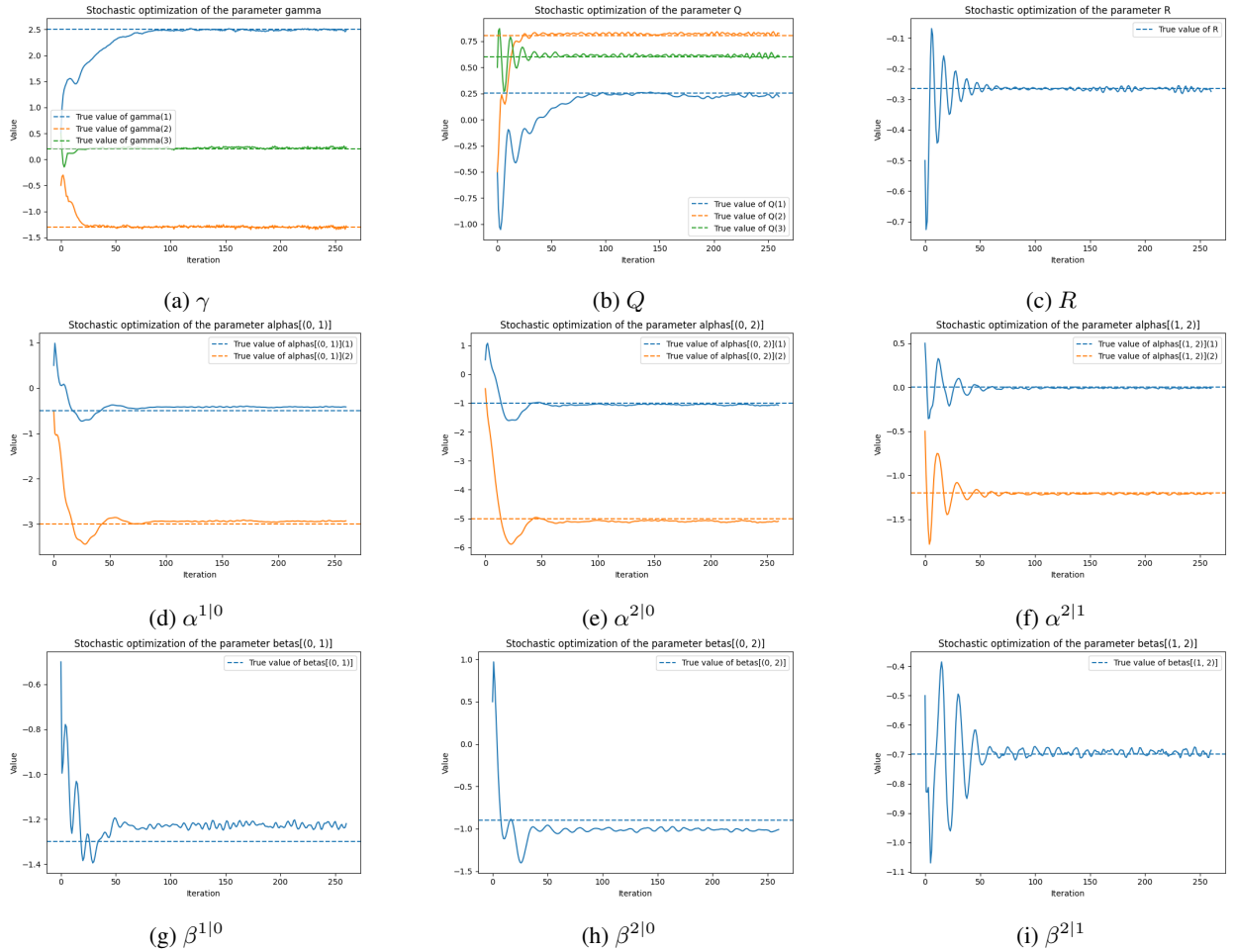


Figure 5: Evolution of the parameters during the optimization of the marginal log-likelihood using stochastic gradient descent. The dotted lines correspond to the true values.

Parameter	True value	Mean	Standard error	RMSE
γ_1	2.5000	2.4960	0.0370	0.0372
γ_2	-1.3000	-1.3039	0.0257	0.0260
γ_3	0.2000	0.1936	0.0267	0.0275
$\bar{Q}_1$	0.2554	0.2243	0.0442	0.0540
$\bar{Q}_2$	0.8047	0.8009	0.0260	0.0263
$\bar{Q}_3$	0.6020	0.6001	0.0276	0.0277
$\bar{R}$	-0.2653	-0.2684	0.0083	0.0089
$\alpha_1^{1 0}$	-0.5000	-0.49997	0.0486	0.0486
$\alpha_2^{1 0}$	-3.0000	-2.9962	0.0800	0.0801
$\alpha_1^{2 0}$	-1.0000	-0.9990	0.0847	0.0847
$\alpha_2^{2 0}$	-5.0000	-4.9897	0.1191	0.1196
$\alpha_1^{2 1}$	0.0000	0.0011	0.0327	0.0327
$\alpha_2^{2 1}$	-1.2000	-1.2045	0.0429	0.0432
$\beta^{1 0}$	-1.3000	-1.3015	0.0501	0.0501
$\beta^{2 0}$	-0.9000	-0.8982	0.0670	0.0670
$\beta^{2 1}$	-0.7000	-0.6981	0.0551	0.0551

Table 1: Comparison of true and inferred parameters (up to 4 decimal places). RMSE is computed using the bias-variance identity.

The algorithm demonstrates good adaptability and robustness, with a notably low bias. The fitting is also very quick to perform, in about 10 to 15 seconds per run on an AMD Ryzen 5 5600 processor, despite the use of mini-batch optimization and the higher complexity of the model induced by the nonlinearity of its functions as well as the presence of numerous possible transitions. It is also to be noted that a minimum number of observations for each transition is required to ensure the stability of the optimization process.

Finally, the convergence patterns shown in Figure 5 are remarkably smooth, a result due to the parallelization of multiple chains and the use of mini-batch optimization.

8 Application to the PAQUID Cohort

8.1 Introduction

The data used in this section originates from the PAQUID cohort [Letenneur et al., 1994], a large prospective population-based study initiated in southwestern France in 1988, aimed at understanding the determinants and trajectories of aging. A subsample of 500 individuals was followed over a period of up to 20 years [Proust-Lima et al., 2017], with repeated measures of cognitive and physical health, as well as socio-demographic characteristics. In particular, global cognitive functioning was assessed through the Mini-Mental State Examination (MMSE, see Figure 6), while physical dependency was evaluated using the HIER scale, which classifies subjects into four ordered states (see Figure 7): 0 (no dependency), 1 (mild dependency), 2 (moderate dependency), and 3 (severe dependency). In this work, we focus on the association between cognitive decline and the progression of physical dependency by jointly modeling the longitudinal trajectory of MMSE scores and the transitions between the four HIER states using a joint multi-state model. More specifically, the state of an individual i at a given time t is defined as the highest HIER dependency score between trial entry and t . This approach allows us to characterize the dynamic interrelationship between cognitive and physical aging processes within a unified statistical framework.

8.2 Model Specification

Since the inference of the transition specific parameters requires one to have observed those said transitions, the state graph was determined based on the dataset, using the `build_buckets` function. Moreover, both time and longitudinal values were normalized in $[-1, 1]$. This technique ensured stable and fast convergence of the optimization process by keeping all parameters on roughly the same scale.

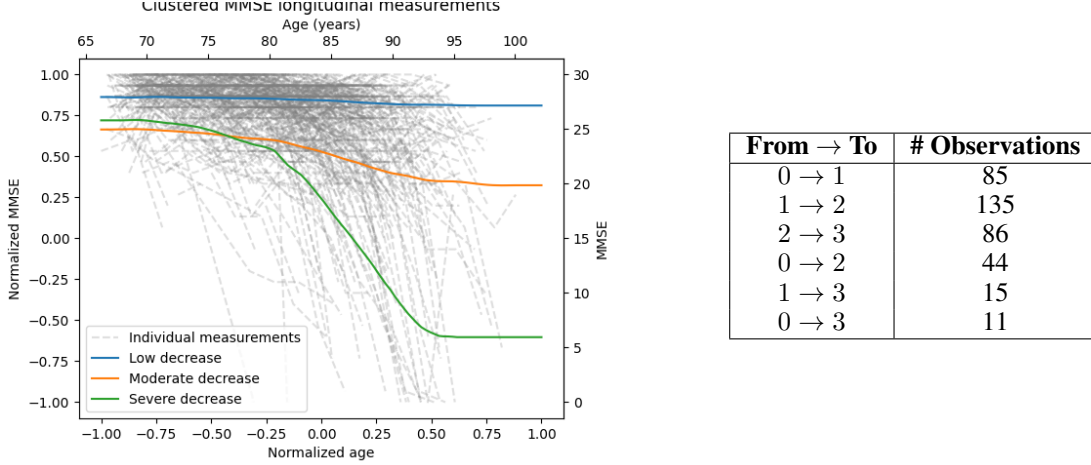


Figure 6: On the left, each light gray line represents one of the 500 observed sequences of (normalized) MMSE scores, and three functional clusters are also represented; on the right, observed transitions between states.

Therefore, the chosen transition graph is as follows, and only allows monotonic transitions from a lower level of physical dependency to a higher one.

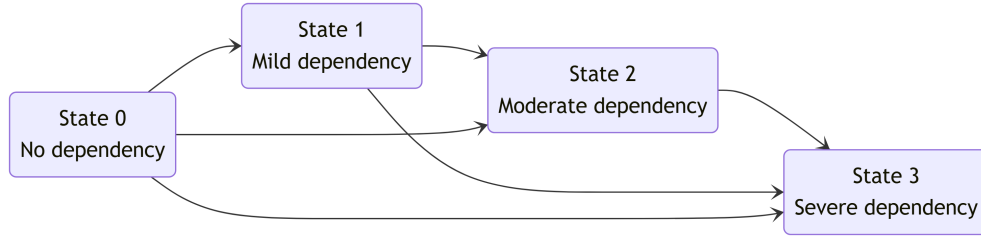


Figure 7: Schematic representation of the four ordered HIER states, with possible transitions. HIER quantifies physical dependency (0: no dependency to 3: severe dependency) [Proust-Lima et al., 2017].

We consider an exponential baseline hazard model for transitions, with a *clock reset* specification. The parameters of these exponential baseline hazards are jointly optimized with the model parameters by passing a list of `base_hazard.parameters()` to the extra argument of the `ModelParams` constructor. Moreover, the `betas` parameter group uses a single shared parameter for all transitions, imposing a particular substructure on the model, where $\forall(k, k') \in E, \beta^{k'|k} = \beta$. This is achieved by passing the same tensor for every transition:

```
# Will be cloned before fitting sharing same memory address
shared_beta = torch.zeros(2)
init_betas = {key: shared_beta for key in buckets}
```

In the light of Figure 6, the regression and link functions were taken to be scaled and shifted hyperbolic tangents such that the normalized MMSE score always has a value of 1 when $t \rightarrow -\infty$ and is not increasing:

$$\begin{cases} h(t, \psi) = g(t, \psi) = \psi_1 \tanh\left(\frac{\psi_3 - t}{\psi_2}\right) + (1 - \psi_1) \\ \psi = \begin{pmatrix} \sigma(\gamma_1 + b_1) \\ \exp(\gamma_2 + b_2) \\ \gamma_3 + b_3 \end{pmatrix}, \quad \sigma(z) = \frac{1}{1 + e^{-z}} \end{cases}$$

The function f is therefore implemented as:

```
def f(
    gamma: torch.Tensor | None,
    x: torch.Tensor | None,
    b: torch.Tensor,
):
    psi = cast(torch.Tensor, gamma) + b
    psi[..., 0] = psi[..., 0].sigmoid()
    psi[..., 1] = psi[..., 1].exp()
    return psi
```

Please note that type hints are not mandatory, but help with code completion and static type checking; this is the right signature as covariates and populational parameters are passed as optional arguments that may be None.

At last, an individual covariates correspond to a pair of binary variables indicating wheter or not the individual has a diploma and is a male ($\mathbb{1}_{\text{CEP}}(i), \mathbb{1}_{\text{male}}(i) \in \{0, 1\}^2$).

8.3 Results

8.3.1 Inference

Fitting was performed using 80% of the data, with the remaining 20% used for testing. With no *a priori* for the covariance matrices, the full option was selected. The model was fitted using the Fit class with a learning rate of 0.5 and the Adam optimizer. The optimization process was run for until convergence based on the same criterion as the simulation study with a lower tolerance of 5%, which was obtained after a little less than 500 iterations with 10 parallel chains.

Before displaying the fitted parameters, the Fisher Information Matrix was computed with the ComputeFIM Job factory implementing the method described in Baey et al. [2023], directly taking into account shared parameters:

```
from jmstate.jobs import ComputeFIM

# Returns the metrics for the FIM, and stores them internally
model.do( # Uses fitting data if no data is provided
    job_factories=ComputeFIM(), # No need to wrap in a list
    max_iterations=100, # Number of samples is n_chains * 100
)
```

The standard errors were then computed as the square root of the diagonal of the inverse of the Fisher Information Matrix with the model.stdeerror property. The results are summarized in Table 2.

Parameter	Inferred value	Standard error	Parameter	Inferred value	Standard error
γ_1	2.6898	0.0561	γ_2	0.2626	0.0137
γ_3	1.3485	0.0025	$\bar{Q}_1$	-0.5428	0.0429
$\bar{Q}_2$	0.8452	0.1068	$\bar{Q}_3$	0.5970	0.0448
$\bar{Q}_4$	0.0041	0.1309	$\bar{Q}_5$	-2.0407	0.1149
$\bar{Q}_6$	0.2357	0.0458	$\bar{R}$	2.1921	0.0135
$\alpha^{3 1}$	-4.1011	1.1296	$\alpha^{1 0}$	0.5156	2.5595
$\alpha^{2 1}$	-1.5068	1.6333	$\alpha^{2 0}$	-2.9875	2.3893
$\alpha^{3 2}$	-2.7624	0.9352	$\alpha^{3 0}$	-6.2268	4.0880
β_1	-0.0267	0.5393	β_2	0.1223	0.6959

Table 2: Inferred parameters and their estimated standard deviations (up to 4 decimal places).

Notably, the estimated standard errors are consistent with the number of observations per transition, with larger standard errors for transitions with fewer observations. Therefore, parameters with too few observations should be interpreted with caution, if at all. The linear covariate parameters also exhibit large standard errors, which indicates the coefficients are not significative in this setting.

The association parameters α being negative, we can infer that cognitive decline is associated with an increase in physical dependency.

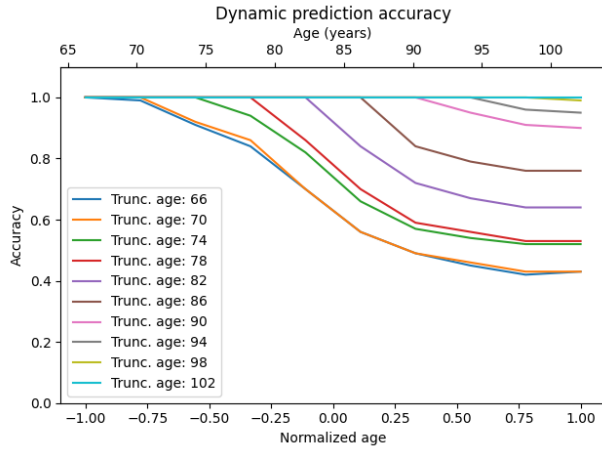
8.3.2 Dynamic Prediction

Using the $n_{\text{test}} = 100$ individuals not included in the model fitting, we performed dynamic prediction with the PredictTrajectories Job. For each individual, longitudinal measurements and trajectories were truncated at various times t , and predictions were made for future time points $u \geq t$ beyond the truncation. Accuracy was then assessed by comparing the most likely predicted state with the true state at each corresponding future time point u , as illustrated in Figure 8.

Formally, the prediction function ξ_u is defined by Example 1, and the accuracy measure at time t for predicted states $(s_i)_i$ is defined as:

$$\text{accuracy}(u) := \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \mathbb{1}_{\xi_{u \wedge C_i}(\tau_i, \kappa_i, \mathcal{T}_i^*) = s_i}.$$

Although $\mathcal{T}_i^*$ is unknown, the quantity $\xi_{u \wedge C_i}(\tau_i, \kappa_i, \mathcal{T}_i^*)$ may be computed based on the censored trajectories. However, incorporating censoring times into prediction functions relies on information that is, in principle, not available at prediction time. Nevertheless, this inclusion is necessary to ensure fairness in accuracy metrics, since individuals who are not censored are generally more likely to have progressed to more advanced states of physical dependency than those who are censored.



Here, the initial drops correspond to the truncation times, before which the accuracy is always 100%.

As expected from dynamic predictions, the accuracy increases as more data is available, therefore when the truncation time grows. We also observe that the longer the prediction horizon, the lower the accuracy of the predictions.

Figure 8: Accuracy of the predicted state at each time point with respect to the true state for different truncation times.

9 Conclusion

We presented `jmstate`, a Python package for joint modeling of nonlinear longitudinal biomarkers and multi-state survival processes. The framework unifies both components on arbitrary directed graphs, extending classical models to Markov and semi-Markov settings and supporting nonlinear submodels. Inference is implemented with scalable stochastic gradient methods.

Simulation studies confirmed robust parameter recovery, and an application to the PAQUID cohort highlighted the ability to capture the interplay between cognitive decline and physical dependency.

Since the model may potentially involve many transitions and therefore a large number of parameters, a strategy of parameter sharing across transitions could help reduce the overall number of parameters. This is especially beneficial in settings with limited data for certain transitions, or when biological or clinical knowledge suggests similar effects across multiple transitions. Parameter sharing improves statistical efficiency, reduces overfitting, and facilitates model interpretability.

Future directions include a spline-based parametrization of baseline hazards to allow for nonparametric estimation. This approach is particularly useful when the true baseline hazard is expected to be complex or non-monotonic, or when parametric forms (such as exponential or Weibull) may be too restrictive and lead to model misspecification.

Another promising direction is the development of efficient visualization tools for multi-state trajectories and dynamic predictions. Interactive representations of individual event histories, transition probabilities, and longitudinal biomarker evolution would greatly facilitate model interpretation and communication of results to applied researchers.

Funding

This work was partially funded by the Stat4Plant project ANR-20-CE45-0012.

Acknowledgments

The authors would like to thank Jean-Benoist Leger for helpful discussions.

A Correctness of Algorithm 1

Proof. First consider the case $t_i^{\text{surv}} = -\infty$. Let $t > T_{i\ell_i}$:

$$\begin{aligned} \mathbb{P}(T_{i(\ell_i+1)} > t \mid T_{i\ell_i}, S_{i\ell_i}) &= \mathbb{P}\left(\bigcap_{s: (E_{i(\ell_i-1)}, s) \in E} T_{i(\ell_i+1)}^s > t \mid T_{i\ell_i}, S_{i\ell_i}\right), \\ &= \prod_{s: (S_{i\ell_i}, s) \in E} \mathbb{P}(T_{i(\ell_i+1)}^s > t \mid T_{i\ell_i}, S_{i\ell_i}), \\ &= \exp\left(-\sum_{s: (S_{i\ell_i}, s) \in E} \Lambda_i^{s|S_{i(\ell_i-1)}}(t \mid \theta, X_i, T_{i\ell_i}, b_i)\right). \end{aligned}$$

Furthermore:

$$\begin{aligned} &\mathbb{P}(S_{i(\ell_i+1)} \mid T_{i(\ell_i+1)} = t, T_{i\ell_i}, S_{i\ell_i}) \\ &\propto p(\{T_{i(\ell_i+1)}^{S_{i(\ell_i+1)}} = t\} \cap \bigcap_{\substack{s: (S_{i\ell_i}, s) \in E, \\ s \neq S_{i(\ell_i+1)}}} \{T_{i(\ell_i+1)}^s \geq t\} \mid T_{i\ell_i}, S_{i\ell_i}), \\ &= \lambda_i^{S_{i(\ell_i+1)}|S_{i\ell_i}}(t \mid \theta, X_i, T_{i\ell_i}, b_i) \exp\left(-\sum_{s: (S_{i\ell_i}, s) \in E} \Lambda_i^{s|S_{i\ell_i}}(t \mid \theta, X_i, T_{i\ell_i}, b_i)\right). \end{aligned}$$

Combining both, we retrieve the same joint density as in formula 3.3.

If $t_i^{\text{surv}} > -\infty$, using the Chasles relation 3.3, one can also check that both densities are equal. Note that the conditioning only affects the first transition, as the next time $\forall \ell_i > 0$, $T_{i\ell_i} \geq t_i^{\text{surv}} \implies \forall \ell_i > 0$, $T_{i\ell_i} \vee t_i^{\text{surv}} = T_{i\ell_i}$. $\square$

B Expression of the Semi-Markov likelihood

Proof. The proof is short and relies on the fact that, using the Semi-Markovian assumption 6:

$$\begin{aligned} p((T_{i(l+1)}, S_{i(l+1)})_{0 \leq l \leq m_i-1} \mid \theta, X_i, b_i) &= \prod_{l=0}^{m_i-1} p((T_{i(l+1)}, S_{i(l+1)}) \mid \theta, X_i, (T_{il'}, S_{il'})_{l' \leq l}, b_i), \\ &= \prod_{l=0}^{m_i-1} p((T_{i(l+1)}, S_{i(l+1)}) \mid \theta, X_i, (T_{il}, S_{il}), b_i). \end{aligned}$$

On the other hand, the probability that no additional event is observed between T_{im_i} and C_i is 1 if the last state reached is absorbing, and otherwise:

$$\mathbb{P}(T_{i(m_i+1)} > C_i \mid \theta, X_i, (T_{im_i}, S_{im_i}), b_i) = \exp\left(-\sum_{s: (S_{im_i}, s) \in E} \Lambda_i^{s|S_{im_i}}(C_i \mid \theta, X_i, T_{im_i}, b_i)\right),$$

and so we recover the formula cited above with the convention that an empty sum is 0. $\square$

References

- Per Kragh Andersen and Niels Keiding. Multi-state models for event history analysis. *Statistical Methods in Medical Research*, 8(2):127–153, 1999. doi: 10.1177/096228029900800202.
- Eleni-Rosalina Andrinopoulou, Paul HC Eilers, Johanna JM Takkenberg, and Dimitris Rizopoulos. Improved dynamic predictions from joint models of longitudinal and survival data with time-varying effects using p-splines. *Biometrics*, 74(2):685–693, 2018.
- Azam Asanjarani, Benoit Lique, and Yoni Nazarathy. Estimation of semi-markov multi-state models: A comparison of the sojourn times and transition intensities approaches. *The International Journal of Biostatistics*, 18(1):243–262, 2022.

- Charlotte Baey, Maud Delattre, Estelle Kuhn, Jean-Benoist Leger, and Sarah Lemler. Efficient preconditioned stochastic gradient descent for estimation in latent variable models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 1430–1453. PMLR, 2023.
- Antoine Caillebotte, Estelle Kuhn, and Sarah Lemler. Estimation and variable selection in high dimension in nonlinear mixed-effects models. *arXiv preprint arXiv:2503.20401*, 2025. doi: 10.48550/arXiv.2503.20401. Version 2 (Aug 2025).
- Daniel Commenges, Philippe Joly, Cécile Proust-Lima, and Benoit Liqueur. Likelihood-based approaches to illness–death models. *Biostatistics*, 7(4):544–556, 2006.
- Marie Davidian and David M. Giltinan. Nonlinear models for repeated measurement data: An overview and update. *Journal of Agricultural, Biological, and Environmental Statistics*, 8:387–419, 2003. doi: 10.1198/1085711032697.
- Liesbeth C. de Wreede, Marta Fiocco, and Hein Putter. The mstate package for estimation and prediction in non- and semi-parametric multi-state and competing risks models. *Computer Methods and Programs in Biomedicine*, 99(3): 261–274, 2010. doi: 10.1016/j.cmpb.2010.01.001.
- Wenjing Dong, Amy H. Herring, and David B. Dunson. Non-markovian modeling of sleep patterns. *Biostatistics*, 9(4): 806–820, 2008.
- Timothy Dozat. Incorporating nesterov momentum into adam. <https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ>, 2016.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.
- Loic Ferrer, Virginie Rondeau, James Dignam, Tom Pickles, Hélène Jacqmin-Gadda, and Cécile Proust-Lima. Joint modelling of longitudinal and multi-state processes: Application to clinical progressions in prostate cancer. *Statistics in Medicine*, 35(22):3933–3948, 2016.
- Marta Fiocco, Hein Putter, and Hans C. van Houwelingen. Modeling non-exponential transition times in multi-state models. *Statistics in Medicine*, 27(30):5566–5581, 2008.
- Haim Frydman. Estimation in the cox model with misclassified covariates using multistate semi-markov models. *Biometrics*, 61(3):847–854, 2005.
- Andrew Gelman, Walter R. Gilks, and Gareth O. Roberts. Weak convergence and optimal scaling of random walk metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997. doi: 10.1214/aoap/1034625254.
- Daniel T Gillespie. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.*, 58(1):35–55, 2007.
- Joseph G. Ibrahim, Haitao Chu, and Liddy M. Chen. Basic concepts and methods for joint models of longitudinal and survival data. *Journal of Clinical Oncology*, 28(16):2796–2801, 2010.
- Christopher H. Jackson. Multi-state models for panel data: The msm package for R. *Journal of Statistical Software*, 38(8):1–29, 2011. doi: 10.18637/jss.v038.i08.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Agnieszka Król, Audrey Mauguen, Yassin Mazroui, Alexandre Laurent, Stefan Michiels, and Virginie Rondeau. Tutorial in joint modeling and prediction: A statistical software for correlated longitudinal outcomes, recurrent events and a terminal event. *Journal of Statistical Software*, 81(3):1–52, 2017. doi: 10.18637/jss.v081.i03.
- Estelle Kuhn and Marc Lavielle. Coupling a stochastic approximation version of em with an mcmc procedure. *ESAIM: Probability and Statistics*, 8:115–131, 2004.
- Luc Letenneur, Daniel Commenges, Jean-François Dartigues, and Pascale Barberger-Gateau. Incidence of dementia and Alzheimer’s disease in elderly community residents of southwestern france. *International Journal of Epidemiology*, 23(6):1256–1261, 1994.
- Frank G. Lether. On the construction of gauss–legendre quadrature rules. *Journal of Computational and Applied Mathematics*, 4(1):47–52, 1978. doi: 10.1016/0771-050X(78)90019-0.
- Nikolaos Limnios and Gabriel Oprisan. *Semi-Markov Processes and Reliability*. Statistics for Industry and Technology. Birkhäuser, 2001.
- Leif Erik Lovblom, Laurent Briollais, Bruce A. Perkins, and George Tomlinson. Modeling multiple correlated end-organ disease trajectories: A tutorial for multistate and joint models with applications in diabetes complications. *Statistics in Medicine*, 43(5):1048–1082, 2024. doi: 10.1002/sim.9984.
- Elisa Luciano and Vladimir Vasiliev. Multistate semi-markov models in finance. *Decisions in Economics and Finance*, 29(1):1–22, 2006.

- Katya Adrienne Lopes Mauff. *Extensions to Joint Models for Longitudinal and Survival Outcomes*. PhD thesis, Erasmus University Rotterdam, Rotterdam, The Netherlands, January 2023. URL https://pure.eur.nl/files/76933469/thesismauff2022_63a06b0c6063e.pdf.
- Van Tuan Nguyen, Adeline Fermanian, Agathe Guilloux, Antoine Barbieri, Sarah Zohar, Anne-Sophie Jannot, and Simon Bussy. An efficient joint model for high dimensional longitudinal and survival data via generic association features. *Biometrics*, 80(4), 2024. In press.
- Giasemi Papageorgiou, Kristina Mauff, Anastasios Tomer, and Dimitris Rizopoulos. An overview of joint modeling of time-to-event and longitudinal outcomes. *Annual Review of Statistics and Its Application*, 6:223–240, 2019. doi: 10.1146/annurev-statistics-030718-104209.
- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS 2017 Workshop on Autodiff*, 2017. arXiv preprint.
- 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. URL <https://github.com/graemeleehickey/joiner/>. R package version 1.2.8.
- Cécile Proust-Lima, Viviane Philipps, and Benoît Liqueur. Estimation of extended mixed models using latent classes and latent processes: The R package lcmm. *Journal of Statistical Software*, 78(2):1–56, 2017. doi: 10.18637/jss.v078.i02.
- Hein Putter, Marta Fiocco, and Ronald B. Geskus. Tutorial in biostatistics: Competing risks and multi-state models. *Statistics in Medicine*, 26(11):2389–2430, 2007. doi: 10.1002/sim.2712.
- Dimitris Rizopoulos. Dynamic predictions and prospective accuracy in joint models for longitudinal and time-to-event data. *Biometrics*, 67(3):819–829, 2011.
- Dimitris Rizopoulos. *Joint Models for Longitudinal and Time-to-Event Data: With Applications in R*. Chapman and Hall/CRC, 2012.
- 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):1–46, 2016. doi: 10.18637/jss.v072.i07.
- Dimitris Rizopoulos. *Jmbayes: Joint Models for Longitudinal and Time-to-Event Data*, 2020. R package Jmbayes, CRAN.
- Dimitris Rizopoulos, Geert Verbeke, and Geert Molenberghs. Shared parameter models under random effects misspecification. *Biometrika*, 95(1):63–74, 2008. doi: 10.1093/biomet/asm082.
- Dimitris Rizopoulos, Pedro Miranda Afonso, and Grigorios Papageorgiou. *Jmbayes2: Extended Joint Models for Longitudinal and Time-to-Event Data*, 2024. URL <https://CRAN.R-project.org/package=Jmbayes2>. R package version 0.5-2.
- Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3): 400–407, 1951.
- Christian Roever, Robert Meyer, and Ursula Kuchler. Statistical modelling of sleep stages: A semi-markov model approach. *Computational Statistics & Data Analysis*, 54(8):2039–2051, 2010.
- Paula Williamson, Ruwanthi Kolamunnage-Dona, Pete Philipson, and Anthony G. Marson. Joint modelling of longitudinal and competing risks data. *Statistics in Medicine*, 27:6426–6438, 2008.
- Russ Wolfinger. Laplace’s approximation for nonlinear mixed models. *Biometrika*, 80(4):791–795, 1993.
- Sean Yiu, Vernon T. Farewell, and Brian D.M. Tom. Clustered multistate models with observation level random effects, mover–stayer effects and dynamic covariates: Modelling transition intensities and sojourn times in a study of psoriatic arthritis. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 67(2):481–500, 2018.