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

---Manuscript Draft---

<table>
<thead>
<tr>
<th>Manuscript Number:</th>
<th>BMRM-D-17-00310R1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Title:</td>
<td>joineRML: A joint model and software package for time-to-event and multivariate longitudinal outcomes</td>
</tr>
<tr>
<td>Article Type:</td>
<td>Software</td>
</tr>
<tr>
<td>Section/Category:</td>
<td>Data analysis, statistics and modelling</td>
</tr>
<tr>
<td>Funding Information:</td>
<td>Medical Research Council (MR/M013227/1) Dr Ruwanthi Kolamunnage-Dona</td>
</tr>
</tbody>
</table>

**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.

**Corresponding Author:** Graeme Hickey, Ph.D.
University of Liverpool Institute of Translational Medicine
UNITED KINGDOM

**First Author:** Graeme Hickey, Ph.D.

**Order of Authors:** Graeme Hickey, Ph.D.
Pete Philipson
Andrea Jorgensen
Ruwanthi Kolamunnage-Dona

**Response to Reviewers:** Response to Reviewers
George Kypriotakis (Reviewer 1)
Comment: JoineRML is a much-needed package for survival and longitudinal data. The accumulation of longitudinal data on multiple processes requires multivariate easy-to-use analytic routines. JoineRML is a promising package in that regard, though still not in a fully developed stage. The authors did not discuss the "frailtypack", another very useful joint modelling framework.

Response and changes: We thank the Reviewer for their positive comments about our R package and manuscript, and for taking the time to review them both. As suggested, we have now mentioned 'frailtypack' as an alternative option for fitting univariate joint models (page 11, line 233).

Comment: The creators of the JoineRML have done an excellent job bringing together important statistical and algorithmic developments and they presented them with clarity. However, there are many additions needed to make this package likely to be used by researchers. It needs more flexibility with the definition of time and change, as the authors acknowledge, and plotting capabilities. It will be great if there was an extension of repeating and competing events predicting a terminal event in the presence of multiple processes.

Response and changes: We thank the Reviewer for their positive comments about our R package and manuscript. We agree that extensions to more complex time-to-event data are required. We now explicitly mention that extensions to competing risks and recurrent events are already handled by the R packages `joineR` and `JM`, and `frailtypack`, respectively (page 11, lines 233-236; page 17, lines 430-432). However, we also note that these packages are limited to univariate longitudinal outcomes. We also agree that additional association structures would add value to this value, and as you noted, this has been reported in the Discussion section (page 17, lines 417-241). We have expanded on the additional S3 methods that are available in the `joineRML` package, which includes some plotting functions (page 11, lines 222-229). Future versions of the package will have more plotting functions around dynamic prediction and model fitting diagnostics.

Comment: The ability to quickly estimate the association of multiple outcomes and survival events with an open source program is definitely welcomed.

Response and changes: We thank the Reviewer for their positive comments about our R package and manuscript. We look forward to it being used by others, as well as future developments.

Rajesh Talluri, Ph.D. (Reviewer 2)

Comment: The authors propose an efficient computational framework to fit a joint model comprising of a multivariate linear mixed sub-model for the longitudinal outcomes, and a Cox proportional hazards regression model with time-varying covariates for the event time sub-model. The association between the models is modelled using a zero-mean multivariate latent Gaussian process. The authors use the MCEM algorithm to infer the estimates of the joint model.

The manuscript is well written and a useful contribution to the development of joint models. I commend the authors for their contribution to the research community to appropriately analyse multiple longitudinal outcomes and time to event outcomes jointly.

However, this manuscript does not show how this software would perform in a realistic scenario when the assumptions of the model are generally violated.

Response and changes: We thank the Reviewer for their positive comments about our manuscript and R package. In order to ensure the manuscript was not excessively long, we had chosen to focus on the underlying methodology and software implementation for this manuscript, rather than explore a large number of simulations. However, we have now added further simulation analyses to the manuscript (pages 12-13, lines 260-271), which we describe below.

Comment: The modelling of association between the longitudinal and event sub
models is a very weak form of association which may lead investigators astray due to incorrect distributional assumptions on the random effects. This is because joint models are very sensitive the distribution of random effects. It would be very informative to researchers who want to use this software to understand how their results would be affected if the distribution of random effects in not normal.

Response and changes: We agree that this is a topic of relevance and importance. Motivated by the Reviewer's comment, we have now updated the simData() function in the joineRML R package to allow for simulating datasets with heavier-tailed random effects distributions. We have used this new function to perform a second smaller simulation study with multivariate t-distribution (5 degrees of freedom) random effects. Bias was minimal, although the standard errors were slightly affects (pages 12-13, lines 260-271).

Comment: The simulation study proves that the MCEM algorithm converges. Simulating data from the same model used to fit the data is not informative, as it is generally expected to retrieve the parameters as long as it is a valid model. Perhaps, the authors could simulate data using a realistic scenario (not using the same parameters as the proposed model and the use of informative censoring or similar techniques to induce artificial correlations that are present in real world data generation) and help the researchers understand how different correlations in the association between longitudinal outcomes and event process convert to the \( \gamma_y \) parameters and how they could interpret them. This is essential to the understanding and would encourage the use of the proposed model by researchers.

Response and changes: As noted by the Reviewer, we have already demonstrated (by simulation) that the algorithm leads to values with low mean-square error and sufficient coverage. This was based on the simulation routine of Austin et al. (Stat Med 2012;31:3946-58.). We also believe we have illustrated the consequence of fitting the wrong model in the clinical example. Namely, in Table 4 we investigate the consequence of ignoring the correlation between longitudinal outcomes. Whilst the effect could be quantified by simulation analysis, it is generally well-recognised that model misspecification can lead to bias (Ibrahim et al., 2010). No changes have been made.

Comment: The performance of the model when the data violates other assumptions of the model needs to be investigated in simulations. This should be available as a diagnostic check in the package. This would safeguard the tool being used inappropriately for any data set and report the results which may be biased or incorrect. These additions would go a long way to validate the results computed from this package and increase the quality of use of this package.

Response and changes: We thank the Reviewer for this comment. The joineRML package includes the resid() function for extract level-0 and level-1 residuals from the multivariate LMM sub-models. However, these residuals are more complex in a joint model setting. We have added this issue to our list of planned future developments (page 17, lines 437-441).
joineRML: A joint model and software package for time-to-event and multivariate longitudinal outcomes

Graeme L Hickey¹*, Pete Philipson², Andrea Jorgensen¹ and Ruwanthi Kolamunnage-Dona¹

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 mea-
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 longi-
tudinal 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 lon-
gitudinal 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 longitudi-
dinal data [8], competing risks data [9, 10], recurrent events data [11], multivariate
time-to-event data [12], non-continuous repeated measurements (e.g. count, binary,
ordinal, and censored data) [13], non-normally and non-parametrically distributed
random effects [14], alternative estimation methodologies (e.g. Bayesian fitting and
conditional estimating equations) [15, 16], and different association structures [17].
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. [18] identified that publicly avail-
able 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 out-
comes. An algorithm proposed by Lin et al. [19] 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 software package—joineRML. A simulation anal-
ysis and real-world data example are used to demonstrate the accuracy of the algo-

**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 $i = 1, \ldots, n$, $y_i = (y_{i1}^\top, \ldots, y_{iK}^\top)$ is the $K$-variate continuous out-

come vector, where each $y_{ik}$ denotes an $(n_{ik} \times 1)$-vector of observed longitudinal

measurements for the $k$-th outcome type: $y_{ik} = (y_{i1k}, \ldots, y_{in_{ik}k})^\top$. 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 $\delta_i$,

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 Hen-
derson 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 in-
dependently for each subject, $W_i(t) = \{W_{i1}^{(1)}(t), \ldots, W_{i1}^{(K)}(t), W_{i2}(t)\}$. This latent
process subsequently links the separate sub-models via association parameters.

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

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

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 $\sigma^2_k$. The mean response is specified as a linear model

$$\mu_{ik}(t) = \bf{x}_{ik}(t)^\top \beta_k,$$

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

$$W_{1i}^{(k)}(t) = \bf{z}_{ik}(t)^\top \beta_{ik},$$
where \( z_{ik}(t) \) is an \( r_k \)-vector of (possibly) time-varying covariates with corresponding subject-and-outcome random effect terms \( b_{ik} \), which follow a zero-mean multivariate normal distribution with \( (r_k \times r_k) \)-variance-covariance matrix \( D_{kk} \). To account for dependence between the different longitudinal outcome outcomes, we let \( \text{cov}(b_{ik}, b_{il}) = D_{kl} \) for \( k \neq l \). Furthermore, we assume \( \varepsilon_{ik}(t) \) and \( b_{ik} \) 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 [20] linear mixed effects model. More flexible specifications of \( W_{i1}^{(k)}(t) \) 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

\[
\lambda_i(t) = \lambda_0(t) \exp \{ \mathbf{v}_i^\top(t) \gamma_v + W_{2i}(t) \},
\]

where \( \lambda_0(\cdot) \) is an unspecified baseline hazard, and \( \mathbf{v}_i(t) \) is a \( q \)-vector of (possibly) time-varying covariates with corresponding fixed effect terms \( \gamma_v \). 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 \( \{ W_{1i}^{(1)}(t), \ldots, W_{1i}^{(K)}(t) \} \):

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

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

**Likelihood**

For each subject \( i \), let \( \mathbf{X}_i = \bigoplus_{k=1}^{K} \mathbf{X}_{ik} \) and \( \mathbf{Z}_i = \bigoplus_{k=1}^{K} \mathbf{Z}_{ik} \) be block-diagonal matrices, where \( \mathbf{X}_{ik} = (\mathbf{x}_{1ik}^\top, \ldots, \mathbf{x}_{r_{ik}}^\top) \) 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, \( \mathbf{Z}_{ik} \). We denote the error terms by a diagonal matrix \( \Sigma_i = \bigoplus_{k=1}^{K} \sigma^2_{ik} \mathbf{I}_{n_{ik}} \) and write the overall variance-covariance matrix for the random effects
as

\[
D = \begin{pmatrix}
D_{11} & \cdots & D_{1K} \\
\vdots & \ddots & \vdots \\
D_{1K} & \cdots & D_{KK}
\end{pmatrix},
\]

where \( I_n \) denotes an \( n \times n \) identity matrix. We further define \( \beta = (\beta_1^\top, \ldots, \beta_K^\top)^\top \)
and \( b_i = (b_{i1}^\top, \ldots, b_{iK}^\top)^\top \). Hence, we can then rewrite the longitudinal outcome
sub-model as

\[
y_i \mid b_i, \beta, \Sigma_i \sim N(X_i \beta + Z_i b_i, \Sigma_i),
\]

with \( b_i \mid D \sim N(0, D) \).

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. \( v_i = v_i(0) \). 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

\[
\prod_{i=1}^n \left( \int_{-\infty}^{\infty} f(y_i \mid b_i, \theta) f(T_i, \delta_i \mid b_i, \theta) f(b_i \mid \theta) db_i \right), \tag{4}
\]

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

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

\[
\prod_{i=1}^n f(y_i \mid \theta) \left( \int_{-\infty}^{\infty} f(T_i, \delta_i \mid b_i, \theta) f(b_i \mid y_i, \theta) db_i \right),
\]

where the marginal distribution \( f(y_i \mid \theta) \) is a multivariate normal density with mean
\( X_i \beta \) and variance-covariance matrix \( \Sigma_i + Z_i D Z_i^\top \), and \( f(b_i \mid y_i, \theta) \) is given by (6).

**MCEM algorithm**

We determine maximum likelihood estimates of the parameters \( \theta \) using the Monte Carlo Expectation Maximisation (MCEM) algorithm [21], by treating the random effects \( b_i \) as missing data. This is effectively the same as the conventional Expectation-Maximisation (EM) algorithm, as used by Wulfsohn and Tsiatis [22]
and Ratcliffe et al. [23] 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, \( \hat{\theta}^{(0)} \), 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,

\[
Q(\theta | \hat{\theta}^{(m)}) = \sum_{i=1}^{n} \mathbb{E}\left\{ \log f(y_i, T_i, \delta_i, b_i | \theta) \right\}
\]

\[
= \sum_{i=1}^{n} \int_{-\infty}^{\infty} \left\{ \log f(y_i, T_i, \delta_i, b_i | \theta) \right\} f(b_i | T_i, \delta_i, y_i; \hat{\theta}^{(m)}) db_i.
\]

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

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

\[
\hat{\theta}^{(m+1)} = \arg\max_{\theta} Q(\theta | \hat{\theta}^{(m)}).
\]

The M-step estimators naturally follow from Wulfsohn and Tsiatis [22] and Lin et al. [19]. Maximizers for all parameters except \( \gamma_v \) and \( \gamma_y \) are available in closed-form; algebraic details are presented in Additional file 1. The parameters \( \gamma = (\gamma_v^T, \gamma_y^T)^T \) are jointly updated using a one-step Newton-Raphson algorithm as

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

where \( \hat{\gamma}^{(m)} \) denotes the value of \( \gamma \) at the current iteration, \( S\left(\hat{\gamma}^{(m)}\right) \) is the corresponding score, and \( I\left(\hat{\gamma}^{(m)}\right) \) 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 \( \gamma \) is computationally expensive to evaluate. Therefore, we also propose a quasi-Newton one-step update by approximating \( I\left(\hat{\gamma}^{(m)}\right) \) by an empirical information matrix for \( \gamma \), which can be considered an analogue of the Gauss-Newton method [24, 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 \( I(\gamma) \).

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

$$\mathbb{E} \left[ h(b_i) \mid T_i, \delta_i, y_i; \hat{\theta} \right] = \frac{\int_{-\infty}^{\infty} h(b_i)f(b_i \mid y_i; \hat{\theta}) f(T_i, \delta_i \mid b_i; \hat{\theta})db_i}{\int_{-\infty}^{\infty} f(b_i \mid y_i; \hat{\theta}) f(T_i, \delta_i \mid b_i; \hat{\theta})db_i}, \quad (5)$$

where \( f(T_i, \delta_i \mid b_i; \hat{\theta}) \) is given by

$$f(T_i, \delta_i \mid b_i; \theta) = \left[ \lambda_0(T_i) \exp \left\{ v_i^\top \gamma_0 + W_2(T_i, b_i) \right\} \right]^{\delta_i} \times \exp \left\{ - \int_0^{T_i} \lambda_0(u) \exp \left\{ v_i^\top \gamma_0 + W_2(u, b_i) \right\} du \right\}$$

and \( f(b_i \mid y_i; \hat{\theta}) \) is calculated from multivariate normal distribution theory as

$$b_i \mid y_i, \theta \sim N \left( A_i \left\{ Z_i^\top \Sigma_i^{-1}(y_i - \bar{X}_i \beta) \right\}, A_i \right), \quad (6)$$

with \( A_i = \left( Z_i^\top \Sigma_i^{-1} Z_i + D_i^{-1}\right)^{-1} \). As this becomes computationally expensive using Gaussian quadrature commensurate with increasing dimension of \( b_i \), we estimate the integrals by MC sampling such that the expectation is approximated by the ratio of the sample means for \( h(b_i)f(T_i, \delta_i \mid b_i; \hat{\theta}) \) and \( f(T_i, \delta_i \mid b_i; \hat{\theta}) \) 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 \( \Omega \sim N(0, I_r) \) and obtain the pairs

$$A_i \left\{ Z_i^\top \Sigma_i^{-1}(y_i - \bar{X}_i \beta) \right\} \pm C_i \Omega,$$

where \( C_i \) is the Cholesky decomposition of \( A_i \) such that \( C_i C_i^\top = A_i \). 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 \( \hat{\theta}^{(0)} \). 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` [25] and `survival` [26],
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 $\gamma_v$ and set $\gamma_{yk} = 0 \forall k$.

For the longitudinal data sub-model, when $K > 1$ we first find the maximum likelihood estimate of $\{\beta, \text{vech}(D), \sigma_1^2, \ldots, \sigma_K^2\}$ 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 $D$ 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.

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

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

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

respectively, for some appropriate choice of $\epsilon_0$, $\epsilon_1$, and $\epsilon_2$, where the maximum is taken over the components of $\theta$. For reference, the R package JM [27] implements (7) (in combination with another rule based on relative change in the likelihood), whereas the R package joineR [28] 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 [29, 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 [21], 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 \[30\] or by pre-specified rules \[31\], an automated approach is preferable and necessary for a software implementation. Booth and Hobert \[32\] 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 \textit{et al.} \[33\]. 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 \( \delta \). 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 100\( K \) 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 \( \epsilon \)-values are too large. To reduce the chance of this occurring, we require that the stopping rule is satisfied for 3 consecutive iterations \[32, 33\]. However, in any case, trace plots should be inspected to confirm convergence is appropriate.

\textit{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 \[34\]. 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. [34] 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. [35, p. 740] and [34, 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 $i' = 1, \ldots, n$. 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 $(\beta^T, \text{vech}(D), \sigma_1^2, \ldots, \sigma_K^2, \gamma_i^T, \gamma_y^T)$. 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 $\int_0^t \lambda_0(u)du$, the profile score vector for $\theta - \lambda = (\beta^T, \text{vech}(D), \sigma_1^2, \ldots, \sigma_K^2, \gamma^T)$ is calculated (see Additional file 1). We approximate the profile information for $\theta - \lambda$ by $I_n^{-1/2}(\hat{\theta} - \lambda_0)$, where $I_n(\theta - \lambda_0)$ is the observed empirical information [24] given by

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

$s_i(\theta - \lambda)$ is the conditional expectation of the complete-data profile score for subject $i$, $S(\theta - \lambda)$ is the score defined by $S(\theta - \lambda) = \sum_{i=1}^n s_i(\theta - \lambda)$, and $a \otimes^2 = a a^\top$ is outer product for a vector $a$. At the maximizer, $S(\theta) = 0$, 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. [34], since the profiling was implicit; that is, because the posterior expectations involve the parameters \( \theta \).

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](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** [36].

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 [25] and survival analysis [26]. 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** [28], **JM** [27], or **frailtypack** [37] 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 [28, 27] and recurrent events [37].

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, \ldots, 5), with each model including subject-and-outcome-specific random-intercepts and random-slopes: \( \mathbf{b}_i = (b_{0i1}, b_{1i1}, b_{0i2}, b_{1i2})^\top \), Correl-
These findings are broadly in agreement with the simulation study conducted here, and other studies (n = 100 subjects per dataset). However, in this case we used a heavier-tailed distribution for the random effects: a multivariate t5 distribution [39]. 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. [40] 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 [41, 42]. Choi et al. [43] provide a review of existing research on the misspecification of random effects in joint modelling.

\[
y_{ijk} = (\beta_{0,k} + b_{00k}) + (\beta_{1,k} + b_{11k})t_j + \beta_{2,k}x_{i1} + \beta_{3,k}x_{i2} + \varepsilon_{ijk}, \text{ for } k = 1, 2;
\]
\[
\lambda_i(t) = \exp\{ (\theta_0 + \theta_1 t) + \gamma_{v1}x_{i1} + \gamma_{v2}x_{i2} + \gamma_{y1}(b_{001} + b_{111}t) + \gamma_{y2}(b_{002} + b_{112}t) \};
\]
\[
b_i \sim N(0, D);
\]
\[
\varepsilon_{ijk} \sim N(0, \sigma_k^2),
\]

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 t5 distribution [39]. 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. [40] 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.
Example

We consider the primary biliary cirrhosis (PBC) data collected at the Mayo Clinic between 1974 to 1984 [44]. This dataset has been widely analyzed using joint modelling methods [45, 46, 17]. 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 bilirubin (denoted \( \text{serBilir} \) in the model and data; measured in units of mg/dl), serum albumin (\( \text{albumin} \); mg/dl), and prothrombin time (\( \text{prothrombin} \); seconds). Patients had a mean 6.3 (SD = 3.7) visits (including baseline). The data can be accessed from the \texttt{joineRML} package via the command \texttt{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:

\[
\log(\text{serBilir}) = (\beta_{0,1} + b_{0i,1}) + (\beta_{1,1} + b_{1i,1})\text{year} + \epsilon_{ij1},
\]

\[
\text{albumin} = (\beta_{0,2} + b_{0i,2}) + (\beta_{1,2} + b_{1i,2})\text{year} + \epsilon_{ij2},
\]

\[
(0.1 \times \text{prothrombin})^{-4} = (\beta_{0,3} + b_{0i,3}) + (\beta_{1,3} + b_{1i,3})\text{year} + \epsilon_{ij3},
\]

\[
b_i \sim N_6(0, D), \text{ and } \epsilon_{ijk} \sim N(0, \sigma_k^2) \text{ for } k = 1, 2, 3;
\]

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

\[
\lambda_i(t) = \lambda_0(t) \exp\{\gamma_v\text{age} + W_{2i}(t)\},
\]

\[
W_{2i}(t) = \gamma_{b11}(b_{0i,1} + b_{1i,1}t) + \gamma_{a12}(b_{0i,2} + b_{1i,2}t) + \gamma_{pro}(b_{0i,3} + b_{1i,3}t).
\]

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 \texttt{lme()} function from the R package \texttt{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.

```r
# 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 \( \epsilon_0 \) 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 [45] 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 $\gamma_v$ were estimated, with the average comparable to the multivariate model estimate. The multivariate model estimates of $\gamma_y = (\gamma_{bil}, \gamma_{alb}, \gamma_{pro})^T$ were substantially attenuated relative to the separate model estimates, although the directions remained consistent. It is also interesting to note that $\gamma_{pro}$ 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 $\gamma$ 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.

```r
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 $\gamma$-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

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. [19]. 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, Bates 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. [47] 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 [14], albeit at the cost of additional computational time. Despite a growing availability of software for univariate joint models, Hickey et al. [18] 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 [48], rstanarm [49], and the Stata package stjm [46]. 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. [50]), 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** [27] and Stata package **stjm** [46]. Future developments of **joineRML** intend to cover several deficiencies. First, **joineRML** currently only permits an association structure of the form \( W_{2i}(t) = \sum_{k=1}^{K} \gamma_k W_{1i}^{(k)}(t) \). 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 [17]. 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 [28, 27] and recurrent events [37] 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 [51]. 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 [52].

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/graeemeleehickey/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 ≥ 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).

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

References


Additional Files

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`.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>formLongFixed</code></td>
<td>a list of formulae for the fixed effects component of each longitudinal</td>
</tr>
<tr>
<td></td>
<td>outcome. The left hand-hand side defines the response, and the right-</td>
</tr>
<tr>
<td></td>
<td>hand side specifies the fixed effect terms.</td>
</tr>
<tr>
<td><code>formLongRandom</code></td>
<td>a list of one-sided formulae specifying the model for the random effects</td>
</tr>
<tr>
<td></td>
<td>effects of each longitudinal outcome.</td>
</tr>
<tr>
<td><code>formSurv</code></td>
<td>a formula specifying the proportional hazards regression model (not</td>
</tr>
<tr>
<td></td>
<td>including the latent association structure).</td>
</tr>
<tr>
<td><code>data</code></td>
<td>a list of data.frame objects for each longitudinal outcome in which to</td>
</tr>
<tr>
<td></td>
<td>interpret the variables named in the <code>formLongFixed</code> and <code>formLongRandom</code>.</td>
</tr>
<tr>
<td></td>
<td>The list structure enables one to include multiple longitudinal outcomes</td>
</tr>
<tr>
<td></td>
<td>with different measurement protocols. If the multiple longitudinal</td>
</tr>
<tr>
<td></td>
<td>outcomes are measured at the same time points for each patient (i.e.</td>
</tr>
<tr>
<td></td>
<td>( t_{ijk} = t_{ij} \forall k )), then a single data.frame object can be</td>
</tr>
<tr>
<td></td>
<td>given instead of a list. It is assumed that each data frame is in long</td>
</tr>
<tr>
<td></td>
<td>format.</td>
</tr>
<tr>
<td><code>survData</code></td>
<td>(optional) a data.frame in which to interpret the variables named in the</td>
</tr>
<tr>
<td></td>
<td><code>formSurv</code>. If <code>survData</code> is not given, then <code>mjoint()</code> looks for the</td>
</tr>
<tr>
<td></td>
<td>time-to-event data in <code>data</code>.</td>
</tr>
<tr>
<td><code>timeVar</code></td>
<td>a character string indicating the time variable in the linear mixed effects</td>
</tr>
<tr>
<td><code>inits</code></td>
<td>(optional) a list of initial values for some or all of the parameters</td>
</tr>
<tr>
<td></td>
<td>estimated in the model.</td>
</tr>
<tr>
<td><code>control</code></td>
<td>(optional) a list of control parameters. These allow for the control of</td>
</tr>
<tr>
<td></td>
<td>( \epsilon_0, \epsilon_1, ) and ( \epsilon_2 ) in (7) and (8); the</td>
</tr>
<tr>
<td></td>
<td>choice of ( N, \delta, ) and convergence criteria; the maximum number</td>
</tr>
<tr>
<td></td>
<td>of MCEM iterations, and the minimum number of MCEM iterations during burn-</td>
</tr>
<tr>
<td></td>
<td>in. Additionally, the control argument <code>gammaOpt</code> can be used to specify</td>
</tr>
<tr>
<td></td>
<td>whether a one-step Newton-Raphson (&quot;NR&quot;) or Gauss-Newton-like (&quot;GN&quot;)</td>
</tr>
<tr>
<td></td>
<td>update should be used for the M-step update of ( \gamma ).</td>
</tr>
</tbody>
</table>

†`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†.

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

†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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>True value</th>
<th>Mean estimated value</th>
<th>Empirical SE</th>
<th>Mean SE</th>
<th>Bias</th>
<th>MSE</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>D11</td>
<td>0.2500</td>
<td>0.2411</td>
<td>0.0435</td>
<td>—</td>
<td>0.0089</td>
<td>0.0020</td>
<td>—</td>
</tr>
<tr>
<td>D21</td>
<td>0.0000</td>
<td>0.0010</td>
<td>0.0136</td>
<td>—</td>
<td>0.0010</td>
<td>0.0002</td>
<td>—</td>
</tr>
<tr>
<td>D31</td>
<td>-0.1250</td>
<td>-0.1212</td>
<td>0.0295</td>
<td>—</td>
<td>0.0038</td>
<td>0.0009</td>
<td>—</td>
</tr>
<tr>
<td>D41</td>
<td>0.0000</td>
<td>0.0007</td>
<td>0.0127</td>
<td>—</td>
<td>0.0006</td>
<td>0.0002</td>
<td>—</td>
</tr>
<tr>
<td>D22</td>
<td>0.0400</td>
<td>0.0396</td>
<td>0.0072</td>
<td>—</td>
<td>0.0004</td>
<td>0.0001</td>
<td>—</td>
</tr>
<tr>
<td>D32</td>
<td>0.0000</td>
<td>0.0002</td>
<td>0.0138</td>
<td>—</td>
<td>0.0002</td>
<td>0.0002</td>
<td>—</td>
</tr>
<tr>
<td>D42</td>
<td>0.0000</td>
<td>0.0001</td>
<td>0.0055</td>
<td>—</td>
<td>0.0001</td>
<td>0.0000</td>
<td>—</td>
</tr>
<tr>
<td>D33</td>
<td>0.2500</td>
<td>0.2420</td>
<td>0.0075</td>
<td>—</td>
<td>0.0080</td>
<td>0.0017</td>
<td>—</td>
</tr>
<tr>
<td>D43</td>
<td>0.0000</td>
<td>0.0007</td>
<td>0.0134</td>
<td>—</td>
<td>0.0007</td>
<td>0.0002</td>
<td>—</td>
</tr>
<tr>
<td>D44</td>
<td>0.0400</td>
<td>0.0399</td>
<td>0.0075</td>
<td>—</td>
<td>0.0001</td>
<td>0.0001</td>
<td>—</td>
</tr>
<tr>
<td>β0,1</td>
<td>0.0000</td>
<td>0.0028</td>
<td>0.0612</td>
<td>0.0660</td>
<td>0.0028</td>
<td>0.0038</td>
<td>0.9660</td>
</tr>
<tr>
<td>β1,1</td>
<td>1.0000</td>
<td>1.0012</td>
<td>0.0218</td>
<td>0.0229</td>
<td>0.0012</td>
<td>0.0005</td>
<td>0.9500</td>
</tr>
<tr>
<td>β2,1</td>
<td>1.0000</td>
<td>1.0010</td>
<td>0.0449</td>
<td>0.0470</td>
<td>0.0010</td>
<td>0.0020</td>
<td>0.9540</td>
</tr>
<tr>
<td>β3,1</td>
<td>1.0000</td>
<td>0.9932</td>
<td>0.0897</td>
<td>0.0925</td>
<td>-0.0068</td>
<td>0.0081</td>
<td>0.9440</td>
</tr>
<tr>
<td>σ12</td>
<td>0.2500</td>
<td>0.2506</td>
<td>0.0165</td>
<td>0.0171</td>
<td>0.0006</td>
<td>0.0003</td>
<td>0.9560</td>
</tr>
<tr>
<td>β0,2</td>
<td>0.0000</td>
<td>-0.0026</td>
<td>0.0637</td>
<td>0.0655</td>
<td>-0.0026</td>
<td>0.0041</td>
<td>0.9660</td>
</tr>
<tr>
<td>β1,2</td>
<td>-1.0000</td>
<td>-1.0011</td>
<td>0.0229</td>
<td>0.0223</td>
<td>-0.0011</td>
<td>0.0005</td>
<td>0.9480</td>
</tr>
<tr>
<td>β2,2</td>
<td>0.0000</td>
<td>0.0008</td>
<td>0.0399</td>
<td>0.0472</td>
<td>0.0008</td>
<td>0.0016</td>
<td>0.9700</td>
</tr>
<tr>
<td>β3,2</td>
<td>0.5000</td>
<td>0.5061</td>
<td>0.0894</td>
<td>0.0923</td>
<td>0.0061</td>
<td>0.0080</td>
<td>0.9540</td>
</tr>
<tr>
<td>σ22</td>
<td>0.2500</td>
<td>0.2501</td>
<td>0.0162</td>
<td>0.0171</td>
<td>0.0001</td>
<td>0.0003</td>
<td>0.9540</td>
</tr>
<tr>
<td>γ11</td>
<td>0.0000</td>
<td>0.0011</td>
<td>0.1243</td>
<td>0.1392</td>
<td>0.0011</td>
<td>0.0155</td>
<td>0.9720</td>
</tr>
<tr>
<td>γ12</td>
<td>1.0000</td>
<td>1.0487</td>
<td>0.2837</td>
<td>0.2750</td>
<td>0.0487</td>
<td>0.0829</td>
<td>0.9340</td>
</tr>
<tr>
<td>γ21</td>
<td>-0.5000</td>
<td>-0.5121</td>
<td>0.1936</td>
<td>0.2084</td>
<td>-0.0121</td>
<td>0.0376</td>
<td>0.9560</td>
</tr>
<tr>
<td>γ22</td>
<td>1.0000</td>
<td>1.0311</td>
<td>0.2220</td>
<td>0.2145</td>
<td>0.0311</td>
<td>0.0502</td>
<td>0.9400</td>
</tr>
</tbody>
</table>
Table 4 Fitted multivariate and separate univariate joint models to the PBC data.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>joineRML (NR)</th>
<th>joineRML (GN)</th>
<th>Bootstrap</th>
<th>joineRML (NR)</th>
<th>joineRML (GN)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Estimate</td>
<td>SE</td>
<td>95% CI†</td>
<td>Estimate</td>
<td>SE</td>
</tr>
<tr>
<td>( \beta_{0,1} )</td>
<td>0.5541</td>
<td>0.0858</td>
<td>(0.3859, 0.7223)</td>
<td>0.5549</td>
<td>0.0846</td>
</tr>
<tr>
<td>( \beta_{1,1} )</td>
<td>0.2009</td>
<td>0.0201</td>
<td>(0.1616, 0.2402)</td>
<td>0.2008</td>
<td>0.0201</td>
</tr>
<tr>
<td>( \beta_{0,2} )</td>
<td>3.5549</td>
<td>0.0356</td>
<td>(3.4850, 3.6248)</td>
<td>3.5546</td>
<td>0.0357</td>
</tr>
<tr>
<td>( \beta_{1,2} )</td>
<td>-0.1245</td>
<td>0.0101</td>
<td>(-0.1444, -0.1047)</td>
<td>-0.1246</td>
<td>0.0101</td>
</tr>
<tr>
<td>( \beta_{0,3} )</td>
<td>0.8304</td>
<td>0.0212</td>
<td>(0.7888, 0.8719)</td>
<td>0.8301</td>
<td>0.0210</td>
</tr>
<tr>
<td>( \beta_{1,3} )</td>
<td>-0.0577</td>
<td>0.0062</td>
<td>(-0.0699, -0.0456)</td>
<td>-0.0577</td>
<td>0.0062</td>
</tr>
<tr>
<td>( \gamma_v )</td>
<td>0.0462</td>
<td>0.0151</td>
<td>(0.0166, 0.0759)</td>
<td>0.0462</td>
<td>0.0152</td>
</tr>
<tr>
<td>( \gamma_{\text{bil}} )</td>
<td>0.8181</td>
<td>0.2046</td>
<td>(0.4171, 1.2191)</td>
<td>0.8187</td>
<td>0.2036</td>
</tr>
<tr>
<td>( \gamma_{\text{alb}} )</td>
<td>-1.7060</td>
<td>0.6181</td>
<td>(-2.9173, -0.4946)</td>
<td>-1.6973</td>
<td>0.6163</td>
</tr>
<tr>
<td>( \gamma_{\text{pro}} )</td>
<td>-2.2085</td>
<td>1.6070</td>
<td>(-5.3582, 0.9412)</td>
<td>-2.2148</td>
<td>1.6133</td>
</tr>
</tbody>
</table>

Notation: SE = standard error; CI = confidence interval; NR = one-step Newton-Raphson update for \( \gamma \); GN = one-step Gauss-Newton-like update for \( \gamma \). †SEs are calculated from the inverse profile empirical information matrix, and confidence intervals are based on normal approximations of the type \( \hat{\theta} \pm 1.96 \text{SE}(\hat{\theta}) \), where \( \hat{\theta} \) 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. aSeparate model fit for \( \text{serBilir} \). bSeparate model fit for albumin. cSeparate model fit for prothrombin.
Figure 1

Click here to download Figure traj_plot.pdf
Click here to access/download
Supplementary Material
appendix.pdf