SOFTWARE

joineRML: A joint model and software package for time-to-event and multivariate longitudinal outcomes

Graeme L Hickey[1], Pete Philipson[2], Andrea Jorgensen[1] and Ruwanthi Kolamunnage-Dona[1*]

*Correspondence: ruwanthi.kolamunnage-dona@liverpool.ac.uk
[1]Department of Biostatistics, Institute of Translational Medicine, University of Liverpool, Waterhouse Building, 1-5 Brownlow Street, L69 3GL Liverpool, UK
Full list of author information is available at the end of the article

Abstract

Background: Joint modelling of longitudinal and time-to-event outcomes has received considerable attention over recent years. Commensurate with this has been a rise in statistical software options for fitting these models. However, these tools have generally been limited to a single longitudinal outcome. Here, we describe the classical joint model to the case of *multiple* longitudinal outcomes, propose a practical algorithm for fitting the models, and demonstrate how to fit the models using a new package for the statistical software platform R, `joineRML`.

Results: A multivariate linear mixed sub-model is specified for the longitudinal outcomes, and a Cox proportional hazards regression model with time-varying covariates is specified for the event time sub-model. The association between models is captured through a zero-mean multivariate latent Gaussian process. The models are fitted using a Monte Carlo Expectation-Maximisation algorithm, and inferences are based on approximate standard errors from the empirical profile information matrix, which are contrasted to an alternative bootstrap estimation approach. We illustrate the model and software on a real data example for patients with primary biliary cirrhosis with three repeatedly measured biomarkers.

Conclusions: An open-source software package capable of fitting multivariate joint models is available. The underlying algorithm and source code makes use of several methods to increase computational speed.

Keywords: Joint modelling; Longitudinal data; Multivariate data; Time-to-event data; Software

Background

In many clinical studies, subjects are followed-up repeatedly and response data collected. For example, routine blood tests might be performed at each follow-up clinic appointment for patients enrolled in a randomized drug trial, and biomarker measurements recorded. An event time is also usually of interest, for example time of death or study drop-out. It has been repeatedly shown elsewhere that if the longitudinal and event-time outcomes are correlated, then modelling the two outcome processes separately, for example using linear mixed models and Cox regression models, can lead to biased effect size estimates [1]. The same criticism has also been levelled at the application of so-called two-stage models [2]. The motivation for using joint models can be broadly separated into interest in drawing inference about (1) the time-to-event process whilst adjusting for the intermittently measured (and potentially error-prone) longitudinal outcomes, and (2) the longitudinal data process whilst adjusting for a potentially informative drop-out mechanism [3]. The literature on joint modelling is extensive, with excellent reviews given by Tsiatis and Davidian [4], Gould *et al.* [5], and the monologue by Rizopoulos [6].

Joint modelling has until recently been predominated by modelling a single longitudinal outcome together with a solitary event time outcome; herein referred to as *univariate joint modelling*. Commensurate with this methodological research has been an increase in wide-ranging clinical applications (e.g. [7]). Recent innovations in the field of joint models have included the incorporation of multivariate longitudinal data [8], competing risks data [9, 10], recurrent events data [11], multivariate time-to-event data [12, 13], non-continuous repeated measurements (e.g. count, binary, ordinal, and censored data) [14], non-normally and non-parametrically distributed random effects [15], alternative estimation methodologies (e.g. Bayesian fitting and conditional estimating equations) [16, 17], and different association structures [18]. In this article, we specifically focus on the first innovation: multivariate longitudinal data. In this situation, we assume that multiple longitudinal outcomes are measured on each subject, which can be unbalanced and measured at different times for each subject.

Despite the inherently obvious benefits of harnessing all data in a single model or the published research on the topic of joint models for multivariate longitudinal data, a recent literature review by Hickey *et al.* [19] identified that publicly available software for fitting such models was lacking, which has translated into limited uptake by biomedical researchers. In this article we present the classical joint model described by Henderson *et al.* [3] extended to the case of multiple longitudinal outcomes. An algorithm proposed by Lin *et al.* [20] is used to fit the model, augmented by techniques to reduce the computational fitting time, including a quasi-Newton update approach, variance reduction method, and dynamic Monte Carlo updates. This algorithm is encoded into a R sofware package–`joineRML`. A simulation analysis and real-world data example are used to demonstrate the accuracy of the algorithm and the software, respectively.

Implementation

As a prelude to the introduction and demonstration of the newly introduced software package, in the following section we describe the underlying model formulation and model fitting methodology.

Model

For each subject $i = 1, \ldots, n$, $\boldsymbol{y}_i = (\boldsymbol{y}_{i1}^\top, \ldots, \boldsymbol{y}_{iK}^\top)$ is the K-variate continuous outcome vector, where each $\boldsymbol{y}_{ik}$ denotes an $(n_{ik} \times 1)$-vector of observed longitudinal measurements for the k-th outcome type: $\boldsymbol{y}_{ik} = (y_{i1k}, \ldots, y_{in_{ik}k})^\top$. Each outcome is measured at observed (possibly pre-specified) times t_{ijk} for $j = 1, \ldots, n_{ik}$, which can differ between subjects and outcomes. Additionally, for each subject there is an event time T_i^*, which is subject to right censoring. Therefore, we observe $T_i = \min(T_i^*, C_i)$, where C_i corresponds to a potential censoring time, and the failure indicator δ_i, which is equal to 1 if the failure is observed ($T_i^* \leq C_i$) and 0 otherwise. We assume that both censoring and measurement times are non-informative.

The model we describe is the natural extension of the model proposed by Henderson *et al.* [3] to the case of multivariate longitudinal data. The model posits an unobserved or latent zero-mean $(K+1)$-variate Gaussian process that is realised independently for each subject, $W_i(t) = \left\{W_{1i}^{(1)}(t), \ldots, W_{1i}^{(K)}(t), W_{2i}(t)\right\}$. This latent process subsequently links the separate sub-models via association parameters.

The k-th longitudinal data sub-model is given by

$$y_{ik}(t) = \mu_{ik}(t) + W_{1i}^{(k)}(t) + \varepsilon_{ik}(t), \tag{1}$$

where $\mu_{ik}(t)$ is the mean response, and $\varepsilon_{ik}(t)$ is the model error term, which we assume to be independent and identically distributed normal with mean 0 and variance σ_k^2. The mean response is specified as a linear model

$$\mu_{ik}(t) = \boldsymbol{x}_{ik}^\top(t)\boldsymbol{\beta}_k, \tag{2}$$

where $\boldsymbol{x}_{ik}(t)$ is a p_k-vector of (possibly) time-varying covariates with corresponding fixed effect terms $\boldsymbol{\beta}_k$. $W_{1i}^{(k)}(t)$ is specified as

$$W_{1i}^{(k)}(t) = \boldsymbol{z}_{ik}^\top(t)\boldsymbol{b}_{ik}, \tag{3}$$

where $\boldsymbol{z}_{ik}(t)$ is an r_k-vector of (possibly) time-varying covariates with corresponding subject-and-outcome random effect terms $\boldsymbol{b}_{ik}$, which follow a zero-mean multivariate normal distribution with $(r_k \times r_k)$-variance-covariance matrix $\boldsymbol{D}_{kk}$. To account for dependence between the different longitudinal outcome outcomes, we let $\text{cov}(\boldsymbol{b}_{ik}, \boldsymbol{b}_{il}) = \boldsymbol{D}_{kl}$ for $k \neq l$. Furthermore, we assume $\varepsilon_{ik}(t)$ and $\boldsymbol{b}_{ik}$ are uncorrelated, and that the censoring times are independent of the random effects. These distributional assumptions together with the model given by (1)–(3) are equivalent to the multivariate extension of the Laird and Ware [21] linear mixed effects model. More flexible specifications of $W_{1i}^{(k)}(t)$ can be used [3], including for example, stationary Gaussian processes. However, we do not consider these cases here owing to the increased computational burden it carries, even for the univariate case.

The sub-model for the time-to-event outcome is given by the hazard model

$$\lambda_i(t) = \lambda_0(t) \exp\left\{\boldsymbol{v}_i^\top(t)\boldsymbol{\gamma}_v + W_{2i}(t)\right\},$$

where $\lambda_0(\cdot)$ is an unspecified baseline hazard, and $\boldsymbol{v}_i(t)$ is a q-vector of (possibly) time-varying covariates with corresponding fixed effect terms $\boldsymbol{\gamma}_v$. Conditional on $W_i(t)$ and the observed covariate data, the longitudinal and time-to-event data generating processes are conditionally independent. To establish a latent association, we specify $W_{2i}(t)$ as a linear combination of $\left\{W_{1i}^{(1)}(t), \dots, W_{1i}^{(K)}(t)\right\}$:

$$W_{2i}(t) = \sum_{k=1}^{K} \gamma_{yk} W_{1i}^{(k)}(t),$$

where $\boldsymbol{\gamma}_y = (\gamma_{y1}, \dots, \gamma_{yK})$ are the corresponding association parameters. To emphasise the dependence of $W_{2i}(t)$ on the random effects, we explicitly write it as $W_{2i}(t, \boldsymbol{b}_i)$ from here onwards. As per $W_{1i}^{(k)}(t)$, $W_{2i}(t, \boldsymbol{b}_i)$ can also be flexibly extended, for example to include subject-specific frailty effects [3].

Estimation

Likelihood

For each subject i, let $\boldsymbol{X}_i = \bigoplus_{k=1}^K \boldsymbol{X}_{ik}$ and $\boldsymbol{Z}_i = \bigoplus_{k=1}^K \boldsymbol{Z}_{ik}$ be block-diagonal matrices, where $\boldsymbol{X}_{ik} = \left(\boldsymbol{x}_{i1k}^\top, \dots, \boldsymbol{x}_{in_{ik}k}^\top\right)$ is an $(n_{ik} \times p_k)$-design matrix, with the j-th row corresponding to the p_k-vector of covariates measured at time t_{ijk}, and $\bigoplus$ denotes the direct matrix sum. The notation similarly follows for the random effects design matrices, $\boldsymbol{Z}_{ik}$. We denote the error terms by a diagonal matrix $\boldsymbol{\Sigma}_i = \bigoplus_{k=1}^K \sigma_k^2 \boldsymbol{I}_{n_{ik}}$ and write the overall variance-covariance matrix for the random effects as

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{D}_{11} & \cdots & \boldsymbol{D}_{1K} \\ \vdots & \ddots & \vdots \\ \boldsymbol{D}_{1K}^\top & \cdots & \boldsymbol{D}_{KK} \end{pmatrix},$$

where $\boldsymbol{I}_n$ denotes an $n \times n$ identity matrix. We further define $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \dots, \boldsymbol{\beta}_K^\top)^\top$ and $\boldsymbol{b}_i = (\boldsymbol{b}_{i1}^\top, \dots, \boldsymbol{b}_{iK}^\top)^\top$. Hence, we can then rewrite the longitudinal outcome sub-model as

$$\begin{aligned} \boldsymbol{y}_i \,|\, \boldsymbol{b}_i, \boldsymbol{\beta}, \boldsymbol{\Sigma}_i &\sim N(\boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\boldsymbol{b}_i, \boldsymbol{\Sigma}_i), \\ \text{with } \boldsymbol{b}_i \,|\, \boldsymbol{D} &\sim N(\boldsymbol{0}, \boldsymbol{D}). \end{aligned}$$

For the estimation, we will assume that the covariates in the time-to-event sub-model are time-independent and known at baseline, i.e. $\boldsymbol{v}_i \equiv \boldsymbol{v}_i(0)$. Extensions of the estimation procedure for time-varying covariates are outlined elsewhere [6, p. 115]. The *observed* data likelihood for the joint outcome is given by

$$\prod_{i=1}^{n} \left(\int_{-\infty}^{\infty} f(\boldsymbol{y}_i \,|\, \boldsymbol{b}_i, \boldsymbol{\theta}) f(T_i, \delta_i \,|\, \boldsymbol{b}_i, \boldsymbol{\theta}) f(\boldsymbol{b}_i \,|\, \boldsymbol{\theta}) d\boldsymbol{b}_i \right), \qquad (4)$$

where $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \text{vech}(\boldsymbol{D}), \sigma_1^2, \dots, \sigma_K^2, \lambda_0(t), \boldsymbol{\gamma}_v^\top, \boldsymbol{\gamma}_y^\top)$ is the collection of unknown parameters that we want to estimate, with vech$(\boldsymbol{D})$ denoting the half-vectorisation operator that returns the vector of lower-triangular elements of matrix $\boldsymbol{D}$.

As noted by Henderson *et al.* [3], the observed data likelihood can be calculated by rewriting it as

$$\prod_{i=1}^{n} f(\boldsymbol{y}_i \,|\, \boldsymbol{\theta}) \left(\int_{-\infty}^{\infty} f(T_i, \delta_i \,|\, \boldsymbol{b}_i, \boldsymbol{\theta}) f(\boldsymbol{b}_i \,|\, \boldsymbol{y}_i, \boldsymbol{\theta}) d\boldsymbol{b}_i \right),$$

where the marginal distribution $f(\boldsymbol{y}_i \,|\, \boldsymbol{\theta})$ is a multivariate normal density with mean $\boldsymbol{X}_i\boldsymbol{\beta}$ and variance-covariance matrix $\boldsymbol{\Sigma}_i + \boldsymbol{Z}_i\boldsymbol{D}\boldsymbol{Z}_i^\top$, and $f(\boldsymbol{b}_i \,|\, \boldsymbol{y}_i, \boldsymbol{\theta})$ is given by (6).

MCEM algorithm

We determine maximum likelihood estimates of the parameters $\boldsymbol{\theta}$ using the Monte Carlo Expectation Maximisation (MCEM) algorithm [22], by treating the random effects $\boldsymbol{b}_i$ as missing data. This is effectively the same as the conventional Expectation-Maximisation (EM) algorithm, as used by Wulfsohn and Tsiatis [23] and Ratcliffe *et al.* [24] in the context of fitting univariate data joint models, except the E-step exploits a Monte Carlo (MC) integration routine as opposed to Gaussian quadrature methods, which we expect to be beneficial when the dimension of random effects becomes large.

Starting from an initial estimate of the parameters, $\hat{\boldsymbol{\theta}}^{(0)}$, the procedure involves iterating between the following two steps until convergence is achieved.

1 *E-step.* At the $(m+1)$-th iteration, we compute the expected log-likelihood of the *complete* data conditional on the *observed* data and the current estimate of the parameters,

$$
\begin{aligned}
Q(\boldsymbol{\theta} \mid \hat{\boldsymbol{\theta}}^{(m)}) &= \sum_{i=1}^{n} \mathbb{E}\Big\{ \log f(\boldsymbol{y}_i, T_i, \delta_i, \boldsymbol{b}_i \mid \boldsymbol{\theta}) \Big\} \\
&= \sum_{i=1}^{n} \int_{-\infty}^{\infty} \Big\{ \log f(\boldsymbol{y}_i, T_i, \delta_i, \boldsymbol{b}_i \mid \boldsymbol{\theta}) \Big\} f(\boldsymbol{b}_i \mid T_i, \delta_i, \boldsymbol{y}_i; \hat{\boldsymbol{\theta}}^{(m)}) d\boldsymbol{b}_i.
\end{aligned}
$$

Here, the complete-data likelihood contribution for subject i is given by the integrand of (4).

2 *M-step.* We maximise $Q(\boldsymbol{\theta} \mid \hat{\boldsymbol{\theta}}^{(m)})$ with respect to $\boldsymbol{\theta}$. Namely, we set

$$
\hat{\boldsymbol{\theta}}^{(m+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \hat{\boldsymbol{\theta}}^{(m)}).
$$

The M-step estimators naturally follow from Wulfsohn and Tsiatis [23] and Lin *et al.* [20]. Maximizers for all parameters except $\boldsymbol{\gamma}_v$ and $\boldsymbol{\gamma}_y$ are available in closed-form; algebraic details are presented in **Additional file 1**. The parameters $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_{\boldsymbol{v}}^\top, \boldsymbol{\gamma}_y^\top)^\top$ are jointly updated using a one-step Newton-Raphson algorithm as

$$
\hat{\boldsymbol{\gamma}}^{(m+1)} = \hat{\boldsymbol{\gamma}}^{(m)} + I\left(\hat{\boldsymbol{\gamma}}^{(m)}\right)^{-1} S\left(\hat{\boldsymbol{\gamma}}^{(m)}\right),
$$

where $\hat{\boldsymbol{\gamma}}^{(m)}$ denotes the value of $\boldsymbol{\gamma}$ at the current iteration, $S\left(\hat{\boldsymbol{\gamma}}^{(m)}\right)$ is the corresponding score, and $I\left(\hat{\boldsymbol{\gamma}}^{(m)}\right)$ is the observed information matrix, which is equal to the derivative of the negative score. Further details of this update are given in **Additional file 1**. The M-step for $\boldsymbol{\gamma}$ is computationally expensive to evaluate. Therefore, we also propose a quasi-Newton one-step update by approximating $I\left(\hat{\boldsymbol{\gamma}}^{(m)}\right)$ by an empirical information matrix for $\boldsymbol{\gamma}$, which can be considered an analogue of the Gauss-Newton method [25, p. 8]. To further compensate for this approximation, we also use a nominal step-size of 0.5 rather than 1, which is used when exactly calculating $I(\boldsymbol{\gamma})$.

The M-step involves terms of the form $\mathbb{E}\left[h(\boldsymbol{b}_i) \mid T_i, \delta_i, \boldsymbol{y}_i; \hat{\boldsymbol{\theta}}\right]$, for known functions $h(\cdot)$. The conditional expectation of a function of the random effects can be written as

$$\mathbb{E}\left[h(\boldsymbol{b}_i) \mid T_i, \delta_i, \boldsymbol{y}_i; \hat{\boldsymbol{\theta}}\right] = \frac{\int_{-\infty}^{\infty} h(\boldsymbol{b}_i) f(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \hat{\boldsymbol{\theta}}) f(T_i, \delta_i \mid \boldsymbol{b}_i; \hat{\boldsymbol{\theta}}) d\boldsymbol{b}_i}{\int_{-\infty}^{\infty} f(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \hat{\boldsymbol{\theta}}) f(T_i, \delta_i \mid \boldsymbol{b}_i; \hat{\boldsymbol{\theta}}) d\boldsymbol{b}_i}, \quad (5)$$

where $f(T_i, \delta_i \mid \boldsymbol{b}_i; \hat{\boldsymbol{\theta}})$ is given by

$$f(T_i, \delta_i \mid \boldsymbol{b}_i; \boldsymbol{\theta}) = \left[\lambda_0(T_i) \exp\left\{\boldsymbol{v}_i^\top \boldsymbol{\gamma}_v + W_{2i}(T_i, \boldsymbol{b}_i)\right\}\right]^{\delta_i} \times \exp\left\{-\int_0^{T_i} \lambda_0(u) \exp\left\{\boldsymbol{v}_i^\top \boldsymbol{\gamma}_v + W_{2i}(u, \boldsymbol{b}_i)\right\} du\right\}$$

and $f(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \hat{\boldsymbol{\theta}})$ is calculated from multivariate normal distribution theory as

$$\boldsymbol{b}_i \mid \boldsymbol{y}_i, \boldsymbol{\theta} \sim N\left(\boldsymbol{A}_i \left\{\boldsymbol{Z}_i^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{X}_i \beta)\right\}, \boldsymbol{A}_i\right), \quad (6)$$

with $\boldsymbol{A}_i = \left(\boldsymbol{Z}_i^\top \boldsymbol{\Sigma}_i^{-1} \boldsymbol{Z}_i + \boldsymbol{D}^{-1}\right)^{-1}$. As this becomes computationally expensive using Gaussian quadrature commensurate with increasing dimension of $\boldsymbol{b}_i$, we estimate the integrals by MC sampling such that the expectation is approximated by the ratio of the sample means for $h(\boldsymbol{b}_i) f(T_i, \delta_i \mid \boldsymbol{b}_i; \hat{\boldsymbol{\theta}})$ and $f(T_i, \delta_i \mid \boldsymbol{b}_i; \hat{\boldsymbol{\theta}})$ evaluated at each MC draw. Furthermore, we use antithetic simulation for variance reduction in the MC integration. Instead of directly sampling from (6), we sample $\boldsymbol{\Omega} \sim N(0, \boldsymbol{I}_r)$ and obtain the *pairs*

$$\boldsymbol{A}_i \left\{\boldsymbol{Z}_i^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{X}_i \beta)\right\} \pm \boldsymbol{C}_i \boldsymbol{\Omega},$$

where $\boldsymbol{C}_i$ is the Cholesky decomposition of $\boldsymbol{A}_i$ such that $\boldsymbol{C}_i \boldsymbol{C}_i^\top = \boldsymbol{A}_i$. Therefore we only need to draw $N/2$ samples using this approach, and by virtue of the negative correlation between the pairs, it leads to a smaller variance in the sample means taken in the approximation than would be obtained from N independent simulations. The choice of N is described below.

Initial values

The EM algorithm requires that initial parameters are specified, namely $\hat{\boldsymbol{\theta}}^{(0)}$. By choosing values close to the maximizer, the number of iterations required to reach convergence should be reduced.

For the time-to-event sub-model, a quasi-two-stage model is fitted when the measurement times are balanced, i.e. when $t_{ijk} = t_{ij}\ \forall k$. That is, we fit *separate* LMMs for each longitudinal outcome as per (1), ignoring the correlation between different outcomes. This is straightforward to implement using standard software, in particular using `lme()` and `coxph()` from the R packages `nlme` [26] and `survival` [27], respectively. From the fitted models, the best linear unbiased predictions (BLUPs) of the separate model random effects are used to estimate each $W_{1i}^{(k)}(t)$ function. These estimates are then included as time-varying covariates in a Cox regression model, alongside any other fixed effect covariates, which can be straightforwardly fitted using standard software. In the situation that the data are not balanced, i.e. when $t_{ijk} \neq t_{ij} \; \forall k$, then we fit a standard Cox proportional hazards regression model to estimate $\boldsymbol{\gamma}_v$ and set $\gamma_{yk} = 0 \; \forall k$.

For the longitudinal data sub-model, when $K > 1$ we first find the maximum likelihood estimate of $\{\boldsymbol{\beta}, \text{vech}(\boldsymbol{D}), \sigma_1^2, \dots, \sigma_K^2\}$ by running a separate EM algorithm for the multivariate linear mixed model. Both the E- and M-step updates are available in closed form, and the initial parameters for this EM algorithm are available from the separate LMM fits, with $\boldsymbol{D}$ initialized as block-diagonal. As these are estimated using an EM rather than MCEM algorithm, we can specify a stricter convergence criterion on the estimates.

Convergence and stopping rules

Two standard stopping rules for the deterministic EM algorithm used to declare convergence are the relative and absolute differences, defined as

$$\Delta_{\text{rel}}^{(m+1)} = \max\left\{\frac{|\hat{\boldsymbol{\theta}}^{(m+1)} - \hat{\boldsymbol{\theta}}^{(m)}|}{|\hat{\boldsymbol{\theta}}^{(m)}| + \epsilon_1}\right\} < \epsilon_0, \text{ and} \quad (7)$$

$$\Delta_{\text{abs}}^{(m+1)} = \max\left\{|\hat{\boldsymbol{\theta}}^{(m+1)} - \hat{\boldsymbol{\theta}}^{(m)}|\right\} < \epsilon_2 \quad (8)$$

respectively, for some appropriate choice of ϵ_0, ϵ_1, and ϵ_2, where the maximum is taken over the components of $\boldsymbol{\theta}$. For reference, the R package `JM` [28] implements (7) (in combination with another rule based on relative change in the likelihood), whereas the R package `joineR` [29] implements (8). The relative difference might be unstable about parameters near zero that are subject to MC error. Therefore, the convergence criterion for each parameter might be chosen separately at each EM iteration based on whether the absolute magnitude is below or above some threshold. A similar approach is adopted in the EM algorithms employed by the software package SAS [30, p. 330].

The choice of N and the monitoring of convergence are conflated when applying a MCEM algorithm, and a dynamic approach is required. As noted by [22], it is computationally inefficient to use a large N in the early phase of the algorithm when the parameter estimates are likely to be far from the maximizer. On the flip side, as the parameter estimates approach the maximizer, the stopping rules will fail as the changes in parameter estimates will be swamped by MC error. Therefore, it has been recommended that one increase N as the estimate moves towards the maximizer. Although this might be done subjectively [31] or by pre-specified rules [32], an automated approach is preferable and necessary for a software implementation. Booth and Hobert [33] proposed an update rule based on a confidence ellipsoid for the maximizer at the $(m+1)$-th iteration, calculated using an approximate sandwich estimator for the maximizer, which accounts for the MC error at each iteration. This approach requires additional variance estimation at each iteration, therefore we opt for a simpler approach described by Ripatti *et al.* [34]. Namely, we calculate a coefficient of variation at the $(m+1)$-th iteration as

$$\text{cv}(\Delta_{\text{rel}}^{(m+1)}) = \frac{\text{sd}(\Delta_{\text{rel}}^{(m-1)}, \Delta_{\text{rel}}^{(m)}, \Delta_{\text{rel}}^{(m+1)})}{\text{mean}(\Delta_{\text{rel}}^{(m-1)}, \Delta_{\text{rel}}^{(m)}, \Delta_{\text{rel}}^{(m+1)})},$$

where $\Delta_{\text{rel}}^{(m+1)}$ is given by (7), and $\text{sd}(\cdot)$ and $\text{mean}(\cdot)$ are the sample standard deviation and mean functions, respectively. If $\text{cv}(\Delta_{\text{rel}}^{(m+1)}) > \text{cv}(\Delta_{\text{rel}}^{(m)})$, then $N := N + \lfloor N/\delta \rfloor$, for some small positive integer δ. Typically, we run the MCEM algorithm with a small N (for a fixed number of iterations—a *burn-in*) before implementing this update rule in order to get into the approximately correct parameter region. Appropriate values for other parameters will be application specific, however we have found $\delta = 3$, $N = 100K$ (for $100K$ burn-in iterations), $\epsilon_1 = 0.001$, and $\epsilon_0 = \epsilon_2 = 0.005$ delivers reasonably accurate estimates in many cases, where K was earlier defined as the number of longitudinal outcomes.

As the EM monotonicity property is lost due to the MC integrations in the MCEM algorithm, convergence might be prematurely declared due to stochasticity if the ϵ-values are too large. To reduce the chance of this occurring, we require that the stopping rule is satisfied for 3 consecutive iterations [33, 34]. However, in any case, trace plots should be inspected to confirm convergence is appropriate.

Standard error estimation

Standard error (SE) estimation is usually based on inverting the observed information matrix. When the baseline hazard is unspecified, as is the case here, this presents several challenges. First, $\hat{\lambda}_0(t)$ will generally be a high-dimensional vector, which might lead to numerical difficulties in the inversion of the observed information matrix [6]. Second, the profile likelihood estimates based on the usual observed information matrix approach are known to be underestimated [35]. The reason for this is that the profile estimates are implicit, since the posterior expectations, given by (5), depend on the parameters being estimated, including $\lambda_0(t)$ [6, p. 67].

To overcome these challenges, Hsieh *et al.* [35] recommended to use bootstrap methods to calculate the SEs. However, this approach is computationally expensive. Moreover, despite the purported theoretical advantages, we also note that recently it has been suggested that bootstrap estimators might actually *overestimate* the SEs; e.g. [36, p. 740] and [35, p. 1041]. At the model development stage, it is often of interest to gauge the strength of association of model covariates, which is not feasible with repeated bootstrap implementations. Hence, an approximate SE estimator is desirable. In either case, the theoretical properties will be contaminated by the addition of MC error from the MCEM algorithm, and it is not yet fully understood what the ramifications of this are. Hence, any standard errors must be interpreted with a degree of caution. We consider two estimators below.

1. Bootstrap method. These are estimated by sampling n subjects with replacement and re-labelling the subjects with indices $i' = 1, \dots, n$. We then re-fit the model to the bootstrap-sampled dataset. It is important to note that we re-sample subjects, not individual data points. This is repeated B-times, for a sufficiently large integer B. Since we already have the MLEs from the fitted model, we can use these as initial values for each bootstrap model fit, thus reducing initial computational overheads in calculating approximate initial parameters. For each iteration, we extract the model parameter estimates for $(\boldsymbol{\beta}^\top, \text{vech}(\boldsymbol{D}), \sigma_1^2, \dots, \sigma_K^2, \boldsymbol{\gamma}_v^\top, \boldsymbol{\gamma}_y^\top)$. Note that we do not estimate SEs for $\lambda_0(t)$ using this approach. However, they are generally not of inferential interest. When B is sufficiently large, the SEs can be estimated from the estimated coefficients of the bootstrap samples. Alternatively, $100(1-\alpha)$%-confidence intervals can be estimated from the the $100\alpha/2$-th and $100(1-\alpha/2)$-th percentiles.

2. Empirical information matrix method. Using the Breslow estimator for $\int_0^t \lambda_0(u) \text{d}u$, the profile score vector for $\boldsymbol{\theta}_{-\lambda} = (\boldsymbol{\beta}^\top, \text{vech}(\boldsymbol{D}), \sigma_1^2, \dots, \sigma_K^2, \boldsymbol{\gamma}^\top)$ is calculated (see **Additional file 1**). We approximate the profile information for $\boldsymbol{\theta}_{-\lambda}$ by $I_e^{-1/2}(\hat{\boldsymbol{\theta}}_{-\lambda_0})$, where $I_e(\boldsymbol{\theta}_{-\lambda_0})$ is the observed empirical information [25] given by

$$I_e(\boldsymbol{\theta}_{-\lambda}) = \sum_{i=1}^n s_i(\boldsymbol{\theta}_{-\lambda})^{\otimes 2} - \frac{1}{n} S(\boldsymbol{\theta}_{-\lambda})^{\otimes 2}, \tag{9}$$

$s_i(\boldsymbol{\theta}_{-\lambda})$ is the conditional expectation of the complete-data profile score for subject i, $S(\boldsymbol{\theta}_{-\lambda})$ is the score defined by $S(\boldsymbol{\theta}_{-\lambda}) = \sum_{i=1}^n s_i(\boldsymbol{\theta}_{-\lambda})$, and $\boldsymbol{a}^{\otimes 2} = \boldsymbol{a}\boldsymbol{a}^\top$ is outer product for a vector $\boldsymbol{a}$. At the maximizer, $S(\hat{\boldsymbol{\theta}}) = 0$, meaning that the right hand-side of (9) is zero. Due to the MC error in the MCEM algorithm, this will not be exactly zero, and therefore we include it in the calculations. As per the bootstrap approach, SEs for the baseline hazard are again not calculated. We note that this SE estimator will be subject to the exact same theoretical limitation of underestimation described by Hsieh *et al.* [35], since the profiling was implicit; that is, because the posterior expectations involve the parameters θ.

Software

The model described here is implemented in the R package `joineRML`, which is available on the The Comprehensive R Archive Network (CRAN) (https://CRAN.R-project.org/package=joineRML). The principal function in `joineRML` is `mjoint()`. The primary arguments for implementing `mjoint()` are summarised in Table 1. To achieve computationally efficiency parts of the MCEM algorithm in `joineRML` are coded in C++ using the Armadillo linear algebra library and integrated using the R package `RcppArmadillo` [37].

A model fitted using the `mjoint()` function returns an object of class `mjoint`. By default, approximate SE estimates are calculated. If one wishes to use bootstrap standard error estimates, then the user can pass the model object to the `bootSE()` function. Several generic functions (or rather, S3 methods) can also be applied to `mjoint` objects, as described in Table 2. These generic functions include common methods, for example `coef()`, which extracts the model coefficients; `ranef()`, which extracts the BLUPs (and optional standard errors); and `resid()`, which extracts the residuals from the linear mixed sub-model. The intention of these functions is to have a common syntax with standard R packages for linear mixed models [26] and survival analysis [27]. Additionally, plotting capabilities are included in `joineRML`. These include trace plots for assessment of convergence of the MCEM algorithm, and caterpillar plots for subject-specific random effects (Table 2).

The package also provides several datasets, and a function `simData()` that allows for simulation of data from joint models with multiple longitudinal outcomes. `joineRML` can also fit univariate joint models, however in this case we would currently recommend that the R packages `joineR` [29], `JM` [28], or `frailtypack` [38] are used, which are optimized for the univariate case and exploits Gaussian quadrature. In addition, these packages allow for extensions to more complex cases; for example, competing risks [29, 28] and recurrent events [38].

Results

Simulation analysis

A simulation study was conducted assuming two longitudinal outcomes and $n = 200$ subjects. Longitudinal data were simulated according to a follow-up schedule of 6 time points (at times $0, 1, \dots, 5$), with each model including subject-and-outcome-specific random-intercepts and random-slopes: $\boldsymbol{b}_i = (b_{0i1}, b_{1i1}, b_{0i2}, b_{1i2})^\top$, Correlation was induced between the 2 outcomes by assuming correlation of -0.5 between the random intercepts for each outcome. Event times were simulated from a Gompertz distribution with shape $\theta_1 = -3.5$ and scale $\exp(\theta_0) = \exp(0.25) \approx 1.28$, following the methodology described by Austin [39]. Independent censoring times were drawn from an exponential distribution with rate 0.05. Any subject where the event and censoring time exceeded 5 was censored at the truncation time $C = 5.1$. For all sub-models, we included a pair of covariates $\boldsymbol{X}_i = (x_{i1}, x_{i2})^\top$, where x_{i1} is a continuous covariate independently drawn from $N(0, 1)$ and x_{i2} is a binary covariate independently drawn from $Bin(1, 0.5)$. The sub-models are given as

$$
\begin{aligned}
y_{ijk} &= (\beta_{0,k} + b_{i0k}) + (\beta_{1,k} + b_{i1k})t_j + \beta_{2,k}x_{i1} + \beta_{3,k}x_{i2} + \varepsilon_{ijk}, \text{ for } k = 1, 2; \\
\lambda_i(t) &= \exp\left\{(\theta_0 + \theta_1 t) + \gamma_{v1}x_{i1} + \gamma_{v2}x_{i2} + \gamma_{y1}(b_{i01} + b_{i11}t) + \gamma_{y2}(b_{i02} + b_{i12}t)\right\}; \\
\boldsymbol{b}_i &\sim N_4(0, D); \\
\varepsilon_{ijk} &\sim N(0, \sigma_k^2),
\end{aligned}
$$

where D is specified unstructured (4×4)-covariance matrix with 10 unique parameters. Simulating datasets is straightforward using the `joineRML` package by means of the `simData()` function. The true parameter values and results from 500 simulations are shown in Table 3. In particular, we display the mean estimate, the bias, the empirical SE (= the standard deviation of the the parameter estimates); the mean SE (= the mean SE of each parameter calculated for each fitted model); the mean square error (MSE), and the coverage. The results confirm that the model fitting algorithm generally performs well.

A second simulation analysis was conducted using the parameters above (with $n = 100$ subjects per dataset). However, in this case we used a heavier-tailed distribution for the random effects: a multivariate t_5 distribution [40]. The bias for the fixed effect coefficients was comparable to the multivariate normal random effects simulation study (above). The empirical standard error was consistently smaller than the mean standard error, resulting in coverage between 95% and 99% for the coefficient parameters. Rizopoulos *et al.* [41] noted that the misspecification of the random effects distributions was minimised as the number of longitudinal measurements per subject increased, but that the standard errors are generally affected. These findings are broadly in agreement with the simulation study conducted here, and other studies [42, 43]. Choi *et al.* [44] provide a review of existing research on the misspecification of random effects in joint modelling.

Example

We consider the primary biliary cirrhosis (PBC) data collected at the Mayo Clinic between 1974 to 1984 [45]. This dataset has been widely analyzed using joint modelling methods [46, 47, 18]. PBC is a long-term liver disease in which the bile ducts in the liver become damaged. Progressively, this leads to a build-up of bile in the liver, which can damage it and eventually lead to cirrhosis. If PBC is not treated or reaches an advanced stage, it can lead to several major complications, including mortality. In this study, 312 patients were randomised to receive D-penicillamine ($n = 158$) or placebo ($n = 154$). In this example we analyse the subset of patients randomized to placebo.

Patients with PBC typically have abnormalities in several blood tests; hence, during follow-up several biomarkers associated with liver function were serially recorded for these patients. We consider three biomarkers: serum bilirunbin (denoted `serBilir` in the model and data; measured in units of mg/dl), serum albumin (`albumin`; mg/dl), and prothrombin time (`prothrombin`; seconds). Patients had a mean 6.3 (SD = 3.7) visits (including baseline). The data can be accessed from the `joineRML` package via the command `data(pbc2)`. Profile plots for each biomarker are shown in Figure 1, indicating distinct differences in trajectories between the those who died during follow-up and those who did not (right-censored cases). A Kaplan-Meier curve for overall survival is shown in Figure 2. There were a total of 69 (44.8%) deaths during follow-up in the placebo subset.

We fit a relatively simple joint model for the purposes of demonstration, which encompasses the following trivariate longitudinal data sub-model:

$$\begin{aligned}
\log(\texttt{serBilir}) &= (\beta_{0,1} + b_{0i,1}) + (\beta_{1,1} + b_{1i,1})\texttt{year} + \varepsilon_{ij1}, \\
\texttt{albumin} &= (\beta_{0,2} + b_{0i,2}) + (\beta_{1,2} + b_{1i,2})\texttt{year} + \varepsilon_{ij2}, \\
(0.1 \times \texttt{prothrombin})^{-4} &= (\beta_{0,3} + b_{0i,3}) + (\beta_{1,3} + b_{1i,3})\texttt{year} + \varepsilon_{ij3}, \\
\boldsymbol{b}_i &\sim N_6(0, \boldsymbol{D}), \text{ and } \varepsilon_{ijk} \sim N(0, \sigma_k^2) \text{ for } k = 1, 2, 3;
\end{aligned}$$

and a time-to-event sub-model for the study endpoint of death:

$$\begin{aligned}
\lambda_i(t) &= \lambda_0(t) \exp\left\{\gamma_v \texttt{age} + W_{2i}(t)\right\}, \\
W_{2i}(t) &= \gamma_{\texttt{bil}}(b_{0i,1} + b_{1i,1}t) + \gamma_{\texttt{alb}}(b_{0i,2} + b_{1i,2}t) + \gamma_{\texttt{pro}}(b_{0i,3} + b_{1i,3}t).
\end{aligned}$$

The log transformation of bilirubin is standard, and confirmed reasonable based on inspection of Q-Q plots for residuals from a separate fitted linear mixed model fitted using the `lme()` function from the R package `nlme`. Albumin did not require transformation. Residuals were grossly non-normal for prothrombin time using both untransformed and log-transformed outcomes. Therefore, a Box-Cox transformation was applied, which suggested an inverse-quartic transform might be suitable, which was confirmed by inspection of a Q-Q plot. The pairwise correlations for baseline measurements between the three transformed markers were 0.19 (prothrombin time *vs.* albumin), −0.30 (bilirubin *vs.* prothrombin time and albumin) The model is fit using the `joineRML` R package (version 0.2.0) using the following code.

```
# Get data
data(pbc2)
placebo <- subset(pbc2, drug == "placebo")

# Fit model
fit.pbc <- mjoint(
    formLongFixed = list(
        "bil" = log(serBilir) ~ year,
        "alb" = albumin ~ year,
        "pro" = (0.1 * prothrombin)^-4 ~ year),
    formLongRandom = list(
        "bil" = ~ year | id,
        "alb" = ~ year | id,
        "pro" = ~ year | id),
    formSurv = Surv(years, status2) ~ age,
    data = placebo,
    timeVar = "year",
    control = list(tol0 = 0.001, burnin = 400)
)
```

Here, we have specified a more stringent tolerance value for ϵ_0 than the default setting in `mjoint()`. Additionally, the burn-in phase was increased to 400 iterations after inspection of convergence trace plots. The model fits in 3.1 minutes on a MacBook Air 1.6GHz Intel Core i5 with 8GB or RAM running R version 3.3.0, having completed 423 MCEM iterations (not including the EM algorithm iterations performed for determining the initial values of the separate multivariate linear mixed sub-model) with a final MC size of $M = 3528$. The fitted model results are shown in Table 4.

The fitted model indicated that an increase in the subject-specific random deviation from the population trajectory of serum bilirubin was significantly associated with increased hazard of death. A significant association was also detected for subject-specific decreases in albumin from the population mean trajectory. However, prothrombin time was not significantly associated with hazard of death, although its direction is clinically consistent with PBC disease. Albert and Shih [46] analysed the first 4-years follow-up from this dataset with the same 3 biomarkers and a discrete event time distribution using a regression calibration model. Their results were broadly consistent, although the effect of prothrombin time on the event time sub-model was strongly significant.

We also fitted 3 univariate joint models to each of the biomarkers and the event time sub-model using the R package `joineR` (version 1.2.0) owing to its optimization for such models. The LMM parameter estimates were similar, although the absolute magnitude of the slopes was smaller for the separate univariate models. Since 3 separate models were fitted, 3 estimates of γ_v were estimated, with the average comparable to the multivariate model estimate. The multivariate model estimates of $\boldsymbol{\gamma}_y = (\gamma_{\texttt{bil}}, \gamma_{\texttt{alb}}, \gamma_{\texttt{pro}})^\top$ were substantially attenuated relative to the separate model estimates, although the directions remained consistent. It is also interesting to note that $\gamma_{\texttt{pro}}$ was statistically significant in the univariate model. However, the univariate models are not accounting for the correlation between different outcomes, whereas the multivariate joint model does.

The model was refitted with the one-step Newton-Raphson update for γ replaced by a Gauss-Newton-like update in a time of 2.2 minutes for 419 MCEM iterations with a final MC size of $M = 6272$. This is easily achieved by running the following code.

```
fit.pbc.gn <- update(fit.pbc, gammaOpt = "GN")
```

In addition, we bootstrapped this model with $B = 100$ samples to estimate SEs and contrast them with the approximate estimates based on the inverse empirical profile information matrix. In practice, one should choose $B > 100$, particularly if using bootstrap percentile confidence intervals; however, we used a small value to reduce the computational burden on this process. In a similar spirit, we relaxed the convergence criteria and lowered reduced the number of burn-in iterations. This is easily implemented by running the following code, taking 1.8 hours to fit.

```
fit.pbc.gn.boot <- bootSE(fit.pbc.gn, nboot = 100, control = list(
    tol0 = 0.005, tol2 = 0.01, convCrit = "sas",
    burnin = 300, mcmaxIter = 350))
```

It was observed that the choice of gradient matrix in the γ-update led to virtually indistinguishable parameter estimates, although we note the same random seed was used in both cases. The bootstrap estimated SEs were broadly consistent with the approximate SEs, with no consistent pattern in underestimation observed.

Discussion

Multivariate joint models introduce three types of correlations: (1) within-subject serial correlation for repeated measures; (2) between longitudinal outcomes correlation; and (3) correlation between the multivariate LMM and time-to-event sub-models. It is important to account for all of these types of correlations; however, some authors have reported collapsing their multivariate data to permit univariate joint models to be fitted. For example, Battes *et al.* [7] used an *ad hoc* approach of either summing or multiplying the three repeated continuous measures (standardized according to clinical upper reference limits of the biomarker assays), and then applying standard univariate joint models. Wang *et al.* [48] fitted separate univariate joint models to each longitudinal outcome in turn. Neither approach takes complete advantage of the correlation between the multiple longitudinal measures and the time-to-event outcome.

Here, we described a new R package `joineRML` that can fit the models described in this paper. This was demonstrated on a real-world dataset. Although in the fitted model we assumed linear trajectories for the biomarkers, splines could be straightforwardly employed, as have been used in other multivariate joint model applications [15], albeit at the cost of additional computational time. Despite a growing availability of software for univariate joint models, Hickey *et al.* [19] noted that there were very few options for fitting joint models involving multivariate longitudinal data. To the best of our knowledge, options are limited to the R packages `JMbayes` [49], `rstanarm` [50], and the Stata package `stjm` [47]. Whilst all of these packages are available, the extension to multivariate data remain features in the developmental versions only. Moreover, none of these incorporates an unspecified baseline hazard.

The first two packages use Markov chain Monte Carlo (MCMC) methods to fit the joint models. Bayesian models are potentially very useful for fitting joint models, and in particular for dynamic prediction; however, MCMC is also computationally demanding, especially in the case of multivariate models. Several other publications have made `BUGS` code available for use with WinBUGS and OpenBUGS (e.g. [51]), but these are not easily modifiable and post-fit computations are cumbersome.

`joineRML` is a new software package developed to fill a void in the joint modelling field, but is still in its infancy relative to highly developed univariate joint model packages such as the R package `JM` [28] and Stata package `stjm` [47]. Future developments of `joineRML` intend to cover several deficiencies. First, `joineRML` currently only permits an association structure of the form $W_{2i}(t) = \sum_{k=1}^{K} \gamma_{yk} W_{1i}^{(k)}(t)$. As has been demonstrated by others, the association might take different forms, including random-slopes and cumulative effects or some combination of multiple structures, and these may also be different for separate longitudinal outcomes [18]. Moreover, it is conceivable that separate longitudinal outcomes may interact in the hazard sub-model. Second, the use of MC integration provides a scalable solution to the issue of increasing dimensionality in the random effects. However, for simpler cases, e.g. bivariate models with random-intercepts and random-slopes (total of 4 random effects), Gaussian quadrature might be computationally superior; this trade-off requires further investigation. Third, `joineRML` can currently only model a single event time. However, there is a growing interest in competing risks [9] and recurrent events data [11], which if incorporated into `joineRML`, would provide a flexible all-round multivariate joint modelling platform. Competing risks [29, 28] and recurrent events [38] have been incorporated into R packages already, but are limited to the case of a solitary longitudinal outcome. Of note, the PBC trial dataset analysed in this study includes times to the competing risk of kidney transplantation. Fourth, with ever-increasing volumes of data collected during routine clinical visits, the need for software to fit joint models with very many longitudinal outcomes is foreseeable [52]. This would likely require the use of approximate methods for the numerical integration or data reduction methods. Fifth, additional residual diagnostics are necessary for assessing possible violations of model assumptions. The `joineRML` package has a `resid()` function for extracting the longitudinal sub-model residuals; however, these are complex for diagnostic purposes due to the informative dropout, hence the development of multiple-imputation based residuals [53].

Conclusions

In this paper we have presented an extension of the classical joint model proposed by Henderson *et al.* [3] and an estimation procedure for fitting the models that builds on the foundations laid by Lin *et al.* [20]. In addition, we described a new R package `joineRML` that can fit the models described in this paper, which leverages the MCEM algorithm which should scale well for increasing number of longitudinal outcomes. This software is timely, as it has previously been highlighted that there is a paucity of software available to fit such models [19]. The software is being regularly updated and improved.

Availability and requirements

Project name: `joineRML`
Project home page: https://github.com/graemeleehickey/joineRML/
Operating system(s): platform independent
Programming language: R
Other requirements: none
License: GNU GPL-3
Any restrictions to use by non-academics: none

Abbreviations

MCEM – Monte Carlo expectation maximisation; EM – expectation maximisation; MC – Monte Carlo; LMM – linear mixed models; BLUP – best linear unbiased prediction; SE – standard error; MLE – maximum likelihood estimate; CRAN – The Comprehensive R Archive Network; PBC – primary biliary cirrhosis; SD – standard deviation

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The R package `joineRML` can be installed directly using `install.packages("joineRML")` in an R console. The source code is available at https://github.com/graemeleehickey/joineRML. Archived versions are available from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/web/packages/joineRML/. `joineRML` is platform independent, requiring R version $\geq 3.3.0$, and is published under a GNU GPL-3 license. The dataset analysed during the current study is bundled with the R package `joineRML`, and can be accessed by running the command `data(pbc2, package = "joineRML")`.

Competing interests

The authors declare that they have no competing interests.

Funding

Funding for the project was provided by the Medical Research Council (Grant number MR/M013227/1). The funder had no role in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

Author's contributions

All authors collaborated in developing the model fitting algorithm reported. The programming and running of the analysis was carried out by GLH. GLH wrote the first draft of the manuscript, with revisions provided by PP, AJ, and RKD. All authors contributed to the manuscript revisions. All authors read and approved the final manuscript.

Acknowledgements

The authors would like to thank Professor Robin Henderson (University of Newcastle) for useful discussions with regards to the MCEM algorithm, and Dr Haiqun Lin (Yale University) for helpful discussions on the likelihood specification.

Author details

[1]Department of Biostatistics, Institute of Translational Medicine, University of Liverpool, Waterhouse Building, 1-5 Brownlow Street, L69 3GL Liverpool, UK. [2]Department of Mathematics, Physics and Electrical Engineering, Northumbria University, Ellison Place, NE1 8ST Newcastle upon Tyne, UK.

References

1. Ibrahim, J.G., Chu, H., Chen, L.M.: Basic concepts and methods for joint models of longitudinal and survival data. Journal of Clinical Oncology **28**(16), 2796–2801 (2010)
2. Sweeting, M.J., Thompson, S.G.: Joint modelling of longitudinal and time-to-event data with application to predicting abdominal aortic aneurysm growth and rupture. Biometrical Journal **53**(5), 750–763 (2011)
3. Henderson, R., Diggle, P.J., Dobson, A.: Joint modelling of longitudinal measurements and event time data. Biostatistics **1**(4), 465–480 (2000)
4. Tsiatis, A.A., Davidian, M.: Joint modeling of longitudinal and time-to-event data: an overview. Statistica Sinica **14**, 809–834 (2004)
5. Gould, A.L., Boye, M.E., Crowther, M.J., Ibrahim, J.G., Quartey, G., Micallef, S., Bois, F.Y.: Joint modeling of survival and longitudinal non-survival data: current methods and issues. report of the dia bayesian joint modeling working group. Statistics in Medicine **34**, 2181–2195 (2015)
6. Rizopoulos, D.: Joint Models for Longitudinal and Time-to-Event Data, with Applications in R. Chapman & Hall/CRC, Boca Raton, FL (2012)
7. Battes, L.C., Caliskan, K., Rizopoulos, D., Constantinescu, A.A., Robertus, J.L., Akkerhuis, M., Manintveld, O.C., Boersma, E., Kardys, I.: Repeated measurements of nt-pro-b-type natriuretic peptide, troponin t or c-reactive protein do not predict future allograft rejection in heart transplant recipients. Transplantation **99**(3), 580–585 (2015)
8. Song, X., Davidian, M., Tsiatis, A.A.: An estimator for the proportional hazards model with multiple longitudinal covariates measured with error. Biostatistics **3**(4), 511–528 (2002)
9. Williamson, P.R., Kolamunnage-Dona, R., Philipson, P., Marson, A.G.: Joint modelling of longitudinal and competing risks data. Statistics in Medicine **27**, 6426–6438 (2008)
10. Hickey, G.L., Philipson, P., Jorgensen, A., Kolamunnage-Dona, R.: A comparison of joint models for longitudinal and competing risks data, with application to an epilepsy drug randomized controlled trial. Journal of the Royal Statistical Society. Series A: Statistics in Society (2018). doi:10.1111/rssa.12348
11. Liu, L., Huang, X.: Joint analysis of correlated repeated measures and recurrent events processes in the presence of death, with application to a study on acquired immune deficiency syndrome. Journal of the Royal Statistical Society. Series C: Applied Statistics **58**(1), 65–81 (2009)
12. Chi, Y.Y., Ibrahim, J.G.: Joint models for multivariate longitudinal and multivariate survival data. Biometrics **62**(2), 432–445 (2006)
13. Hickey, G.L., Philipson, P., Jorgensen, A., Kolamunnage-Dona, R.: Joint models of longitudinal and time-to-event data with more than one event time outcome: a review. The International Journal of Biostatistics (2018). doi:10.1515/ijb-2017-0047
14. Andrinopoulou, E.-R., Rizopoulos, D., Takkenberg, J.J.M., Lesaffre, E.: Combined dynamic predictions using joint models of two longitudinal outcomes and competing risk data. Statistical Methods in Medical Research **0**(0), 1–18 (2015)
15. Rizopoulos, D., Ghosh, P.: A bayesian semiparametric multivariate joint model for multiple longitudinal outcomes and a time-to-event. Statistics in Medicine **30**(12), 1366–1380 (2011)
16. Faucett, C.L., Thomas, D.C.: Simultaneously modelling censored survival data and repeatedly measured covariates: a gibbs sampling approach. Statistics in Medicine **15**(15), 1663–1685 (1996)
17. Song, X., Davidian, M., Tsiatis, A.A.: A semiparametric likelihood approach to joint modeling of longitudinal and time-to-event data. Biometrics **58**(4), 742–53 (2002)
18. Andrinopoulou, E.-R., Rizopoulos, D.: Bayesian shrinkage approach for a joint model of longitudinal and survival outcomes assuming different association structures. Statistics in Medicine **35**(26), 4813–4823 (2016)
19. Hickey, G.L., Philipson, P., Jorgensen, A., Kolamunnage-Dona, R.: Joint modelling of time-to-event and multivariate longitudinal outcomes: recent developments and issues. BMC Medical Research Methodology **16**(1), 1–15 (2016)
20. Lin, H., McCulloch, C.E., Mayne, S.T.: Maximum likelihood estimation in the joint analysis of time-to-event and multiple longitudinal variables. Statistics in Medicine **21**, 2369–2382 (2002)
21. Laird, N.M., Ware, J.H.: Random-effects models for longitudinal data. Biometrics **38**(4), 963–74 (1982)
22. Wei, G.C., Tanner, M.A.: A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. Journal of the American Statistical Association **85**(411), 699–704 (1990)
23. Wulfsohn, M.S., Tsiatis, A.A.: A joint model for survival and longitudinal data measured with error. Biometrics

53(1), 330–339 (1997)
24. Ratcliffe, S.J., Guo, W., Ten Have, T.R.: Joint modeling of longitudinal and survival data via a common frailty. Biometrics **60**(4), 892–899 (2004)
25. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions, 2nd edn. Wiley-Interscience, ??? (2008)
26. Pinheiro, J.C., Bates, D.M.: Mixed-Effects Models in S and S-PLUS. Springer, New York (2000)
27. Therneau, T.M., Grambsch, P.M.: Modeling Survival Data: Extending the Cox Model, p. 350. Springer, New Jersey (2000)
28. Rizopoulos, D.: Jm: an r package for the joint modelling of longitudinal and time-to-event data. Journal of Statistical Software **35**(9), 1–33 (2010)
29. Philipson, P., Sousa, I., Diggle, P.J., Williamson, P., Kolamunnage-Dona, R., Henderson, R., Hickey, G.L.: `joineR`: Joint Modelling of Repeated Measurements and Time-to-event Data. (2017). R package version 1.2.0. https://github.com/graemeleehickey/joineR/
30. Dmitrienko, A., Molenberghs, G., Chuang-Stein, C., Offen, W.W.: Analysis of Clinical Trials Using SAS: A Practical Guide. SAS Institute, ??? (2005)
31. Law, N.J., Taylor, J.M., Sandler, H.: The joint modeling of a longitudinal disease progression marker and the failure time process in the presence of cure. Biostatistics **3**(4), 547–563 (2002)
32. McCulloch, C.E.: Maximum likelihood algorithms for generalized linear mixed models. Journal of the American Statistical Association **92**(437), 162–170 (1997)
33. Booth, J.G., Hobert, J.P.: Maximizing generalized linear mixed model likelihoods with an automated monte carlo em algorithm. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **61**(1), 265–285 (1999)
34. Ripatti, S., Larsen, K., Palmgren, J.: Maximum likelihood inference for multivariate frailty models using an automated monte carlo em algorithm. Lifetime Data Analysis **8**(2002), 349–360 (2002)
35. Hsieh, F., Tseng, Y.K., Wang, J.L.: Joint modeling of survival and longitudinal data: Likelihood approach revisited. Biometrics **62**(4), 1037–1043 (2006)
36. Xu, C., Baines, P.D., Wang, J.L.: Standard error estimation using the em algorithm for the joint modeling of survival and longitudinal data. Biostatistics **15**(4), 731–44 (2014)
37. Eddelbuettel, D., Sanderson, C.: Rcpparmadillo: accelerating r with high-performance c++ linear algebra. Computational Statistics and Data Analysis **71**, 1054–1063 (2014)
38. Król, A., Mauguen, A., Mazroui, Y., Laurent, A., Michiels, S., Rondeau, V.: 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)
39. Austin, P.C.: Generating survival times to simulate cox proportional hazards models with time-varying covariates. Statistics in Medicine **31**(29), 3946–3958 (2012)
40. Genz, A., Bretz, F.: Computation of Multivariate Normal and T Probabilities. Springer, Berlin (2009)
41. Rizopoulos, D., Verbeke, G., Molenberghs, G.: Shared parameter models under random-effects misspecification. Biometrika **95**(1), 63–74 (2008)
42. Xu, J., Zeger, S.L.: The evaluation of multiple surrogate endpoints. Biometrics **57**(1), 81–87 (2001)
43. Pantazis, N., Touloumi, G.: Robustness of a parametric model for informatively censored bivariate longitudinal data under misspecification of its distributional assumptions: a simulation study. Statistics in Medicine **26**, 5473–5485 (2007)
44. Choi, J., Zeng, D., Olshan, A.F., Cai, J.: Joint modeling of survival time and longitudinal outcomes with flexible random effects. Lifetime Data Analysis **24**(1), 126–152 (2018)
45. Murtaugh, P.A., Dickson, E.R., Van Dam, G.M., Malinchoc, M., Grambsch, P.M., Langworthy, A.L., Gips, C.H.: Primary biliary cirrhosis: prediction of short-term survival based on repeated patient visits. Hepatology **20**(1), 126–134 (1994)
46. Albert, P.S., Shih, J.H.: An approach for jointly modeling multivariate longitudinal measurements and discrete time-to-event data. Annals of Applied Statistics **4**(3), 1517–1532 (2010)
47. Crowther, M.J., Abrams, K.R., Lambert, P.C.: Joint modeling of longitudinal and survival data. The Stata Journal **13**(1), 165–184 (2013)
48. Wang, P., Shen, W., Boye, M.E.: Joint modeling of longitudinal outcomes and survival using latent growth modeling approach in a mesothelioma trial. Health Services and Outcomes Research Methodology **12**(2-3), 182–199 (2012)
49. Rizopoulos, D.: 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)
50. Carpenter, B., Gelman, A., Hoffman, M.D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M.A., Li, P., Riddell, A.: Stan: a probabilistic programming language. Journal of Statistical Software **76**(1), 1–32 (2017)

51. Andrinopoulou, E.-R., Rizopoulos, D., Takkenberg, J.J.M., Lesaffre, E.: Joint modeling of two longitudinal outcomes and competing risk data. Statistics in Medicine **33**(18), 3167–3178 (2014)
52. Jaffa, M.A., Gebregziabher, M., Jaffa, A.A.: A joint modeling approach for right censored high dimensiondal multivariate longitudinal data. Journal of Biometrics and Biostatistics **5**(4), 1000203 (2014)
53. Rizopoulos, D., Verbeke, G., Molenberghs, G.: Multiple-imputation-based residuals and diagnostic plots for joint models of longitudinal and survival outcomes. Biometrics **66**(1), 20–29 (2010)

Additional Files

Additional file 1

An appendix (appendix.pdf) is available that includes details on the score vector and M-step estimators.

Figures

Figure 1 Longitudinal trajectory plots. The black lines show individual subject trajectories, and the coloured lines show smoothed (LOESS) curves stratified by whether the patient experienced the endpoint (blue) or not (red).

Figure 2 Kaplan-Meier curve for overall survival. A pointwise 95% band is shown (dashed lines). In total, 69 patients (of 154) died during follow-up.

Tables

Table 1 The primary arguments[†] with descriptions for the `mjoint()` function in the R package `joineRML`.

Argument	Description
`formLongFixed`	a `list` of `formulae` for the fixed effects component of each longitudinal outcome. The left hand-hand side defines the response, and the right-hand side specifies the fixed effect terms.
`formLongRandom`	a `list` of one-sided `formulae` specifying the model for the random effects effects of each longitudinal outcome.
`formSurv`	a `formula` specifying the proportional hazards regression model (not including the latent association structure).
`data`	a list of `data.frame` objects for each longitudinal outcome in which to interpret the variables named in the `formLongFixed` and `formLongRandom`. The list structure enables one to include multiple longitudinal outcomes with different measurement protocols. If the multiple longitudinal outcomes are measured at the same time points for each patient (i.e. $t_{ijk} = t_{ij}\ \forall k$), then a single `data.frame` object can be given instead of a `list`. It is assumed that each data frame is in *long format*.
`survData`	(optional) a `data.frame` in which to interpret the variables named in the `formSurv`. If `survData` is not given, then `mjoint()` looks for the time-to-event data in `data`.
`timeVar`	a `character` string indicating the time variable in the linear mixed effects model.
`inits`	(optional) a `list` of initial values for some or all of the parameters estimated in the model.
`control`	(optional) a `list` of control parameters. These allow for the control of ϵ_0, ϵ_1, and ϵ_2 in (7) and (8); the choice of N, δ, and convergence criteria; the maximum number of MCEM iterations, and the minimum number of MCEM iterations during burn-in. Additionally, the control argument `gammaOpt` can be used to specify whether a one-step Newton-Raphson (`="NR"`) or Gauss-Newton-like (`="GN"`) update should be used for the M-step update of γ.

[†]`mjoint()` also takes the optional additional arguments `verbose`, which if `TRUE` allows for monitoring updates at each MCEM algorithm iteration, and `pfs`, which if `FALSE` can force the function not to calculate post-fit statistics such as the BLUPs and associated standard errors of the random effects and approximate standard errors of the model parameters. In general, these arguments are not required.

Table 2 Additional functions with descriptions that can be applied to objects of class `mjoint`[†].

Function(s)	Returns
`logLik`, `AIC`, `BIC`	the log-likelihood, Akaike information criterion and Bayesian information criterion statistics, respectively.
`coef`, `fixef`	the fixed effects parameter estimates.
`ranef`	the BLUPs (and optional standard errors).
`print`[†], `summary`*	short and long model summary outputs, respectively.
`fitted`, `resid`	the fitted and raw residuals from the multivariate LMM sub-model, respectively.
`plot`[‡]	the MCEM algorithm convergence trace plots.
`sigma`	the residual standard errors from the LMM sub-model.
`vcov`	the variance-covariance matrix of the main parameters of the fitted model (except the baseline hazard).
`getVarCov`	the random effects variance-covariance matrix.
`confint`	the confidence intervals based on asymptotic normality.
`update`	specific parts of a fitted model can be updated, e.g. by adding or removing terms from a sub-model, and then re-fitted.
`sampleData`	sample data (with or without replacement) from a joint model.

[†]`print()` also applies to objects of class `summary.mjoint` and `bootSE` inheriting from the `summary()` and `bootSE()` functions, respectively. [‡]`plot()` also accepts objects of class `ranef.mjoint` inheriting from the `ranef()` function, which displays a caterpillar plot (with 95% prediction intervals) for each random effect. *`summary()` can also take the optional argument of an object of class `bootSE` inheriting from the function `bootSE()`, which overrides the approximate SEs and CIs with those from a bootstrap estimation routine.

Table 3 Results of simulation study.

Parameter	True value	Mean estimated value	Empirical SE	Mean SE	Bias	MSE	Coverage
D_{11}	0.2500	0.2411	0.0435	—	-0.0089	0.0020	—
D_{21}	0.0000	0.0010	0.0136	—	0.0010	0.0002	—
D_{31}	-0.1250	-0.1212	0.0295	—	0.0038	0.0009	—
D_{41}	0.0000	-0.0006	0.0127	—	-0.0006	0.0002	—
D_{22}	0.0400	0.0396	0.0072	—	-0.0004	0.0001	—
D_{32}	0.0000	-0.0002	0.0138	—	-0.0002	0.0002	—
D_{42}	0.0000	-0.0001	0.0055	—	-0.0001	0.0000	—
D_{33}	0.2500	0.2420	0.0400	—	-0.0080	0.0017	—
D_{43}	0.0000	0.0007	0.0134	—	0.0007	0.0002	—
D_{44}	0.0400	0.0399	0.0075	—	-0.0001	0.0001	—
$\beta_{0,1}$	0.0000	0.0028	0.0612	0.0660	0.0028	0.0038	0.9660
$\beta_{1,1}$	1.0000	1.0012	0.0218	0.0229	0.0012	0.0005	0.9500
$\beta_{2,1}$	1.0000	1.0010	0.0449	0.0470	0.0010	0.0020	0.9540
$\beta_{3,1}$	1.0000	0.9932	0.0897	0.0925	-0.0068	0.0081	0.9440
σ_1^2	0.2500	0.2506	0.0165	0.0171	0.0006	0.0003	0.9560
$\beta_{0,2}$	0.0000	-0.0026	0.0637	0.0655	-0.0026	0.0041	0.9660
$\beta_{1,2}$	-1.0000	-1.0011	0.0229	0.0223	-0.0011	0.0005	0.9480
$\beta_{2,2}$	0.0000	0.0008	0.0399	0.0472	0.0008	0.0016	0.9700
$\beta_{3,2}$	0.5000	0.5061	0.0894	0.0923	0.0061	0.0080	0.9540
σ_2^2	0.2500	0.2501	0.0162	0.0171	0.0001	0.0003	0.9540
γ_{v1}	0.0000	0.0011	0.1243	0.1392	0.0011	0.0155	0.9720
γ_{v2}	1.0000	1.0487	0.2837	0.2750	0.0487	0.0829	0.9340
γ_{y1}	-0.5000	-0.5121	0.1936	0.2084	-0.0121	0.0376	0.9560
γ_{y2}	1.0000	1.0311	0.2220	0.2145	0.0311	0.0502	0.9400

Table 4 Fitted multivariate and separate univariate joint models to the PBC data.

	joineRML (NR)			joineRML (GN)			Bootstrap		joineR		
	Estimate	SE	95% CI†	Estimate	SE	95% CI†	SE	95% CI‡	Estimate	SE	95% CI‡
$\beta_{0,1}$	0.5541	0.0858	(0.3859, 0.7223)	0.5549	0.0846	(0.3892, 0.7207)	0.0800	(0.4264, 0.7435)	0.5545	0.0838	(0.3802, 0.7031)
$\beta_{1,1}$	0.2009	0.0201	(0.1616, 0.2402)	0.2008	0.0201	(0.1614, 0.2402)	0.0204	(0.1669, 0.2468)	0.1808	0.0209	(0.1430, 0.2324)
$\beta_{0,2}$	3.5549	0.0356	(3.4850, 3.6248)	3.5546	0.0357	(3.4846, 3.6245)	0.0255	(3.4972, 3.5904)	3.5437	0.0333	(3.4418, 3.6095)
$\beta_{1,2}$	-0.1245	0.0101	(-0.1444, -0.1047)	-0.1246	0.0101	(-0.1444, -0.1047)	0.0120	(-0.1489, -0.1063)	-0.0997	0.0113	(-0.1256, -0.0773)
$\beta_{0,3}$	0.8304	0.0212	(0.7888, 0.8719)	0.8301	0.0210	(0.7888, 0.8713)	0.0196	(0.7953, 0.8638)	0.8233	0.0220	(0.7818, 0.8677)
$\beta_{1,3}$	-0.0577	0.0062	(-0.0699, -0.0456)	-0.0577	0.0062	(-0.0698, -0.0455)	0.0057	(-0.0698, -0.0486)	-0.0447	0.0052	(-0.0555, -0.0362)
									0.0575^a	0.0123^a	$(0.0314, 0.0760)^a$
γ_v	0.0462	0.0151	(0.0166, 0.0759)	0.0462	0.0152	(0.0165, 0.0759)	0.0173	(0.0198, 0.0880)	0.0413^b	0.0150^b	$(0.0113, 0.0714)^b$
									0.0424^c	0.0157^c	$(0.0146, 0.0724)^c$
$\gamma_{\texttt{bil}}$	0.8181	0.2046	(0.4171, 1.2191)	0.8187	0.2036	(0.4197, 1.2177)	0.2153	(0.5172, 1.4021)	1.2182	0.1654	(0.9800, 1.5331)
$\gamma_{\texttt{alb}}$	-1.7060	0.6181	(-2.9173, -0.4946)	-1.6973	0.6163	(-2.9053, -0.4893)	0.7562	(-3.3862, -0.5188)	-3.0770	0.6052	(-4.7133, -2.1987)
$\gamma_{\texttt{pro}}$	-2.2085	1.6070	(-5.3582, 0.9412)	-2.2148	1.6133	(-5.3768, 0.9472)	1.6094	(-5.3050, 0.6723)	-7.2078	1.2640	(-10.5247, -5.2616)

Notation: SE = standard error; CI = confidence interval; NR = one-step Newton-Raphson update for γ; GN = one-step Gauss-Newton-like update for γ. †SEs are calculated from the inverse profile empirical information matrix, and confidence intervals are based on normal approximations of the type $\hat{\boldsymbol{\theta}} \pm 1.96\text{SE}(\hat{\boldsymbol{\theta}})$, where $\hat{\boldsymbol{\theta}}$ denote the estimated maximum likelihood estimates. ‡SEs and confidence intervals are derived from $B = 100$ bootstrap samples, with confidence intervals based on the 2.5% and 97.5% percentiles. NB. one model failed to converge using `joineRML` within the maximum number of MC iterations, and so SEs and CIs are based on 99 bootstrap samples only. ^{a}Separate model fit for `serBilir`. ^{b}Separate model fit for `albumin`. ^{c}Separate model fit for `prothrombin`.