Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

The R Package JMbayes for Fitting Joint Models for Longitudinal and Time-to-Event Data Using MCMC

Dimitris Rizopoulos
Erasmus Medical Center Rotterdam

Abstract

Joint models for longitudinal and time-to-event data constitute an attractive modeling framework that has received a lot of interest in the recent years. This paper presents the capabilities of the R package **JMbayes** for fitting these models under a Bayesian approach using Markov chain Monte Carlo algorithms. **JMbayes** can fit a wide range of joint models, including among others joint models for continuous and categorical longitudinal responses, and provides several options for modeling the association structure between the two outcomes. In addition, this package can be used to derive dynamic predictions for both outcomes, and offers several tools to validate these predictions in terms of discrimination and calibration. All these features are illustrated using a real data example on patients with primary biliary cirrhosis.

Keywords: survival analysis, time-varying covariates, random effects, mixed models, dynamic predictions, validation.

1. Introduction

Joint models for longitudinal and time-to-event data constitute an attractive modeling paradigm that currently enjoys great interest in the statistics and medical literature (Rizopoulos and Lesaffre 2014; Rizopoulos 2012; Tsiatis and Davidian 2004). These models are utilized in follow-up studies where interest is in associating a longitudinal response with an event time outcome. In general, there are mainly two settings in which such type of models are required. First, when one is interested in measuring the strength of the association between the hazard of an event and a time-varying covariate, then we should pay special attention to the attributes of the covariate process. In particular, when this is an endogenous time-varying covariate (Kalbfleisch and Prentice 2002, Section 6.3), standard methods, such as the time-dependent

Cox model (Therneau and Grambsch 2000), are not optimal for measuring this association. Endogenous covariates are covariates, which are measured on the sample units themselves, for instance, biomarkers or other parameters measured on patients during follow-up. The two important features of such covariates are that their existence and/or future path is directly related to the event status, and that they are measured with error. This could be measurement error originating from procedure/test used to measure the longitudinal outcome and/or biological variation. By postulating a model for the joint distribution of the covariate and event processes we explicitly acknowledge for both of these features, and hence we obtain a more accurate estimate for their association. The second case in which joint models are of use is when one needs to account for incomplete data in the longitudinal outcome. More specifically, when the probability of missingness depends on unobserved longitudinal responses, then in order to obtain valid inferences we need to postulate a model for the joint distribution of the longitudinal and missingness processes (Little and Rubin 2002; Molenberghs and Kenward 2007). In this context, three main frameworks have been proposed to define such joint distributions, namely, selection, pattern mixture and shared parameter models. The majority of the models that have been proposed in the literature under these frameworks have focused on standard designs assuming a fixed set of time points at which subjects are expected to provide measurements. Nonetheless, in reality, subjects often do not adhere to the posited study schedule and may skip visits and dropout from the study at random time points. Even though in many of those occasions information on the exact dropout time is available, the typical convention in selection and pattern mixture modeling has been to ignore this feature and coerce measurements back to discrete follow-up times. Another alternative that makes better use of the available data is to acknowledge that dropout occurs in continuous time and consider it as a time-to-event outcome.

Following the increasing use of these models, currently there are several software implementations available to fit them. The R package **JM** Rizopoulos (2014, 2012, 2010) fits joint models for a continuous longitudinal outcome and an event time process under maximum likelihood. Several types of association structures are supported and the package also allows to fit joint models with competing risk survival data. In addition, **JM** can be used to calculate dynamic predictions for either of the two outcomes. The R package **joineR** (Philipson, Sousa, Diggle, Williamson, Kolamunnage-Dona, and Henderson 2012) similarly fits joint models for a continuous longitudinal outcome and a time-to-event, following the formulation of Henderson, Diggle, and Dobson (2000). In addition, the **stjm** package for STATA (Crowther 2013; Crowther, Abrams, and Lambert 2013) implements joint modeling of a normal longitudinal response and a time-to-event using maximum likelihood, with emphasis on parametric time-to-event models. The implementation of joint models in SAS and **WinBUGS** has been discussed by Guo and Carlin (2004). Finally, contrary to the previous software implementations, function `Jointlcmm()` from the R package **lcmm** (Proust-Lima, Philipps, Diakite, and Liquet 2013) fits joint latent class models for a continuous longitudinal outcome and a survival outcome using maximum likelihood; these models postulate that the association between the two processes is captured by categorical random effects (i.e., latent classes).

In this paper we introduce the R package **JMbayes**, available from CRAN (`http://cran.r-project.org/package=JMbayes`), that fits joint models under a Bayesian approach. Compared to the aforementioned packages, **JMbayes** can fit a wider range of joint models that also includes joint models for categorical or censored longitudinal responses. In addition, it provides greater flexibility for modeling the association structure between the two outcomes,

with the possibility of different terms from the longitudinal submodel entering the linear predictor of the survival submodel, while also allowing for general transformations of these terms. Moreover, the package provides more extensive capabilities to derive dynamic predictions for both outcomes compared to package **JM**, it allows to combine predictions from different models using innovative Bayesian model averaging techniques, and facilitates the utilization of these predictions in practice using a web interface. Furthermore, it offers more extensive tools to quantify the quality of these predictions in terms of discrimination and calibration, and code is provided for their validation. The rest of the paper is organised as follows. Section 2 presents a short review of the underlying methodological framework behind joint models. Section 3 gives the details behind the implementation of joint models in package **JMbayes**, and Section 4 illustrates in detail the use of the package in a real dataset on patients with primary biliary cirrhosis. Finally, Section 5 presents how dynamic predictions for the longitudinal and event time outcomes are defined, and how they can be calculated and validated with the package.

2. Theoretical framework

Let $\mathcal{D}_n = \{T_i, \delta_i, \boldsymbol{y}_i; i = 1, \ldots, n\}$ denote a sample from the target population, where T_i^* denotes the true event time for the i-th subject , C_i the censoring time, $T_i = \min(T_i^*, C_i)$ the corresponding observed event time, and $\delta_i = I(T_i^* \leq C_i)$ the event indicator, with $I(\cdot)$ being the indicator function that takes the value 1 when $T_i^* \leq C_i$, and 0 otherwise. In addition, we let $\boldsymbol{y}_i$ denote the $n_i \times 1$ longitudinal response vector for the i-th subject, with element y_{il} denoting the value of the longitudinal outcome taken at time point t_{il}, $l = 1, \ldots, n_i$.

To accommodate different types of longitudinal responses in a unified framework, we postulate a generalized linear mixed effects model. In particular, the conditional distribution of $\boldsymbol{y}_i$ given a vector of random effects $\boldsymbol{b}_i$ is assumed to be a member of the exponential family, with linear predictor given by

$$g\big[E\{y_i(t) \mid \boldsymbol{b}_i\}\big] = \eta_i(t) = \boldsymbol{x}_i^\top(t)\boldsymbol{\beta} + \boldsymbol{z}_i^\top(t)\boldsymbol{b}_i, \tag{1}$$

where $g(\cdot)$ denotes a known one-to-one monotonic link function, and $y_i(t)$ denotes the value of the longitudinal outcome for the i-th subject at time point t, $\boldsymbol{x}_i(t)$ and $\boldsymbol{z}_i(t)$ denote the time-dependent design vectors for the fixed-effects $\boldsymbol{\beta}$ and for the random effects $\boldsymbol{b}_i$, respectively. The random effects are assumed to follow either a multivariate normal distribution with mean zero and variance-covariance matrix $\boldsymbol{D}$ or a multivariate Student's-t distribution with mean zero, scale covariance matrix $\boldsymbol{D}$ and df degrees of freedom. For the survival process, we assume that the risk for an event depends on a function of the subject-specific linear predictor $\eta_i(t)$. More specifically, we have

$$\begin{aligned} h_i(t \mid \mathcal{H}_i(t), \boldsymbol{w}_i(t)) &= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \Pr\{t \leq T_i^* < t + \Delta t \mid T_i^* \geq t, \mathcal{H}_i(t), \boldsymbol{w}_i(t)\} \\ &= h_0(t) \exp\big[\boldsymbol{\gamma}^\top \boldsymbol{w}_i(t) + f\{\mathcal{H}_i(t), \boldsymbol{b}_i, \boldsymbol{\alpha}\}\big], \quad t > 0, \end{aligned} \tag{2}$$

where $\mathcal{H}_i(t) = \{\eta_i(s), 0 \leq s < t\}$ denotes the history of the underlying longitudinal process up to t, $h_0(\cdot)$ denotes the baseline hazard function, $\boldsymbol{w}_i(t)$ is a vector of exogenous, possibly time-varying, covariates with corresponding regression coefficients $\boldsymbol{\gamma}$. Parameter vector $\boldsymbol{\alpha}$ quantifies the association between features of the marker process up to time t and the hazard of an event at the same time point. Various options for the form of function $f(\cdot)$ are presented in

Section 4.3. To complete the specification of the survival process we need to make appropriate assumptions for the baseline hazard function $h_0(\cdot)$. To model this function, while still allowing for flexibility, we use a B-splines approach. In particular, the logarithm of the baseline hazard function is expressed as

$$\log h_0(t) = \gamma_{h_0,0} + \sum_{q=1}^{Q} \gamma_{h_0,q} B_q(t, \boldsymbol{v}), \tag{3}$$

where $B_q(t, \boldsymbol{v})$ denotes the q-th basis function of a B-spline with knots $v_1, \ldots, v_Q$ and $\boldsymbol{\gamma}_{h_0}$ the vector of spline coefficients. Increasing the number of knots Q increases the flexibility in approximating $\log h_0(\cdot)$; however, we should balance bias and variance and avoid over-fitting. A standard rule of thumb is to keep the total number of parameters, including the parameters in the linear predictor in (2) and in the model for $h_0(\cdot)$, between 1/10 and 1/20 of the total number of events in the sample (Harrell 2001, Section 4.4). After the number of knots has been decided, their location can be based on percentiles of the observed event times T_i or of the true event times $\{T_i : T_i^* \leq C_i, i = 1, \ldots, n\}$ in order to allow for more flexibility in the region of greatest density. A standard alternative approach that avoids the task of choosing the appropriate number and position of the knots is to include a relatively high number of knots (e.g., 15 to 20) and appropriately penalize the B-spline regression coefficients $\boldsymbol{\gamma}_{h_0}$ for smoothness (Eilers and Marx 1996).

Under the Bayesian approach, estimation of joint model's parameters proceeds using Markov chain Monte Carlo (MCMC) algorithms. The expression for the posterior distribution of the model parameters is derived under the assumptions that given the random effects, both the longitudinal and event time process are assumed independent, and the longitudinal responses of each subject are assumed independent. Formally we have,

$$p(\boldsymbol{y}_i, T_i, \delta_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}) = p(\boldsymbol{y}_i \mid \boldsymbol{b}_i, \boldsymbol{\theta})\, p(T_i, \delta_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}), \tag{4}$$

$$p(\boldsymbol{y}_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}) = \prod_l p(y_{il} \mid \boldsymbol{b}_i, \boldsymbol{\theta}), \tag{5}$$

where $\boldsymbol{\theta}$ denotes the full parameter vector, and $p(\cdot)$ denotes an appropriate probability density function. Under these assumptions the posterior distribution is analogous to:

$$p(\boldsymbol{\theta}, \boldsymbol{b}) \propto \prod_{i=1}^{n} \prod_{l=1}^{n_i} p(y_{il} \mid \boldsymbol{b}_i, \boldsymbol{\theta})\, p(T_i, \delta_i \mid \boldsymbol{b}_i, \boldsymbol{\theta})\, p(\boldsymbol{b}_i \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{6}$$

where

$$p(y_{il} \mid \boldsymbol{b}_i, \boldsymbol{\theta}) = \exp\Big\{ \Big[y_{il}\psi_{il}(\boldsymbol{b}_i) - c\{\psi_{il}(\boldsymbol{b}_i)\} \Big] \Big/ a(\varphi) - d(y_{il}, \varphi) \Big\},$$

with $\psi_{il}(\boldsymbol{b}_i)$ and φ denoting the natural and dispersion parameters in the exponential family, respectively, $c(\cdot)$, $a(\cdot)$, and $d(\cdot)$ are known functions specifying the member of the exponential family, and for the survival part

$$p(T_i, \delta_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}) = h_i(T_i \mid \mathcal{H}_i(T_i))^{\delta_i} \exp\Big\{ - \int_0^{T_i} h_i(s \mid \mathcal{H}_i(s))\, ds \Big\},$$

with $h_i(\cdot)$ given by (2). The integral in the definition of the survival function

$$S_i(t \mid \mathcal{H}_i(t), \boldsymbol{w}_i(t)) = \exp\Big\{ - \int_0^{t} h_0(s) \exp\big[\boldsymbol{\gamma}^\top \boldsymbol{w}_i(t) + f\{\mathcal{H}_i(s), \boldsymbol{\alpha}\} \big] ds \Big\}, \tag{7}$$

does not have a closed-form solution, and thus a numerical method must be employed for its evaluation. Standard options are the Gauss-Kronrod and Gauss-Legendre quadrature rules.

For the parameters $\boldsymbol{\theta}$ we take standard prior distributions. In particular, for the vector of fixed effects of the longitudinal submodel $\boldsymbol{\beta}$, for the regression parameters of the survival model $\boldsymbol{\gamma}$, for the vector of spline coefficients for the baseline hazard $\boldsymbol{\gamma}_{h_0}$, and for the association parameter $\boldsymbol{\alpha}$ we use independent univariate diffuse normal priors. The penalized version of the B-spline approximation to the baseline hazard can be fitted by specifying for $\boldsymbol{\gamma}_{h_0}$ the improper prior (Lang and Brezger 2004):

$$p(\boldsymbol{\gamma}_{h_0} \mid \tau_h) \propto \tau_h^{\rho(K)/2} \exp\Big(-\frac{\tau_h}{2}\boldsymbol{\gamma}_{h_0}^\top \boldsymbol{K}\boldsymbol{\gamma}_{h_0}\Big),$$

where τ_h is the smoothing parameter that takes a Gamma$(1, 0.005)$ hyper-prior in order to ensure a proper posterior for $\boldsymbol{\gamma}_{h_0}$, $\boldsymbol{K} = \Delta_r^\top \Delta_r$, where Δ_r denotes r-th difference penalty matrix, and $\rho(\boldsymbol{K})$ denotes the rank of $\boldsymbol{K}$. For the covariance matrix of the random effects we assume an inverse Wishart prior, and when fitting a joint model with a normally distributed longitudinal outcome, we take an inverse-Gamma prior for the variance of the error terms σ^2. More details regarding Bayesian estimation of joint models can be found in Ibrahim, Chen, and Sinha (2001, Chapter 7) and Brown, Ibrahim, and DeGruttola (2005).

3. The R package JMbayes

3.1. Design

In many regards the design of package **JMbayes** is similar to the one of package **JM** for fitting joint models under maximum likelihood. In particular, **JMbayes** has a basic model-fitting function called `jointModelBayes()`, which accepts as main arguments a linear mixed effects object fit as returned by functions `lme()` of package **nlme** (Pinheiro, Bates, DebRoy, Sarkar, and R Development Core Team 2014) or from function `glmmPQL()` from package **MASS** (Venables and Ripley 2002), and a survival object fit as returned by function `coxph()` of package **survival** (Therneau and Lumley 2014). The final required argument is `timeVar`, a character string that denotes the name of the time variable in the mixed model. By default `jointModelBayes()` fits joint models with a linear mixed effects submodel for a continuous longitudinal outcome, and a relative risk submodel of the form (2) with $f\{\mathcal{H}_i(t), \boldsymbol{b}_i, \boldsymbol{\alpha}\} = \alpha\eta_i(t)$, i.e., the risk for an event at time t is associated with the subject-specific mean of the longitudinal outcome at the same time point. Joint models for other types of longitudinal outcomes can be fitted by appropriately specifying argument `densLong`, and arguments `param`, `extraForm` and `transFun` can be used to add extra terms involving components of the longitudinal process and possibly transform these terms. A detailed account on the use of these arguments, with examples, is given in Section 4. The baseline hazard is by default approximated using penalized B-splines; regression splines can be instead invoked by appropriately setting argument `baseHaz`. The number and position of the knots can be controlled via the `lng.in.kn` and `knots` control arguments. The former defines the number of knots to use (by default placed at equally spaced percentiles of the observed event times), whereas argument `knots` can be invoked to specify knots at specific positions. The type of numerical integration algorithm used to approximate the survival function (7) is specified with the control argument `GQsurv` with options `"GaussKronrod"` (default) and `"GaussLegendre"`,

while the number of quadrature points is specified using the control argument `GQsurv.k` (for the Gauss-Kronrod rule only 7 or 15 can be specified). The fitting process can be further appropriately tweaked using a series of extra control arguments explained in the following section and in the help file of `jointModelBayes()`. In addition, the default values of the parameters of the prior distributions can be altered using the `priors` argument, and analogously the default initial values using argument `init`.

3.2. Implementation details

The MCMC algorithm that samples from the posterior conditional distributions of the parameters and the random effects is implemented by the internal function `MCMCfit()`. For the majority of the posterior conditionals random walk Metropolis is used, with exceptions for the precision parameter of the error terms distribution when a linear mixed model is used for the longitudinal outcome in which case slice sampling is used, and for the random effects precision matrix $\boldsymbol{D}^{-1}$ in which case when the random effects are assumed normally distributed the posterior conditional is a Wishart distribution (if argument `df.RE` of `jointModelBayes()` is not `NULL` the distribution of the random effects is assumed to be a Student's-t distribution with `df.RE` degrees of freedom; in this case the random effects precision matrix is updated with a Metropolis-Hastings algorithm). The implementation behind `MCMCfit()` takes full advantage of the separately fitted mixed effects and Cox models in order to appropriately define the covariance matrix of the normal proposal distributions for the random walk Metropolis algorithm. In particular, for $\boldsymbol{\beta}$ and $\boldsymbol{b}_i$ these covariance matrices are taken from the mixed model, whereas for the regression coefficients in the linear predictor of the survival submodel and the B-spline coefficients $\boldsymbol{\gamma}_{h_0}$ a two-stage approach is employed, where a time-dependent Cox model is fitted using the mixed model to compute $f\{\mathcal{H}_i(t), \boldsymbol{b}_i, \boldsymbol{\alpha}\}$. These proposal distributions are tuned during an adaptive phase of `n.adapt` iterations (default 3000), where every `n.batch` iterations (default 100) the acceptance rates of the algorithms are checked. Following this a burn-in period of `n.burnin` iterations (default 3000) is performed, and after these iterations the algorithm continues to run for an extra `n.iter` iterations (default 20000). The chains are thinned according to the `n.thin` argument (default is to keep 2000 iterations for each parameter in order to limit the memory size of the resulting joint model object). To improve the mixing of the chains hierarchical centering is used whenever the largest variance in the covariance matrix of the random effects $\boldsymbol{D}$ is greater than σ^2/n, with σ^2 denoting the measurement error variance and n the sample size.

From the two schools of running MCMC algorithms, namely the 'one long chain school' and the 'multiple shorter chains school', **JMbayes** implements the former. Users who wish to check convergence using multiple chains can still do it but with a bit of extra programming. More specifically, they could call `jointModelBayes()` with different initial values (by appropriately specifying argument `init`), and following they could extract component `mcmc` from the fitted models, which is the list of simulated values for each parameter. These lists could subsequently be processed using the **coda** package (Plummer, Best, Cowles, and Vines 2006) and perform these diagnostic tests.

3.3. Speed of computations and flexibility

As it is often the case with Bayesian computations, the time required to fit the model can be relatively long. Several experiments we have done indicate that for simple joint models (e.g.,

as the one presented in Section 4.1) the Bayesian estimation procedure based on **JMbayes** is slower that the maximum likelihood implementation based on **JM**. On the other hand, as the complexity of the model increases (e.g., the models presented in Section 4.3), then the time required to fit the model under the Bayesian approach versus maximum likelihood is roughly the same. However, the major advantage of the Bayesian approach compared to maximum likelihood is that it makes it relatively easy and straightforward to implement several types of extensions for joint models. For example, the models for categorical and censored longitudinal responses presented in Section 4.2 require much greater programming effort for their implementation under maximum likelihood than under the Bayesian approach. In addition, we have also performed comparisons of **JMbayes** using **WinBUGS**, **OpenBUGS** and **JAGS** (actually earlier versions of **JMbayes** were running the MCMC using **JAGS**). Even though the MCMC algorithm in **JMbayes** is written in pure R code, it is many times faster than the aforementioned programs. The main reasons behind this counter-intuitive result (e.g., compared to **JAGS** which is written in C++) is the fact that in order to implement joint models in these programs it is required to use the so-called 'zeros-trick' for specifying a new sampling distribution, because the survival submodel is not of a standard form. In addition, it is also required to implement within the **WinBUGS** or **JAGS** program the numerical integration with respect to time required to approximate the survival function (7). Both of these features make fitting joint models with these programs quite computationally intensive. In the following sections, where we present the basic capabilities of **JMbayes**, we also provide the time required to fit each joint model. These timings are based on R version 3.1.3 (2015-03-09) running on 64-bit Windows 7 Professional, in a laptop with an Intel i7-3840QM CPU @ 2.80 GHz and 8.0 GB RAM.

4. Practical use of JMbayes

4.1. The basic joint model

We will illustrate the capabilities of package **JMbayes** using the primary biliary cirrhosis (PBC) data collected by the Mayo Clinic from 1974 to 1984 (Murtaugh, Dickson, Van Dam, Malincho, Grambsch, Langworthy, and Gips 1994). PBC is a chronic, fatal, but rare liver disease characterized by inflammatory destruction of the small bile ducts within the liver, which eventually leads to cirrhosis of the liver. Patients with PBC have abnormalities in several blood tests, such as elevated levels of serum bilirubin. For our analysis we will consider 312 patients who have been randomized to D-penicillamine and 154 placebo. During follow-up several biomarkers associated with PBC have been collected for these patients. Here we focus on serum bilirubin levels, which is considered one of the most important ones associated with disease progression. Patients had on average 6.2 measurements (std. deviation 3.8 measurements), with a total of 1945 observations. In package **JMbayes** the PBC data are available in the data frames `pbc2` and `pbc2.id` containing the longitudinal and survival information, respectively (i.e., the former is in the long format while the latter contains a single row per patient).

We start by loading packages **JMbayes** and **lattice** (Sarkar 2008) and defining the indicator `status2` for the composite event, namely transplantation or death:

```
R> library("JMbayes")
R> library("lattice")
R> pbc2$status2 <- as.numeric(pbc2$status != "alive")
R> pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")
```

Descriptive plots for the survival and longitudinal outcomes are presented in Figures 1 and 2 that depict the Kaplan-Meier estimate of transplantation-free survival for the two treatment groups, and the sample subject-specific longitudinal trajectories for patients with and without the endpoint, respectively.

```
R> sfit <- survfit(Surv(years, status2) ~ drug, data = pbc2.id)
R> plot(sfit, lty = 1:2, lwd = 2, col = 1:2, mark.time = FALSE,
+    xlab = "Time (years)", ylab = "Transplantation-free Survival")
R> legend("topright", levels(pbc2.id$drug), lty = 1:2, col = 1:2, lwd = 2,
+    cex = 1.3, bty = "n")
```

```
R> pbc2$status2f <- factor(pbc2$status2, levels = 0:1,
+    labels = c("alive", "transplanted/dead"))
R> xyplot(log(serBilir) ~ year | status2f, group = id, data = pbc2,
+    panel = function (x, y, ...) {
+      panel.xyplot(x, y, type = "l", col = 1, ...)
+      panel.loess(x, y, col = 2, lwd = 2)
+    }, xlab = "Time (years)", ylab = "log(serum Bilirubin)")
```

We continue by separately fitting a linear mixed model for the longitudinal and a Cox model for the survival one. Careful investigation of the shapes of the log serum bilirubin profiles indicates that for many individuals these seem to be nonlinear. Hence, to allow for flexibility in the specification of these profiles we include natural cubic splines in both the fixed- and random-effects parts of the mixed model. This model can be fitted using the following call to functions `lme()` and `ns()` (the latter from package **splines**):

```
R> lmeFit.pbc1 <- lme(log(serBilir) ~ ns(year, 2), data = pbc2,
+    random = ~ ns(year, 2) | id)
```

Analogously, in the Cox model we control for treatment and age, and also allow for their interaction:

```
R> coxFit.pbc1 <- coxph(Surv(years, status2) ~ drug * age, data = pbc2.id,
+    x = TRUE)
```

In the call to `coxph()` argument `x` is set to `TRUE` such that the design matrix is also included in the resulting model object. Using as main arguments the `lmeFit.pbc1` and `coxFit.pbc1` objects, the corresponding joint model is fitted using the code (time to fit: 4.4 min):

```
R> jointFit.pbc1 <- jointModelBayes(lmeFit.pbc1, coxFit.pbc1,
+    timeVar = "year", n.iter = 30000)
R> summary(jointFit.pbc1, include.baseHazCoefs = TRUE)
```

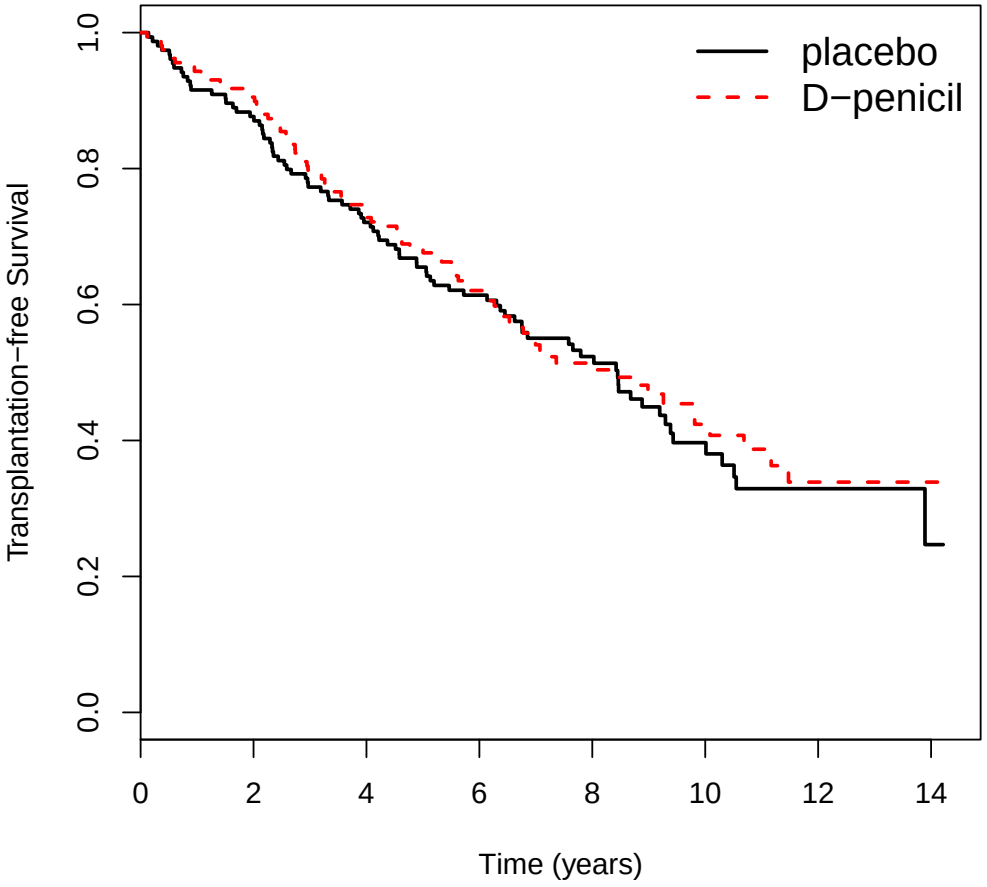

Figure 1: Kaplan-Meier estimator of transplantation-free survival probabilities for the two treatment groups.

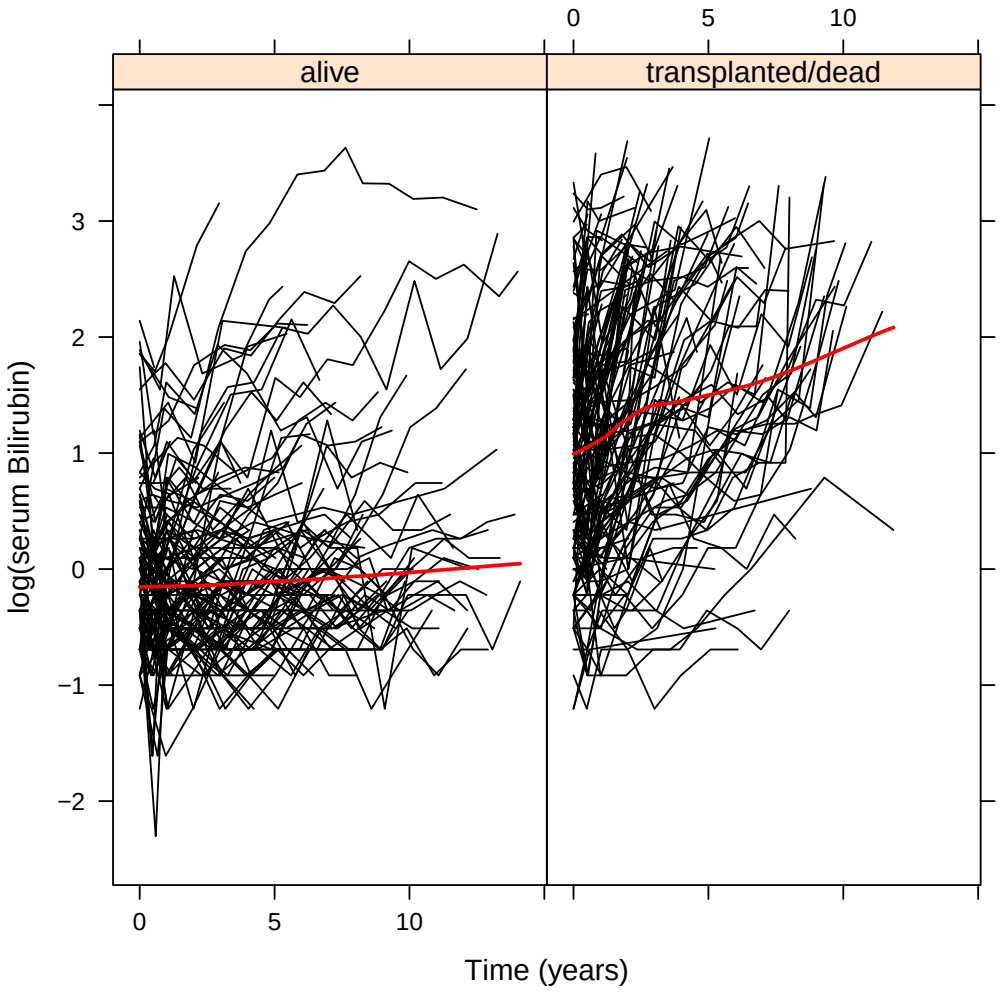

Figure 2: Subject-specific longitudinal trajectories for log serum bilirubin for patients with and without an event. The red line represents the loess smoother.

```
Call:
jointModelBayes(lmeObject = lmeFit.pbc1, survObject = coxFit.pbc1,
    timeVar = "year", n.iter = 30000)

Data Descriptives:
Longitudinal Process                Event Process
Number of Observations: 1945        Number of Events: 169 (54.2%)
Number of subjects: 312

Joint Model Summary:
Longitudinal Process: Linear mixed-effects model
Event Process: Relative risk model with penalized-spline-approximated
                baseline risk function
Parameterization: Time-dependent value

     LPML      DIC      pD
 -3165.22 6082.505 929.577

Variance Components:
              StdDev    Corr
(Intercept)   1.0061  (Intr)  n(,2)1
ns(year, 2)1  2.2246  0.3167
ns(year, 2)2  2.0783  0.2574  0.5050
Residual      0.3031

Coefficients:
Longitudinal Process
              Value Std.Err Std.Dev   2.5%  97.5%      P
(Intercept)  0.5115  0.0013  0.0578 0.3959 0.6234 <0.001
ns(year, 2)1 2.2841  0.0057  0.1484 2.0040 2.5831 <0.001
ns(year, 2)2 2.0898  0.0098  0.1521 1.7921 2.3801 <0.001

Event Process
                     Value Std.Err  Std.Dev     2.5%     97.5%      P
drugD-penicil      -0.7776  0.1205   0.7762  -2.2608    0.9006  0.309
age                 0.0404  0.0024   0.0126   0.0158    0.0668 <0.001
drugD-penicil:age   0.0125  0.0023   0.0153  -0.0196    0.0412  0.390
Assoct              1.4099  0.0045   0.0935   1.2317    1.6008 <0.001
Bs.gammas1         -6.8062  0.1233   0.7456  -8.3592   -5.3616 <0.001
Bs.gammas2         -6.7254  0.1125   0.7035  -8.1929   -5.3429 <0.001
Bs.gammas3         -6.6383  0.1057   0.6770  -8.0585   -5.3072 <0.001
Bs.gammas4         -6.5510  0.1020   0.6621  -7.9550   -5.2446 <0.001
Bs.gammas5         -6.4789  0.1016   0.6589  -7.8555   -5.1731 <0.001
Bs.gammas6         -6.4118  0.1178   0.6607  -7.8002   -5.1002 <0.001
Bs.gammas7         -6.3360  0.1148   0.6525  -7.6745   -5.0398 <0.001
Bs.gammas8         -6.2786  0.0980   0.6418  -7.5757   -5.0260 <0.001
Bs.gammas9         -6.2331  0.0972   0.6392  -7.5321   -4.9854 <0.001
```

```
Bs.gammas10        -6.2072  0.0978   0.6426  -7.5139   -4.9338 <0.001
Bs.gammas11        -6.1930  0.0972   0.6435  -7.4904   -4.9074 <0.001
Bs.gammas12        -6.1933  0.1006   0.6577  -7.5542   -4.8367 <0.001
Bs.gammas13        -6.2070  0.1031   0.6699  -7.5953   -4.7674 <0.001
Bs.gammas14        -6.2401  0.1088   0.6952  -7.6701   -4.7598 <0.001
Bs.gammas15        -6.2758  0.1178   0.7335  -7.7973   -4.7051 <0.001
Bs.gammas16        -6.3086  0.1300   0.7881  -7.9092   -4.6258 <0.001
Bs.gammas17        -6.3507  0.1415   0.8461  -8.0745   -4.5813 <0.001
tauBs             523.7904 29.3190 272.1537 143.7075 1153.3762     NA

MCMC summary:
iterations: 30000
adapt: 3000
burn-in: 3000
thinning: 15
time: 4.4 min
```

As explained earlier, argument `timeVar` is a character string that specifies the name of the time variable in the mixed model (the scale of time (e.g., days, months, years) in both the mixed effects and Cox models must be the same). In addition, using the control argument `n.iter` we specified that after adaption and burn-in, the MCMC should run for 30000 iterations. The default call to `jointModelBayes()` includes in the linear predictor of the relative risk model the subject-specific linear predictor of the mixed model $\eta_i(t)$, which in this case represents the average subject-specific log serum bilirubin level. The output of the `summary()` method is rather self-explanatory and contains model summary statistics, namely LPML (the log pseudo marginal likelihood value), DIC (deviance information criterion), and pD (the effective number of parameters component of DIC), posterior means for all parameters, and standard errors (based on an estimate of the effective sample size using time series methodology–function `effectiveSize()` in package **coda**; Plummer *et al.* 2006), standard deviations, 95% credibility intervals and tail probabilities for all regression coefficients in the two submodels. The association parameter α is denoted in the output as `Assoct`. The tail probabilities, under the column with the heading `P`, are calculated as $2 \times \min\{\Pr(\theta > 0), \Pr(\theta < 0)\}$, with θ denoting here the corresponding regression coefficient from the longitudinal or the survival submodel. The results suggest that serum bilirubin is strongly associated with the risk for the composite event, with a doubling of serum bilirubin levels, resulting in a 2.7-fold[1] (95% CI: 2.3; 3) increase of the risk. From the output we observe that by default 15 knots (that correspond to 17 coefficients) are used for the approximation of the baseline hazard functions using the penalized splines approach. Given that the spline coefficients are suitably and automatically penalized we do not expect any over-fitting issues even though we use a relatively high number of knots. To further investigate this we show in Appendix A the results of a short sensitivity analysis in which we have re-fitted model `jointFit.pbc1` using 10 and 20 knots. The results suggested little sensitivity in the number and positioning of the knots. In Appendix B we show how the `plot()` method can be used to produce diagnostic plots for investigating the convergence of the MCMC. By default the function produces the trace plots, but by altering the `which` argument it can also produce the plots of the auto-correlation function, kernel

[1] A difference of 0.693 in the log-scale for serum bilirubin corresponds to a ratio of 2 in the original scale, and hence $\exp(0.693 \times$ `Assoct`$)$ gives the corresponding hazard ratio for a doubling of serum bilirubin.

density estimation for each parameter, and the plot of the conditional predictive ordinate probabilities.

Left-truncation and exogenous time-varying covariates

When it is of interest to fit joint models with left-truncated survival times and/or it is also of interest to control for a number of exogenous time-varying covariates in the survival submodel, this can be done by first fitting a time-varying Cox model (using the counting process formulation in function `Surv()`) and supplying this as the second argument of `jointModelBayes()`. An example can be found in Appendix C.

4.2. Extended joint models

The previous section showed how the basic joint model for a continuous normally distributed longitudinal outcome and a time-to-event can be fitted in **JMbayes**. In this section we will illustrate how joint models for other types of longitudinal responses may be fitted using function `jointModelBayes()` by suitably specifying argument `densLong`. In particular, this argument accepts a function that calculates the probability density function (and its natural logarithm) of the longitudinal outcome, with arguments `y` denoting the vector of longitudinal responses y, `eta.y` the subject-specific linear predictor $\eta_i(t)$, `scale` a potential scale parameter (e.g., the standard deviation of the error terms), `log` a logical denoting whether the logarithm of the density is computed, and `data` a data frame that contains variables that are potentially required in the definition of `densLong`. To better illustrate the use of this function, we present here three examples of joint models with more elaborate longitudinal outcomes. We start with an extension of model `jointFit.pbc1` that allows for a more heavier-tailed error distribution, that is,

```
R> dLongST <- function (y, eta.y, scale, log = FALSE, data) {
+    dgt(x = y, mu = eta.y, sigma = scale, df = 4, log = log)
+  }
```

Function `dgt()` of package **JMbayes** calculates the probability density function of the generalized Student's-t distribution (i.e., a Student's-t with mean parameter `mu` and scale parameter `sigma`). Supplying this function in the `densLong` fits the corresponding joint model (time to fit: 4.3 min):

```
R> jointFit.pbc2 <- jointModelBayes(lmeFit.pbc1, coxFit.pbc1,
+    timeVar = "year", densLong = dLongST)
R> summary(jointFit.pbc2)

. . .

Variance Components:
              StdDev    Corr
(Intercept)   1.0087  (Intr)  n(,2)1
ns(year, 2)1  2.2300  0.3497
ns(year, 2)2  1.8899  0.2960  0.5676
Residual      0.2087
```

```
Coefficients:
Longitudinal Process
              Value Std.Err Std.Dev   2.5%  97.5%      P
(Intercept)  0.5150  0.0013  0.0584 0.4037 0.6322 <0.001
ns(year, 2)1 2.2707  0.0076  0.1482 1.9886 2.5642 <0.001
ns(year, 2)2 2.0616  0.0102  0.1452 1.7770 2.3534 <0.001

Event Process
                     Value Std.Err  Std.Dev     2.5%     97.5%      P
drugD-penicil      -0.8616  0.1444   0.7911  -2.3682    0.6809  0.315
age                 0.0349  0.0017   0.0109   0.0135    0.0573 <0.001
drugD-penicil:age   0.0162  0.0027   0.0148  -0.0131    0.0430  0.305
Assoct              1.3479  0.0060   0.0896   1.1786    1.5178 <0.001
tauBs             509.3855 32.4885 280.8333 136.9825 1218.6747     NA

. . .
```

We observe some slight changes in the regression coefficients of both submodels, where a doubling of serum bilirubin levels is now associated with a 2.5-fold (95% CI: 2.3; 2.9) increase of the risk for the composite event.

Following we illustrate the use of `densLong` for fitting a joint model with a dichotomous (binary) longitudinal outcome. Since in the PBC data there was no binary biomarker recorded during follow-up, we artificially create one by dichotomizing serum bilirubin at the threshold value of 1.8 mg/dL. To fit the corresponding joint model we need first to fit a mixed effects logistic regression for the longitudinal binary outcome using function `glmmPQL()` from package **MASS**. The syntax is:

```
R> pbc2$serBilirD <- as.numeric(pbc2$serBilir > 1.8)
R> lmeFit.pbc2 <- glmmPQL(serBilirD ~ year, random = ~ year | id,
+    family = binomial, data = pbc2)
```

As for continuous longitudinal outcomes, this mixed effects model object is merely used to extract the required data (response vector, design matrices for fixed and random effects), and starting values for the parameters and random effects. The definition of `densLong` and the call to `jointModelBayes()` take the form (time to fit: 3 min):

```
R> dLongBin <- function (y, eta.y, scale, log = FALSE, data) {
+    dbinom(x = y, size = 1L, prob = plogis(eta.y), log = log)
+  }
R> jointFit.pbc3 <- jointModelBayes(lmeFit.pbc2, coxFit.pbc1,
+    timeVar = "year", densLong = dLongBin)
R> summary(jointFit.pbc3)

. . .
```

```
Variance Components:
             StdDev    Corr
(Intercept)  6.7560  (Intr)
year         1.1778  0.4718


Coefficients:
Longitudinal Process
              Value Std.Err Std.Dev    2.5%   97.5%      P
(Intercept) -1.7330  0.0221  0.4459 -2.6406 -0.8538 <0.001
year         0.8961  0.0030  0.0843  0.7412  1.0779 <0.001

Event Process
                     Value Std.Err  Std.Dev    2.5%     97.5%      P
drugD-penicil      -0.5764  0.2866   1.0522 -2.8183    1.3712  0.586
age                 0.0357  0.0038   0.0148  0.0061    0.0613  0.023
drugD-penicil:age   0.0075  0.0051   0.0196 -0.0273    0.0485  0.715
Assoct              0.2234  0.0052   0.0342  0.1661    0.3026 <0.001

. . .
```

As we have already seen, the default parameterization posits that the subject-specific linear predictor $\eta_i(t)$ from the mixed model is included as a time-varying covariate in the relative risk model. This means that, in this case, the estimate of the association parameter $\alpha = 0.2$ denotes the log hazard ratio for a unit increase in the log odds of having serum bilirubin above 1.8 mg/dL. The user has great flexibility in defining her own density function for the longitudinal outcome; for example, we can easily fit a mixed effects probit regression instead of using the logit link by defining `densLong` as

```
R> dLongBin <- function (y, eta.y, scale, log = FALSE, data) {
+    dbinom(x = y, size = 1L, prob = pnorm(eta.y), log = log)
+  }
```

As a final example, we illustrate how `densLong` can be utilized to fit joint models with censored longitudinal data (detection limit problem) by making use of extra variables in the data frame containing the longitudinal information. Similarly to the previous example, the biomarkers collected in PBC study were not subject to detection limits, and therefore we again artificially create a censored version of serum bilirubin with values below the threshold value of 0.8 mg/dL set equal to the detection limit of 0.8 mg/dL. The code creating the censored longitudinal response vector is:

```
R> pbc2$CensInd <- as.numeric(pbc2$serBilir <= 0.8)
R> pbc2$serBilir2 <- pbc2$serBilir
R> pbc2$serBilir2[pbc2$serBilir2 <= 0.8] <- 0.8
```

In addition to the censored version of serum bilirubin we have also included in the data frame `pbc2` the left censoring indicator `CensInd`. We again assume a normal error distribution for

the logarithm of serum bilirubin but in the definition of the corresponding density function we need to account for left censoring, that is for observations above the detection limit we use the density function whereas for observations below this limit we use the cumulative distribution function. The definition of the left-censored density becomes:

```
R> censdLong <- function (y, eta.y, scale, log = FALSE, data) {
+    log.f <- dnorm(x = y, mean = eta.y, sd = scale, log = TRUE)
+    log.F <- pnorm(q = y, mean = eta.y, sd = scale, log.p = TRUE)
+    ind <- data$CensInd
+    log.dens <- (1 - ind) * log.f + ind * log.F
+    if (log) log.dens else exp(log.dens)
+  }
```

Note that the left censoring indicator is extracted from the `data` argument of `censdLong()`. Again in order to fit the joint model, we first need to fit the linear mixed model for the censored response variable `serBilir2` and following supply this object and the `censdLong()` to `jointModelBayes()`, i.e., (time to fit: 4.4 min)

```
R> lmeFit.pbc3 <- lme(log(serBilir2) ~ ns(year, 2), data = pbc2,
+    random = ~ ns(year, 2) | id)
R> jointFit.pbc4 <- jointModelBayes(lmeFit.pbc3, coxFit.pbc1,
+    timeVar = "year", densLong = censdLong)
R> summary(jointFit.pbc4)

. . .

Variance Components:
              StdDev    Corr
(Intercept)   1.2148  (Intr)  n(,2)1
ns(year, 2)1  2.5787  0.1980
ns(year, 2)2  2.3263  0.1579  0.4756
Residual      0.3284

Coefficients:
Longitudinal Process
              Value Std.Err Std.Dev   2.5%  97.5%      P
(Intercept)  0.3929  0.0019  0.0725 0.2508 0.5314 <0.001
ns(year, 2)1 2.2681  0.0071  0.1718 1.9363 2.6059 <0.001
ns(year, 2)2 2.1280  0.0106  0.1782 1.7763 2.4880 <0.001

Event Process
                     Value Std.Err  Std.Dev    2.5%     97.5%      P
drugD-penicil      -0.6215  0.1229   0.6901 -1.8997    0.7849  0.369
age                 0.0424  0.0018   0.0104  0.0208    0.0624 <0.001
drugD-penicil:age   0.0106  0.0021   0.0127 -0.0156    0.0344  0.407
Assoct              1.3908  0.0082   0.1043  1.1939    1.5983 <0.001

. . .
```

The `lmeFit.pbc3` mixed model fitted by `lme()` ignores the left censoring issue, but we should stress again that this model is just used to extract the relevant information in order to fit the joint model. From the results we observe that the estimate of the association parameter α is relatively close to the estimate obtained in model `jointFit.pbc1` that was based on the original (uncensored) version of serum bilirubin. As mentioned above, function `censdLong()` accounts for left censoring, but it could be easily modified to cover right or interval censored longitudinal responses.

4.3. Association structures

The joint models we fitted in Sections 4.1 and 4.2 assumed that the hazard for an event at any time t is associated with the current underlying value of the biomarker at the same time point, denoted as $\eta_i(t)$, and the strength of this association is measured by parameter α. This formulation is the standard formulation used for time-varying covariates. We should stress however, that this is not the only option that we have. In general, there could be other characteristics of the subjects' longitudinal profiles that are more strongly predictive for the risk of an event; for example, the rate of increase/decrease of the biomarker's levels or a suitable summary of the whole longitudinal trajectory, among others. In this section we illustrate how such association structures could be postulated and fitted with `jointModelBayes()`.

We start with the parameterization proposed by Ye, Lin, and Taylor (2008), Brown (2009) and Rizopoulos (2012) that posits that the risk depends on both the current true value of the trajectory and its slope at time t. More specifically, the relative risk survival submodel takes the form,

$$h_i(t) = h_0(t) \exp\big\{\boldsymbol{\gamma}^\top \boldsymbol{w}_i(t) + \alpha_1 \eta_i(t) + \alpha_2 \eta_i'(t)\big\}, \tag{8}$$

where $\eta_i'(t) = d\{\boldsymbol{x}_i^\top(t)\boldsymbol{\beta} + \boldsymbol{z}_i^\top(t)\boldsymbol{b}_i\}/dt$. The interpretation of parameter α_1 remains the same as in the standard parameterization but also corrected for $\eta_i'(t)$. Parameter α_2 measures the association between the slope of the true longitudinal trajectory at time t and the risk for an event at the same time point, provided that $\eta_i(t)$ remains constant. To fit the joint model with the extra slope term in the relative risk submodel we need to specify the `param` and `extraForm` arguments of `jointModelBayes()`. The first one is a character string with options `"td-value"` (default) that denotes that only the current value term $\eta_i(t)$ is included, `"td-extra"` which means that only the extra, user-defined, term is included, and `"td-both"` which means that both $\eta_i(t)$ and the user-defined terms are included. The exact definition of the extra term is provided via the argument `extraForm` which is a list with four components, namely

* `"fixed"` an R formula specifying the fixed-effects part of the extra term,
* `"random"` an R formula specifying the random-effects part of the extra term,
* `"indFixed"` an integer vector denoting which of the fixed effects of the original mixed model are encountered in the definition of the extra term, and,
* `"indRandom"` an integer vector denoting which of the random effects of the original mixed model are encountered in the definition of the extra term.

For example, to include the slope term $\eta_i'(t)$ under the linear mixed model `lmeFit.pbc1`, this list takes the form:

```
R> dForm <- list(fixed = ~ 0 + dns(year, 2), random = ~ 0 + dns(year, 2),
+    indFixed = 2:3, indRandom = 2:3)
```

Function `dns()` computes numerically (with a central difference approximation) the derivative of a natural cubic spline as calculated by function `ns()` (there is also a similar function `dbs()` that computes numerically the derivative of a cubic spline as calculated by function `bs()`). The corresponding joint model is fitted with the code (time to fit: 5.3 min):

```
R> jointFit.pbc12 <- update(jointFit.pbc1, param = "td-both",
+    extraForm = dForm)
R> summary(jointFit.pbc12)

. . .

Event Process
                     Value Std.Err  Std.Dev    2.5%     97.5%      P
drugD-penicil      -0.4086  0.1589   0.9553 -2.1773    1.5366  0.633
age                 0.0428  0.0017   0.0119  0.0203    0.0673 <0.001
drugD-penicil:age   0.0061  0.0032   0.0186 -0.0322    0.0403  0.693
Assoct              1.3530  0.0064   0.1010  1.1594    1.5561 <0.001
AssoctE             2.0905  0.0546   0.5425  1.0127    3.1455 <0.001
tauBs             378.1742 46.1695 268.8869 40.2447 1036.5408     NA

. . .
```

We observe that both the current level and the current slope of the longitudinal profile are strongly associated with the risk for the composite event. For patients with the same treatment and age at baseline, and who have the same underlying level of serum bilirubin at time t, if serum bilirubin has increased by 50% within a year then the corresponding hazard ratio is 2.3[2] (95% CI: 1.5; 3.6).

A common characteristic of the two parameterizations we have seen so far is that the risk for an event at any time t is assumed to be associated with features of the longitudinal trajectory at the same time point (i.e., current value $\eta_i(t)$ and current slope $\eta_i'(t)$). However, this assumption may not always be appropriate, and we may benefit from allowing the risk to depend on a more elaborate function of the history of the time-varying covariate (Sylvestre and Abrahamowicz 2009). In the context of joint models, one option to account for the history of the longitudinal outcome is to include in the linear predictor of the relative risk submodel the integral of the longitudinal trajectory from baseline up to time t (Brown 2009; Rizopoulos 2012). More specifically, the survival submodel takes the form

$$h_i(t) = h_0(t) \exp\left\{\boldsymbol{\gamma}^\top \boldsymbol{w}_i(t) + \alpha \int_0^t \eta_i(s) ds\right\},$$

where for any particular time point t, α measures the strength of the association between the risk for an event at time point t and the area under the longitudinal trajectory up to the

[2] A difference of 0.4054 in the log-scale for serum bilirubin corresponds to a ratio of 1.5, and hence $\exp(0.4054 \times$ `Assoct`$)$ gives the corresponding hazard ratio for a 50% increase of serum bilirubin.

same time t, with the area under the longitudinal trajectory taken as a summary of the whole marker history $\mathcal{H}_i(t) = \{\eta_i(s), 0 \leq s < t\}$. To fit a joint model with this term in the linear predictor of the survival submodel, we need first again to appropriately define the formulas that calculate its fixed-effects and random-effects parts. Similarly to including the slope term, the list with these formulas takes the form

```
R> iForm <- list(fixed = ~ 0 + year + ins(year, 2),
+    random = ~ 0 + year + ins(year, 2),
+    indFixed = 1:3, indRandom = 1:3)
```

where function `ins()` calculates numerically (using the Gauss-Kronrod rule) the integral of function `ns()`. The corresponding joint model is fitted by supplying this list in the `extraForm` argument and by also setting in argument `param` that we only want to include the extra term in the linear predictor of the survival submodel (time to fit: 4.5 min):

```
R> jointFit.pbc13 <- update(jointFit.pbc1, param = "td-extra",
+    extraForm = iForm)
R> summary(jointFit.pbc13)

. . .

Event Process
                      Value Std.Err  Std.Dev    2.5%    97.5%      P
drugD-penicil       -0.8781  0.0938   0.6724 -2.0859   0.4829  0.207
age                  0.0365  0.0016   0.0106  0.0138   0.0564 <0.001
drugD-penicil:age    0.0117  0.0017   0.0125 -0.0124   0.0344  0.350
AssoctE              0.2211  0.0025   0.0199  0.1833   0.2614 <0.001
tauBs              297.3440 30.7747 195.3019 63.8090 792.4058     NA

. . .
```

To explicitly denote that in the relative risk submodel we only want to include the user-defined integral term, we have set argument `param` to `"td-extra"`. Similarly to the previous results we observe that the area under the longitudinal profile of log serum bilirubin is strongly associated with the risk for an event, with a unit increase corresponding to a 1.2-fold (95% CI: 1.2; 1.3) increase of the risk.

A feature of the aforementioned definition of the cumulative effect functional form is that it places the same weight to early and later changes in the longitudinal profile. This assumption may not be reasonable especially when t is considerably later than zero. For example, it would often be reasonable to assume that the hazard at time t would be more strongly related to the values of the longitudinal outcomes that are close to t than zero. An extension of the cumulative effects parameterization that allows for such differential weights is

$$h_i(t) = h_0(t) \exp\left\{\boldsymbol{\gamma}^\top \boldsymbol{w}_i(t) + \alpha \int_0^t \varpi(t-s)\eta_i(s)ds\right\},$$

where $\varpi(t-s)$ denotes an appropriately chosen weight function that places different weights at different time points. A possible family of functions with this property are probability

density functions of known parametric distributions, such as the normal, the Student's-t and the logistic. The scale parameter in these densities and the degrees of freedom parameter in the Student's-t density can be used to tune the relative weights of more recent marker values compared to older ones. Estimation of the parameters of the weight function can shed light on the important question of how much of the biomarker's history do we really need to predict future events. Function `jointModelBayes()` can fit joint models with the weighted integrand parameterization with user specified weight functions that could have up to three parameters. As an example, the following piece of code defines a weight function based on the probability density function of the normal distribution and has as parameter the standard deviation:

```
R> wf <- function (u, parms, t.max) {
+    num <- dnorm(x = u, sd = parms)
+    den <- pnorm(q = c(0, t.max), sd = parms)
+    num / (den[2L] - den[1L])
+  }
```

The first argument `u` denotes the time lag $t - s$. The denominator is used to normalize the weight function taking into account the length of the follow-up period, where `t.max` denotes the largest observed event time. The joint model is fitted using the code (time to fit: 1.2 hours):

```
R> jointFit.pbc13w <- update(jointFit.pbc1, estimateWeightFun = TRUE,
+    weightFun = wf,
+    priorShapes = list(shape1 = dunif),
+    priors = list(priorshape1 = c(0, 10)))
R> summary(jointFit.pbc13w)

. . .

Event Process
                      Value Std.Err  Std.Dev     2.5%     97.5%      P
drugD-penicil      -0.7146  0.1568   0.8959  -2.2237    1.2752  0.377
age                 0.0415  0.0022   0.0130   0.0174    0.0663 <0.001
drugD-penicil:age   0.0129  0.0032   0.0175  -0.0267    0.0428  0.407
Assoct              1.3970  0.0056   0.0965   1.2142    1.5933 <0.001
shape1              0.0904  0.0078   0.0373   0.0406    0.1719 <0.001
tauBs             478.9661 30.4305 266.3022 130.6122 1146.1424     NA

. . .
```

Arguments `priorShapes` and `priors` are used to specify the prior distribution and its parameters for the shape parameters of the weight function. As mentioned above, up to three shape parameters can be specified. The estimate of the standard deviation is 0.1, implying that only the bilirubin measurements within the last 0.3 years (three times the standard deviation) before t are associated with the risk of an event at t. This interval is very narrow and this is why the estimated association parameter from this joint model is practically equal to the estimate of the association parameter we obtained from `jointFit.pbc1`, which postulates

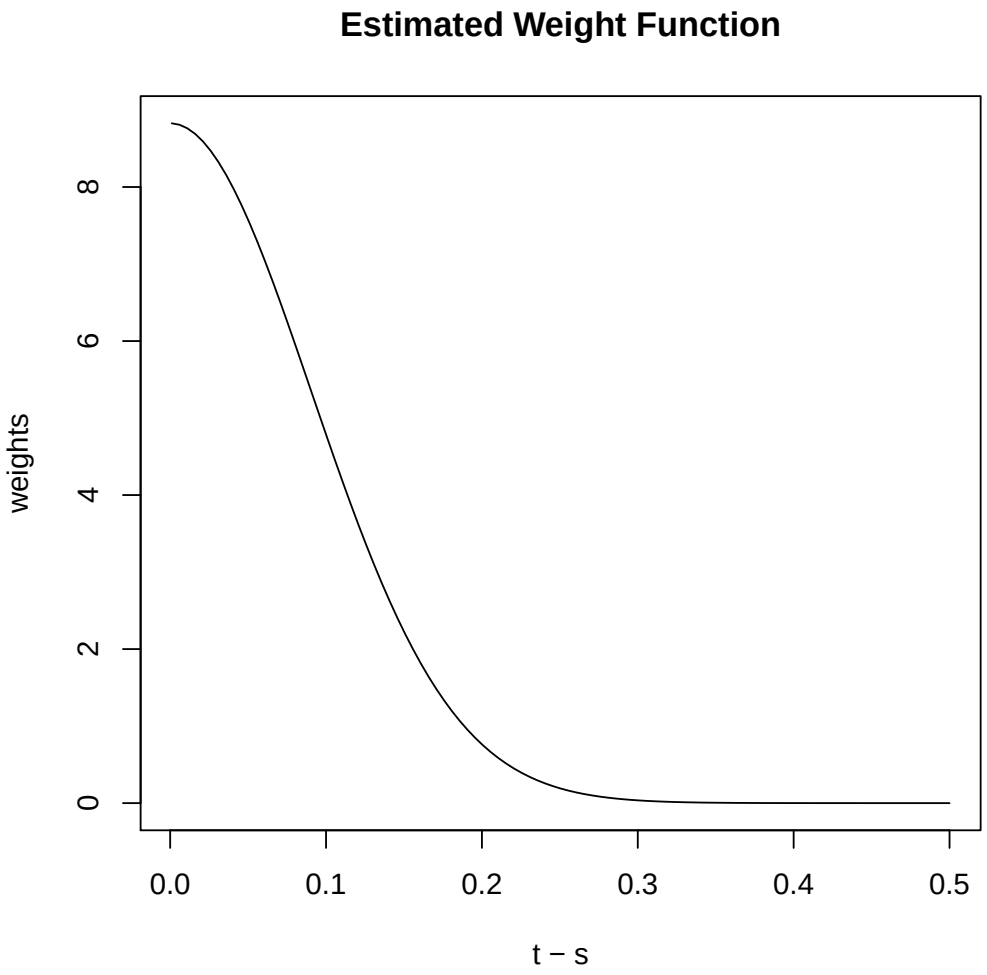

Figure 3: Estimated weight function for the weighted cumulative effects parameterization.

that the hazard at t is associated with the level of serum bilirubin at the same time point. A visual representation of the estimated weight functions is provided by the `plot()` method and is illustrated in Figure 3 (argument `max.t` defines the upper limit of the interval $[0, max.t]$ in which $\varpi(t-s)$ is plotted):

```
R> plot(jointFit.pbc13w, which = "weightFun", max.t = 0.5)
```

The final type of association structure we consider assumes that only the random effects are shared between the two processes, namely

$$h_i(t) = h_0(t) \exp(\boldsymbol{\gamma}^\top \boldsymbol{w}_i(t) + \boldsymbol{\alpha}^\top \boldsymbol{b}_i), \tag{9}$$

This type of parameterization is more meaningful when a simple random-intercepts and random-slopes structure is assumed for the longitudinal submodel, in which case the random effects express subject-specific deviations from the average intercept and average slope. Under

this setting this parameterization postulates that patients who have a lower/higher level for the longitudinal outcome at baseline (i.e., intercept) or who show a steeper increase/decrease in their longitudinal trajectories (i.e., slope) are more likely to experience the event. In that respect, this formulation shares similarities with the time-dependent slopes formulation (8). A joint model with a relative risk model of the form (9) can be fitted by setting argument `param` to `"shared-RE"` in the call to `jointModelBayes()`; for example, for the PBC dataset a joint model with this parameterization is fitted with the code (time to fit: 6 min):

```
R> jointFit.pbc14 <- update(jointFit.pbc1, param = "shared-RE",
+    n.iter = 50000)
R> summary(jointFit.pbc14)

. . .

Event Process
                        Value Std.Err  Std.Dev    2.5%    97.5%      P
drugD-penicil         -0.2928  0.1361   0.9196 -1.9593   1.6204  0.718
age                    0.0439  0.0015   0.0120  0.0203   0.0668 <0.001
drugD-penicil:age      0.0056  0.0024   0.0172 -0.0291   0.0373  0.716
Assoct:(Intercept)     1.3655  0.0178   0.1512  1.0922   1.6833 <0.001
Assoct:ns(year, 2)1    0.5381  0.0054   0.0629  0.4112   0.6587 <0.001
Assoct:ns(year, 2)2    0.1545  0.0121   0.1010 -0.0220   0.3813  0.083
tauBs                167.5994 25.2895 120.5442 39.2550 501.8257     NA

. . .
```

The results suggest that both the baseline levels of the underlying log serum bilirubin (i.e., parameter `Assoct:(Intercept)`) as well as the longitudinal evolution of the marker (i.e., parameters `Assoct:ns(year, 2)1` and `Assoct:ns(year, 2)2`) are strongly related to the hazard of the composite event.

The aforementioned association structures represent just a few options from the plethora of available choices that we have to link the two outcomes under the joint modeling framework. This raises the important question of how to chose the most appropriate structure for a particular dataset we have at hand. In general, this is a difficult question because more than one association structures could be potentially used at the same time (an option that is not currently offered by **JMbayes**). When one is interested in understanding the interrelationships between the longitudinal and survival outcomes, then the choice of the most appropriate structure can be approached as a model selection problem. Different joint models could be fitted assuming different structures and the optimal model could be selected using information criteria, such as the DIC. When interest is in predictions, then an alternative option that accounts for model uncertainty is to utilize model averaging techniques (see Section 5.2).

4.4. Transformation functions

The previous section illustrated several options for the definition of function $f(\cdot)$ in (2) for studying which features of the longitudinal process are associated with the event of interest. Yet another set of options for function $f(\cdot)$ would be to consider adding interaction or nonlinear

terms for the components of the longitudinal outcome that are included in the linear predictor of the relative risk model. Such options are provided in `jointModelBayes()` by suitably specifying argument `transFun`. This should be a function (or a list of two functions) with arguments `x` denoting the term from the longitudinal model, and `data` a data frame that contains other variables that potentially should be included in the calculation. These functions need to be vectorized with respect to `x`. When a single function is provided, then this function is applied to the current value term $\eta_i(t)$ and potentially also to the extra term provided by the user if `param` was set to `"td-both"`. If a list is provided, then this should be a named list with components `"value"` and `"extra"` providing separate functions for the current value and the user-defined terms, respectively. We illustrate how these transformation functions can be used in practice by extending model `jointFit.pbc12`, which included the current value term $\eta_i(t)$ and the current slope term $\eta_i'(t)$, by including the quadratic effect of $\eta_i(t)$ and the interaction of $\eta_i'(t)$ with the randomized treatment, i.e.,

$$\begin{aligned} h_i(t) \quad = \quad & h_0(t) \exp\big[\gamma_1 \texttt{D-penicil}_i + \gamma_2 \texttt{Age}_i + \gamma_3 (\texttt{D-penicil}_i \times \texttt{Age}_i) \\ & + \alpha_1 \eta_i(t) + \alpha_2 \{\eta_i(t)\}^2 + \alpha_3 \eta_i'(t) + \alpha_4 \{\eta_i'(t) \times \texttt{D-penicil}_i\}\big]. \end{aligned}$$

To fit the corresponding joint model we first define the two transformation functions as:

```
R> tf1 <- function (x, data) {
+    cbind(x, "^2" = x*x)
+  }

R> tf2 <- function (x, data) {
+    cbind(x, "D-penicil" = x * (data$drug == 'D-penicil'))
+  }
```

Following we update the call to `jointFit.pbc12` and supply the list of the two functions in argument `transFun` (time to fit: 5.8 min),

```
R> jointFit.pbc15 <- update(jointFit.pbc12,
+    transFun = list("value" = tf1, "extra" = tf2))
R> summary(jointFit.pbc15)

. . .

Event Process
                     Value Std.Err  Std.Dev    2.5%    97.5%      P
drugD-penicil      -0.5655  0.0583   0.7549 -2.0492   0.9193  0.466
age                 0.0418  0.0009   0.0082  0.0255   0.0582 <0.001
drugD-penicil:age   0.0078  0.0012   0.0149 -0.0220   0.0364  0.575
Assoct              0.8492  0.0225   0.3485  0.1966   1.5435  0.009
Assoct:^2           0.1506  0.0064   0.1029 -0.0456   0.3497  0.146
AssoctE             2.3196  0.0540   0.7075  0.9285   3.7049  0.001
AssoctE:D-penicil   0.3233  0.0611   0.9117 -1.4358   2.1332  0.724
tauBs             343.3803 16.5619 196.4634 75.3488 812.4817     NA

. . .
```

There is no evidence that the effect of $\eta_i(t)$ could be nonlinear nor that the association between $\eta_i'(t)$ and the hazard is different between the two treatment groups.

4.5. Supporting functions

Several supporting functions are available in the package that extract or calculate useful statistics based on the fitted joint model. In particular, function `jointModelBayes()` return objects of class `"JMbayes"`, for which there are S3 methods defined for several of the standard generic functions in R. The most important are enlisted below:

Functions `coef()` and `fixef()` extract the estimated coefficients (posterior means) for the two submodels from a fitted joint model. For the survival process both provide the same output, but for the longitudinal model, the former returns the subject-specific regression coefficients (i.e., the fixed effects plus their corresponding random effects estimates), whereas the latter only returns the estimated fixed effects.

Function `ranef()` extracts the posterior means for the random effects for each subject. The function also extracts estimates for the dispersion matrix of the posterior of the random effects using argument `postVar`.

Function `vcov()` extracts the estimated variance-covariance matrix of the parameters from the MCMC sample.

Functions `fitted()` and `residuals()` compute several kinds of fitted values and residuals, respectively, for the two outcomes. For the longitudinal outcome the `fitted()` method computes the marginal $\boldsymbol{X}\hat{\boldsymbol{\beta}}$ and subject-specific $\boldsymbol{X}\hat{\boldsymbol{\beta}} + \boldsymbol{Z}\hat{\boldsymbol{b}}_i$ fitted values, where $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{b}}_i$ denote the posterior means of the fixed and random effects, whereas for the survival outcome it computes the cumulative hazard function for every subject at every time point a longitudinal measurement was collected. Analogously, method `residuals()` calculates the marginal and subject-specific residuals for the longitudinal outcome, and the martingale residuals for the survival outcome.

Function `anova()` can be used to compare joint models on the basis of the DIC, pD and LPML values.

Function `plot()` produces diagnostic plots for the MCMC sample, including trace plots, auto-correlation plots and kernel density estimation plots.

Function `predict()` produces subject-specific and marginal predictions for the longitudinal outcome, while function `survfitJM()` produces subject-specific predictions for the event time outcome, along with associated confidence or prediction confidence intervals. The use of these functions is explained in detail and illustrated in Section 5.

Function `logLik()` calculates the log-likelihood value for the posterior means of the parameters and the random effects, and can be also used to obtain the marginal log-likelihood (integrated over the parameters and random effects) using the Laplace approximation.

Function `xtable()` returns the LaTeX code to produce the table of posterior means, posterior standard deviations, and 95% credibility intervals from a fitted joint model. This is a method for the generic function `xtable()` from package **xtable** (Dahl 2014).

5. Dynamic predictions

5.1. Definitions and estimation

In recent years there has been increasing interest in medical research towards personalized medicine. In particular, physicians would like to tailor decision making on the characteristics of individuals patients with aim to optimize medical care. In the same sense, patients who are informed about their individual health risk often decide to adjust their lifestyles to mitigate it. In this context it is often of interest to utilize results from tests performed on patients on a regular basis to derive medically-relevant summary measures, such as survival probabilities. Joint models constitute a valuable tool that can be used to derive such probabilities and also provide predictions for future biomarker levels. More specifically, under the Bayesian specification of the joint model, presented in Section 2, we can derive subject-specific predictions for either the survival or longitudinal outcomes (Yu, Taylor, and Sandler 2008; Rizopoulos 2011, 2012; Taylor, Park, Ankerst, Proust-Lima, Williams, Kestin, Bae, Pickles, and Sandler 2013). To put it more formally, based on a joint model fitted in a sample $\mathcal{D}_n = \{T_i, \delta_i, \boldsymbol{y}_i; i = 1, \ldots, n\}$ from the target population, we are interested in deriving predictions for a new subject j from the same population that has provided a set of longitudinal measurements $\mathcal{Y}_j(t) = \{y_j(t_{jl}); 0 \leq t_{jl} \leq t, l = 1, \ldots, n_j\}$, and has a vector of baseline covariates $\boldsymbol{w}_j$. The fact that biomarker measurements have been recorded up to t, implies that subject j was event-free up to this time point, and therefore it is more relevant to focus on conditional subject-specific predictions, given survival up to t. In particular, for any time $u > t$ we are interested in the probability that subject j will survive at least up to u, i.e.,

$$\pi_j(u \mid t) = \Pr(T_j^* \geq u \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{w}_j, \mathcal{D}_n).$$

Similarly, for the longitudinal outcome we are interested in the predicted longitudinal response at u, i.e.,

$$\omega_j(u \mid t) = E\big\{y_j(u) \mid T_j^* > t, \mathcal{Y}_j(t), \mathcal{D}_n\big\}.$$

The dynamic nature of both $\pi_j(u \mid t)$ and $\omega_j(u \mid t)$ is evident from the fact that when new information is recorded for subject j at time $t' > t$, we can update these predictions to obtain $\pi_j(u \mid t')$ and $\omega_j(u \mid t')$, and therefore proceed in a time-dynamic manner.

Under the joint modeling framework of Section 2, estimation of either $\pi_j(u \mid t)$ or $\omega_j(u \mid t)$ is based on the corresponding posterior predictive distributions, namely

$$\pi_j(u \mid t) = \int \Pr(T_j^* \geq u \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D}_n)\, d\boldsymbol{\theta},$$

for the survival outcome, and analogously

$$\omega_j(u \mid t) = \int E\big\{y_j(u) \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta}\big\}\, p(\boldsymbol{\theta} \mid \mathcal{D}_n)\, d\boldsymbol{\theta},$$

for the longitudinal one. The calculation of the first part of each integrand takes full advantage of the conditional independence assumptions (4) and (5). In particular, we observe that the first term of the integrand of $\pi_j(u \mid t)$ can be rewritten by noting that:

$$\begin{aligned}
\Pr(T_j^* \geq u \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta}) &= \int \Pr(T_j^* \geq u \mid T_j^* > t, \boldsymbol{b}_j, \boldsymbol{\theta})\, p(\boldsymbol{b}_j \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta})\, d\boldsymbol{b}_j \\
&= \int \frac{S_j\big\{u \mid \mathcal{H}_j(u, \boldsymbol{b}_j), \boldsymbol{\theta}\big\}}{S_j\big\{t \mid \mathcal{H}_j(t, \boldsymbol{b}_j), \boldsymbol{\theta}\big\}}\, p(\boldsymbol{b}_j \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta})\, d\boldsymbol{b}_j,
\end{aligned}$$

whereas for $\omega_j(u \mid t)$ we similarly have:

$$\begin{aligned} E\{y_j(u) \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta}\} &= \int E\{y_j(u) \mid \boldsymbol{b}_j, \boldsymbol{\theta}\}\, p(\boldsymbol{b}_j \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta})\, d\boldsymbol{b}_j \\ &= \boldsymbol{x}_j^\top(u)\boldsymbol{\beta} + \boldsymbol{z}_j^\top(u)\bar{\boldsymbol{b}}_j^{(t)}, \end{aligned}$$

with

$$\bar{\boldsymbol{b}}_j^{(t)} = \int \boldsymbol{b}_j\, p(\boldsymbol{b}_j \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta})\, d\boldsymbol{b}_j.$$

Combining these equations with the MCMC sample from the posterior distribution of the parameters for the original data $\mathcal{D}_n$, we can devise a simple simulation scheme to obtain Monte Carlo estimates of $\pi_j(u \mid t)$ and $\omega_j(u \mid t)$. More details can be found in Yu *et al.* (2008), Rizopoulos (2011, 2012), and Taylor *et al.* (2013).

In package **JMbayes** these subject-specific predictions for the survival and longitudinal outcomes can be calculated using functions `survfitJM()` and `predict()`, respectively. As an illustration we show how these functions can be utilized to derive predictions for Patient 2 from the PBC dataset using joint model `jointFit.pbc15`. We first extract the data of this patient in a separate data frame

```
R> ND <- pbc2[pbc2$id == 2, ]
```

Function `survfitJM()` has two required arguments, the joint model object based on which predictions will be calculated, and the data frame with the available longitudinal data and baseline information. For Patient 2 estimates of $\pi_j(u \mid t)$ are calculated with the code:

```
R> sfit.pbc15 <- survfitJM(jointFit.pbc15, newdata = ND)
R> sfit.pbc15

Prediction of Conditional Probabilities for Event
        based on 200 Monte Carlo samples

$`2`
    times   Mean Median  Lower  Upper
1  8.8325 1.0000 1.0000 1.0000 1.0000
1  8.9232 0.9888 0.9911 0.9669 0.9977
2  9.2496 0.9479 0.9595 0.8475 0.9896
3  9.5759 0.9063 0.9274 0.7295 0.9824
4  9.9022 0.8642 0.8964 0.6142 0.9757
5 10.2286 0.8223 0.8649 0.5075 0.9695
6 10.5549 0.7808 0.8350 0.4017 0.9637
7 10.8812 0.7404 0.8022 0.2929 0.9585
8 11.2076 0.7016 0.7739 0.2166 0.9539
```

By default `survfitJM()` assumes that the patient was event-free up to the time point of the last longitudinal measurement (if the patient was event-free up to a later time point, this can be specified using argument `last.time`). In addition, by default `survfitJM()` estimates of $\pi_j(u \mid t)$ using 200 Monte Carlo samples (controlled with argument `M`) for the time

points $\{u : u > t_\ell, \ell = 1, \ldots, 35\}$, with $\{t_\ell, \ell = 1, \ldots, 35\}$ calculated as `seq(min(Time), quantile(Time, 0.9) + 0.01, length.out = 35)` with `Time` denoting the observed event times variable. The user may override these default time points and specify her own using argument `survTimes`. In the output of `survfitJM()` we obtain as estimates of $\pi_j(u \mid t)$ the mean and median over the Monte Carlo samples along with the 95% pointwise confidence intervals. If only point estimates are of interest, `survfitJM()` provides the option (by setting argument `simulate` to `FALSE`) to use a first order estimator of $\pi_j(u \mid t)$ calculated as:

$$\tilde{\pi}_j(u \mid t) = \frac{S_j\{u \mid \mathcal{H}_j(u, \hat{\boldsymbol{b}}_j), \hat{\boldsymbol{\theta}}\}}{S_j\{t \mid \mathcal{H}_j(t, \hat{\boldsymbol{b}}_j), \hat{\boldsymbol{\theta}}\}},$$

where $\hat{\boldsymbol{\theta}}$ denotes here the posterior means of the model parameters, and $\hat{\boldsymbol{b}}_j$ the mode of the posterior density $p(\boldsymbol{b}_j \mid T_j^* > t, \mathcal{Y}_j(t), \hat{\boldsymbol{\theta}})$ with respect to $\boldsymbol{b}_j$. The corresponding `plot()` method for objects created by `survfitJM()` produces the figure of estimated conditional survival probabilities; for Patient 2 this is depicted in Figure 4. By setting logical argument `include.y` to `TRUE`, the fitted longitudinal profile is also included in the plot, i.e.,

```
R> plot(sfit.pbc15, estimator = "mean", include.y = TRUE,
+    conf.int = TRUE, fill.area = TRUE, col.area = "lightgrey")
```

Argument `estimator` specifies whether the `"mean"` or the `"median"` over the 200 Monte Carlo samples should be used as an estimate of $\pi_j(u \mid t)$, and in addition arguments `conf.int`, `fill.area` and `col.area` control the appearance of the 95% confidence intervals. In a similar manner, predictions for the longitudinal outcome are calculated by the `predict()` function. For example, predictions of future log serum bilirubin levels for Patient 2 are produced with the code:

```
R> Ps.pbc15 <- predict(jointFit.pbc15, newdata = ND, type = "Subject",
+    interval = "confidence", return = TRUE)
```

Argument `type` specifies if subject-specific or marginal predictions are to be computed[3], argument `interval` specifies the type of interval to compute (i.e., confidence or prediction), and by setting argument `return` to `TRUE`, `predict()` returns the data frame supplied in the required argument `newdata` having as extra columns the corresponding predictions and the limits of the confidence/prediction interval. This option facilitates plotting these predictions by a simple call to `xyplot()`, i.e.,

```
R> last.time <- with(Ps.pbc15, year[!is.na(low)][1])
R> xyplot(pred + low + upp ~ year, data = Ps.pbc15, type = "l",
+    lty = c(1,2,2), col = c(2,1,1),
+    abline = list(v = last.time, lty = 3),
+    xlab = "Time (years)", ylab = "Predicted log(serum bilirubin)")
```

The first line of the code extracts from the data frame `Ps.pbc15` the last time point at which Patient 2 was still alive, which is passed in the `abline` argument that produces Figure 5.

[3]By marginal predictions we refer to $\boldsymbol{x}_i^\top(t)\hat{\boldsymbol{\beta}}$, whereas by subject-specific to $\boldsymbol{x}_i^\top(t)\hat{\boldsymbol{\beta}} + \boldsymbol{z}_i^\top(t)\hat{\boldsymbol{b}}_i$.

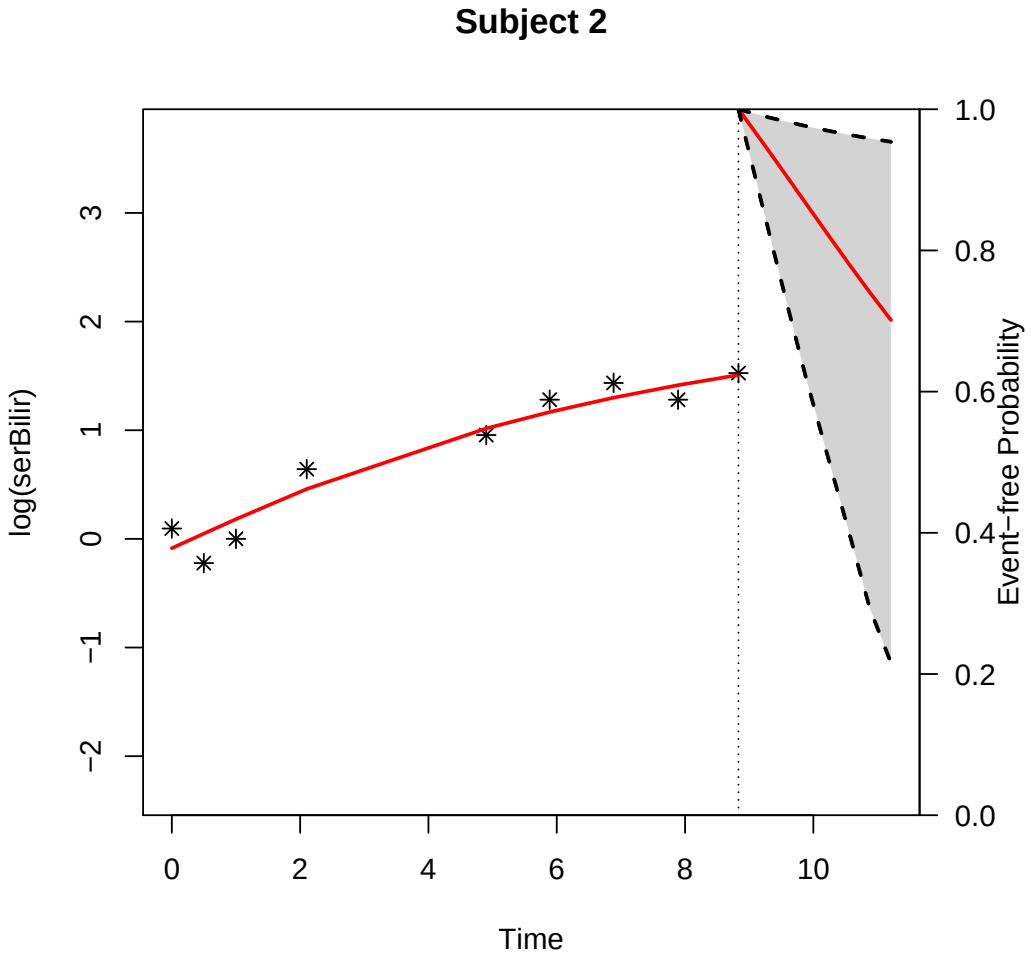

Figure 4: Estimated conditional survival probabilities for Patient 2 from the PBC dataset.

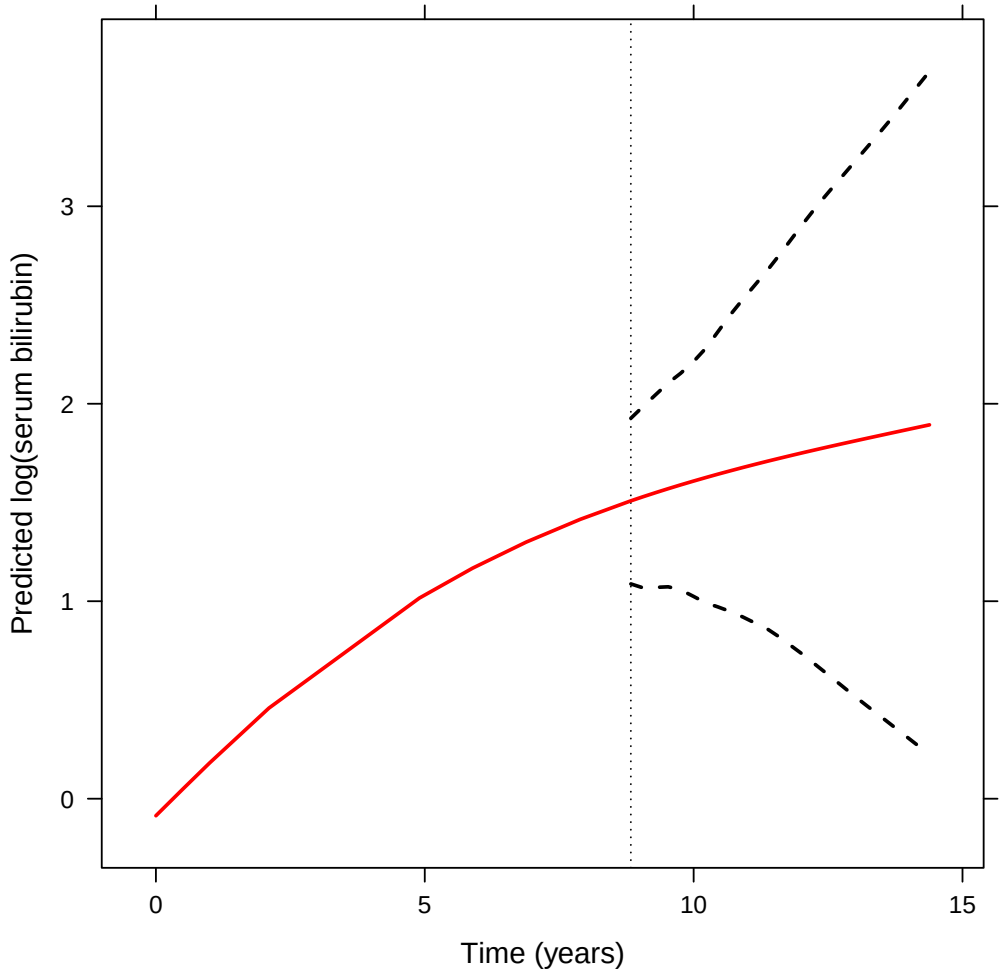

Figure 5: Predicted longitudinal trajectory (with a 95% pointwise confidence interval) for Patient 2 from the PBC dataset. The dotted line denotes the last time point Patient 2 was still event-free.

Web interface using ***shiny***

To facilitate the use of package **JMbayes** for deriving individualized predictions, a web interface has been written using package **shiny** (RStudio and Inc. 2014). This is available in the demo folder of the package and can be invoked with a call to the `runDynPred()` function, i.e.,

```
R> runDynPred()
```

With this interface users may load an R workspace with the fitted joint model(s), load the data of the new subject, and subsequently obtain dynamic estimates of $\pi_j(u \mid t)$ and $\omega_j(u \mid t)$ (i.e., an estimate after each longitudinal measurement). Several additional options are provided to calculate predictions based on different joint models (if the R workspace contains more than one model), to obtain estimates at specific horizon times, and to extract the dataset with the estimated conditional survival probabilities. A detailed description of the options of this app is provided in the 'Help' tab within the app.

5.2. Bayesian model averaging

Section 4.3 demonstrated that there are several choices to link the longitudinal and event time outcomes. When faced with this problem, the common practice in prognostic modeling is to base predictions on a single model that has been selected based on an automatic algorithm, such as, backward, forward or stepwise selection, or on likelihood-based information criteria, such as, AIC, BIC, DIC and their variants. However, what is often neglected in this procedure is the issue of model uncertainty. For example, if we choose a model using any of these criteria, say DIC, we usually treat it as the true model, even if there could be more than one model with DIC values of similar magnitude. In addition, when it comes to using a model for deriving predictions, we implicitly make the assumption that this model is adequate for all future patients. This seldom will be true in clinical practice. In our setting, a joint model with a specific formulation of the association structure may produce more accurate predictions for subjects with specific longitudinal profiles, while other models with other association structures may produce better predictions for subjects whose profiles have other characteristics. Here we follow another approach and we explicitly take into account model uncertainty by combining predictions under different association structures using Bayesian model averaging (BMA) (Hoeting, Madigan, Raftery, and Volinsky 1999; Rizopoulos, Hatfield, Carlin, and Takkenberg 2014).

We focus here on dynamic BMA predictions of survival probabilities. BMA predictions for the longitudinal outcome can be produced with similar methodology. Following the definitions of Section 5.1, we assume that we have available data $\mathcal{D}_n = \{T_i, \delta_i, \boldsymbol{y}_i; i = 1, \ldots, n\}$ based on which we fit $M_1, \ldots, M_K$ joint models with different association structures. Interest is in calculating predictions for a new subject j from the same population who has provided a set of longitudinal measurements $\mathcal{Y}_j(t)$, and has a vector of baseline covariates $\boldsymbol{w}_j$. We let $\mathcal{D}_j(t) = \{T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{w}_j\}$ denote the available data for this subject. The model-averaged probability of subject j surviving time $u > t$, given her survival up to t is given by the expression:

$$\Pr(T_j^* > u \mid \mathcal{D}_j(t), \mathcal{D}_n) = \sum_{k=1}^{K} \Pr(T_j^* > u \mid M_k, \mathcal{D}_j(t), \mathcal{D}_n)\, p(M_k \mid \mathcal{D}_j(t), \mathcal{D}_n). \tag{10}$$

The first term in the right-hand side of (10) denotes the model-specific survival probabilities, derived in Section 5.1, and the second term denotes the posterior weights of each of the competing joint models. The unique characteristic of these weights is that they depend on the observed data of subject j, in contrast to classic applications of BMA where the model weights depend only on $\mathcal{D}_n$ and are the same for all subjects. This means that, in our case, the model weights are both subject- and time-dependent, and therefore, for different subjects, and even for the same subject but at different times points, different models may have higher posterior probabilities (Rizopoulos *et al.* 2014). Hence, this framework is capable of better tailoring predictions to each subject than standard prognostic models, because at any time point we base risk assessments on the models that are more probable to describe the association between the observed longitudinal trajectory of a subject and the risk for an event.

For the calculation of the model weights we observe that these are written as (Rizopoulos *et al.* 2014):

$$p(M_k \mid \mathcal{D}_j(t), \mathcal{D}_n) = \frac{p(\mathcal{D}_j(t) \mid M_k)\, p(\mathcal{D}_n \mid M_k)\, p(M_k)}{\sum_{\ell=1}^{K} p(\mathcal{D}_j(t) \mid M_\ell)\, p(\mathcal{D}_n \mid M_\ell)\, p(M_\ell)},$$

where

$$p(\mathcal{D}_j(t) \mid M_k) = \int p(\mathcal{D}_j(t) \mid \tilde{\boldsymbol{\theta}}_k)\, p(\tilde{\boldsymbol{\theta}}_k \mid M_k)\, d\tilde{\boldsymbol{\theta}}_k$$

with $\tilde{\boldsymbol{\theta}}_k^\top = (\boldsymbol{\theta}_k^\top, \boldsymbol{b}_j^\top)$, and $p(\mathcal{D}_n \mid M_k)$ defined analogously. The likelihood part $p(\mathcal{D}_n \mid \boldsymbol{\theta}_k)$ is based on (6), and similarly $p(\mathcal{D}_j(t) \mid \boldsymbol{\theta}_k)$ equals

$$p(\mathcal{D}_j(t) \mid \tilde{\boldsymbol{\theta}}_k) = p(\mathcal{Y}_j(t) \mid \boldsymbol{b}_j, \boldsymbol{\theta}_k)\, S_j(t \mid \boldsymbol{b}_j, \boldsymbol{\theta}_k)\, p(\boldsymbol{b}_j \mid \boldsymbol{\theta}_k).$$

Thus, the subject-specific information in the model weights at time t comes from the available longitudinal measurements $\mathcal{Y}_j(t)$ but also from the fact that this subject has survived up to t. We should note that the new subject j does not contribute any new information about $\boldsymbol{\theta}_k$ (i.e., we do not refit the models using the data of this subject), the information for the parameters only comes from the original dataset in which the joint models have been fitted via the posterior distribution $p(\boldsymbol{\theta}_k \mid \mathcal{D}_n, M_k)$. For more information we refer the interested reader to Rizopoulos *et al.* (2014). A priori we assume that all models are equally probable, i.e., $p(M_k) = 1/K$, for all $k = 1, \ldots, K$. Closed-form expressions for the marginal densities $p(\mathcal{D}_n \mid M_k)$ and $p(\mathcal{D}_j(t) \mid M_k)$ are obtained by means of Laplace approximations (Tierney and Kadane 1986) performed in two-steps, namely, first integrating out the random effects and then the parameters.

In package **JMbayes** BMA predictions for either the survival or longitudinal outcome can be calculated using function `bma.combine()`. This function accepts a series or a list of objects returned by either `survfitJM()` or `predict()` and a vector of posterior model weights, and returns a single object of the same class as the input objects with the combined predictions. We illustrate how this function can be used to produce the BMA prediction of $\pi_j(u \mid t)$ using the first five measurements of Patient 2 from the PBC dataset based on joint models `jointFit.pbc1`, `jointFit.pbc12`, `jointFit.pbc13`, `jointFit.pbc14`, and `jointFit.pbc15`. We start by computing the posterior model weights. As seen above for the calculation of these weights we need to compute the marginal densities $p(\mathcal{D}_n \mid M_k)$ and $p(\mathcal{D}_j(t) \mid M_k)$. The former is obtained using the `logLik()` method for `JMbayes` objects, and the latter using function `marglogLik()`. The following code illustrates how this can be achieved:

```
R> Models <- list(jointFit.pbc1, jointFit.pbc12, jointFit.pbc13,
+    jointFit.pbc14, jointFit.pbc15)

R> log.p.Dj.Mk <- sapply(Models, marglogLik, newdata = ND[1:5, ])
R> log.p.Dn.Mk <- sapply(Models, logLik, marginal.thetas = TRUE)
R> log.p.Mk <- log(rep(1/length(Models), length(Models)))
```

Argument `newdata` of `marglogLik()` is used to provide the available data $\mathcal{D}_j(t)$ of the j-th subject, whereas argument `marginal.thetas` is invoked in order the `logLik()` method to compute the marginal log-likelihood. As just mentioned, we should stress that `marglogLik()` and `logLik()` compute $\log p(\mathcal{D}_j(t) \mid M_k)$ and $\log p(\mathcal{D}_n \mid M_k)$, respectively. Hence, to calculate the weights we need to transform them back to the original scale, i.e.,

```
R> weightsBMA <- log.p.Dj.Mk + log.p.Dn.Mk + log.p.Mk
R> weightsBMA <- exp(weightsBMA - mean(weightsBMA))
R> weightsBMA <- weightsBMA / sum(weightsBMA)
```

From the model we have considered for the PBC data we see that model `jointFit.pbc14` practically dominates the weights; however, in other examples it can well be that more than one models contribute substantially in these weights (Rizopoulos *et al.* 2014). Following we calculate the conditional survival probabilities based on each model, using `survfitJM()`:

```
R> survPreds <- lapply(Models, survfitJM, newdata = ND[1:5, ])
```

and finally we combine them using the call to `bma.combine()`:

```
R> survPreds.BMA <- bma.combine(JMlis = survPreds, weights = weightsBMA)
R> survPreds.BMA

Prediction of Conditional Probabilities for Event
        based on 200 Monte Carlo samples

$`2`
     times   Mean Median  Lower  Upper
1   4.9009 1.0000 1.0000 1.0000 1.0000
1   5.0072 0.9936 0.9943 0.9842 0.9981
2   5.3336 0.9724 0.9758 0.9353 0.9919
3   5.6599 0.9493 0.9554 0.8849 0.9850
4   5.9862 0.9242 0.9325 0.8306 0.9775
5   6.3126 0.8973 0.9071 0.7745 0.9691
6   6.6389 0.8686 0.8803 0.7141 0.9599
7   6.9652 0.8384 0.8520 0.6518 0.9497
8   7.2916 0.8069 0.8220 0.5906 0.9386
9   7.6179 0.7744 0.7901 0.5327 0.9274
10  7.9442 0.7412 0.7582 0.4719 0.9149
11  8.2706 0.7077 0.7247 0.4136 0.9009
12  8.5969 0.6743 0.6931 0.3585 0.8871
```

```
13  8.9232 0.6410 0.6617 0.3064 0.8721
14  9.2496 0.6083 0.6298 0.2583 0.8560
15  9.5759 0.5763 0.6002 0.2144 0.8392
16  9.9022 0.5452 0.5679 0.1750 0.8220
17 10.2286 0.5152 0.5344 0.1406 0.8011
18 10.5549 0.4863 0.5029 0.1147 0.7830
19 10.8812 0.4587 0.4718 0.0997 0.7684
20 11.2076 0.4324 0.4464 0.0870 0.7502
```

5.3. Predictive accuracy

The assessment of the predictive performance of time-to-event models has received a lot of attention in the statistical literature. In general two main lines have emerged, namely one focusing on calibration, i.e., how well the model predicts the observed data (Schemper and Henderson 2000; Gerds and Schumacher 2006) and a second one focusing on discrimination, i.e., how well can the model discriminate between patients that had the event from patients that did not (Harrell, Kerry, and Mark 1996; Pencina, D'Agostino, D'Agostino, and Vasan 2008). In the following we present discrimination and calibration measures suitably adapted to the dynamic prediction setting and their implementation in **JMbayes**.

Discrimination

To measure the discriminative capability of a longitudinal marker we focus on a time interval of medical relevance within which the occurrence of events is of interest. In this setting, a useful property of the model would be to successfully discriminate between patients who are going to experience the event within this time frame from patients who will not. To put this formally, as before, we assume that we have collected longitudinal measurements $\mathcal{Y}_j(t) = \{y_j(t_{jl}); 0 \leq t_{jl} \leq t, l = 1, \ldots, n_j\}$ up to time point t for subject j. We are interested in events occurring in the medically-relevant time frame $(t, t + \Delta t]$ within which the physician can take an action to improve the survival chance of the patient. Under the assumed model and the methodology presented in Section 5.1, we can define a prediction rule using $\pi_j(t + \Delta t \mid t)$ that takes into account the available longitudinal measurements $\mathcal{Y}_j(t)$. In particular, for any value c in $[0, 1]$ we can term subject j as a case if $\pi_j(t + \Delta t \mid t) \leq c$ (i.e., occurrence of the event) and analogously as a control if $\pi_j(t + \Delta t \mid t) > c$. Thus, in this context, we define sensitivity and specificity as

$$\Pr\big\{\pi_j(t + \Delta t \mid t) \leq c \mid T_j^* \in (t, t + \Delta t]\big\},$$

and

$$\Pr\big\{\pi_j(t + \Delta t \mid t) > c \mid T_j^* > t + \Delta t\big\},$$

respectively. For a randomly chosen pair of subjects $\{i, j\}$, in which both subjects have provided measurements up to time t, the discriminative capability of the assumed model can be assessed by the area under the receiver operating characteristic curve (AUC), which is obtained for varying c and equals,

$$\text{AUC}(t, \Delta t) = \Pr\big[\pi_i(t + \Delta t \mid t) < \pi_j(t + \Delta t \mid t) \mid \{T_i^* \in (t, t + \Delta t]\} \cap \{T_j^* > t + \Delta t\}\big],$$

that is, if subject i experiences the event within the relevant time frame whereas subject j does not, then we would expect the assumed model to assign higher probability of surviving longer than $t+\Delta t$ for the subject who did not experience the event. To summarize the discriminating power of the assumed model over the whole follow-up period, we need to take into account that the number of subjects contributing to the comparison of the fitted $\pi_i(t+\Delta t \mid t)$ with the observed data is not the same for all time points t. Following an approach similar to Antolini, Boracchi, and Biganzoli (2005) and Heagerty and Zheng (2005), we can utilize a weighted average of AUCs, i.e.,

$$\mathrm{C}_{dyn}^{\Delta t} = \int_0^\infty \mathrm{AUC}(t, \Delta t)\ \Pr\{\mathcal{E}(t)\}\ dt \Big/ \int_0^\infty \Pr\{\mathcal{E}(t)\}\ dt, \tag{11}$$

where $\mathcal{E}(t) = \big[\{T_i^* \in (t, t+\Delta t]\} \cap \{T_j^* > t+\Delta t\}\big]$, and $\Pr\{\mathcal{E}(t)\}$ denotes the probability that a random pair is comparable at t. We can call $\mathrm{C}_{dyn}^{\Delta t}$ a dynamic concordance index since it summarizes the concordance probabilities over the follow-up period. Note also that $\mathrm{AUC}(t, \Delta t)$ and as a result also $\mathrm{C}_{dyn}^{\Delta t}$ depend on the length Δt of the time interval of interest, which implies that different models may exhibit different discrimination power for different Δt.

For the estimation of $\mathrm{AUC}(t, \Delta t)$ and $\mathrm{C}_{dyn}^{\Delta t}$ we need to take care of two issues, namely, the calculation of the integrals in the definition of (11) and censoring. For the former we use the 15-point Gauss-Kronrod quadrature rule. Estimation of $\mathrm{AUC}(t, \Delta t)$ is directly based on its definition, namely by appropriately counting the concordant pairs of subjects. More specifically, we have

$$\widehat{\mathrm{AUC}}(t, \Delta t) = \widehat{\mathrm{AUC}}_1(t, \Delta t) + \widehat{\mathrm{AUC}}_2(t, \Delta t).$$

$\widehat{\mathrm{AUC}}_1(t, \Delta t)$ refers to the pairs of subjects who are comparable (i.e, their observed event times can be ordered),

$$\Omega_{ij}^{(1)}(t) \quad = \quad \big[\{T_i \in (t, t+\Delta t]\} \cap \{\delta_i = 1\}\big] \cap \{T_j > t+\Delta t\},$$

where $i, j = 1, \ldots, n$ with $i \neq j$. For such comparable subjects i and j, we can estimate and compare their survival probabilities $\pi_i(t+\Delta t \mid t)$ and $\pi_j(t+\Delta t \mid t)$, based on the methodology presented in Section 5.1. This leads to a natural estimator for $\mathrm{AUC}_1(t, \Delta t)$ as the proportion of concordant subjects out of the set of comparable subjects at time t:

$$\widehat{\mathrm{AUC}}_1(t, \Delta t) = \frac{\sum_{i=1}^n \sum_{j=1; j\neq i}^n I\{\hat{\pi}_i(t+\Delta t \mid t) < \hat{\pi}_j(t+\Delta t \mid t)\} \times I\{\Omega_{ij}^{(1)}(t)\}}{\sum_{i=1}^n \sum_{j=1; j\neq i}^n I\{\Omega_{ij}^{(1)}(t)\}},$$

where $I(\cdot)$ denotes the indicator function. Analogously, $\widehat{\mathrm{AUC}}_2(t, \Delta t)$ refers to the pairs of subjects who due to censoring cannot be compared, namely

$$\Omega_{ij}^{(2)}(t) = \big[\{T_i \in (t, t+\Delta t]\} \cap \{\delta_i = 0\}\big] \cap \{T_j > t+\Delta t\},$$

with again $i, j = 1, \ldots, n$ with $i \neq j$. Concordant subjects in this set contribute to the overall AUC appropriately weighted with the probability that they would be comparable, i.e.,

$$\widehat{\mathrm{AUC}}_2(t, \Delta t) = \frac{\sum_{i=1}^n \sum_{j=1; j\neq i}^n I\{\hat{\pi}_i(t+\Delta t \mid t) < \hat{\pi}_j(t+\Delta t \mid t)\} \times I\{\Omega_{ij}^{(2)}(t)\} \times \hat{\nu}_i(t+\Delta t \mid T_i)}{\sum_{i=1}^n \sum_{j=1; j\neq i}^n I\{\Omega_{ij}^{(2)}(t)\} \times \hat{\nu}_i(t+\Delta t \mid T_i)},$$

with $\hat{\nu}_i(t+\Delta t \mid T_i) = 1 - \hat{\pi}_i(t+\Delta t \mid T_i)$ being the probability that subject i who survived up to time T_i will have the event before $t+\Delta t$.

Having estimated $\text{AUC}(t, \Delta t)$, the next step in estimating $\text{C}_{dyn}^{\Delta t}$ is to obtain estimates for the weights $\Pr\{\mathcal{E}(t)\}$. We observe that these can be rewritten as

$$\begin{aligned}\Pr\{\mathcal{E}(t)\} &= \Pr\big[\{T_i^* \in (t, t+\Delta t]\} \cap \{T_j^* > t+\Delta t\}\big] \\ &= \Pr(T_i^* \in (t, t+\Delta t]) \times \Pr(T_j^* > t+\Delta t) \\ &= \big\{S(t) - S(t+\Delta t)\big\} S(t+\Delta t),\end{aligned}$$

where the simplification in the second line comes from the independence of subjects i and j, and $S(\cdot)$ here denotes the marginal survival function. In practice calculation of $\text{C}_{dyn}^{\Delta t}$ is restricted into a follow-up interval $[0, t_{max}]$ where we have information. Let $t_1, \ldots, t_{15}$ denote the re-scaled abscissas of the Gauss-Kronrod rule in the interval $[0, t_{max}]$ with corresponding weights $\varpi_1, \ldots, \varpi_{15}$. We combine the estimates $\widehat{\text{AUC}}(t_k, \Delta t)$, $k = 1, \ldots, 15$ with the estimates of the weights $\Pr\{\mathcal{E}(t)\}$ to obtain

$$\widehat{\text{C}}_{dyn}^{\Delta t} = \frac{\sum_{k=1}^{15} \varpi_k \widehat{\text{AUC}}(t_k, \Delta t) \times \widehat{\Pr}\{\mathcal{E}(t_k)\}}{\sum_{k=1}^{15} \varpi_k \widehat{\Pr}\{\mathcal{E}(t_k)\}},$$

where $\widehat{\Pr}\{\mathcal{E}(t_k)\} = \big\{\widehat{S}(t_k) - \widehat{S}(t_k + \Delta t)\big\}\widehat{S}(t_k + \Delta t)$, with $\widehat{S}(\cdot)$ denoting here the Kaplan-Meier estimate of the marginal survival function $S(\cdot)$.

The $\text{AUC}(t, \Delta t)$ and the dynamic discrimination index can be calculated for joint models fitted by `jointModelBayes()` using functions `aucJM()` and `dynCJM()`, respectively. In addition, function `rocJM()` computes the time-dependent sensitivity and specificity defined above, and its `plot()` method produces the plot of the receiver operating characteristic (ROC) curve. We illustrate their use based again on joint model `jointFit.pbc15`. The basic call to either `aucJM()` or `rocJM()` requires the user to provide the fitted joint model object, the data frame upon which the AUC or the ROC curve is to be calculated, the time point t (argument `Tstart`) up to which longitudinal measurements are to be used and the length of the time window Δt (argument `Dt`)[4]. The following call to `rocJM()` and its `plot()` method produces Figure 6.

```
R> roc.pbc15 <- rocJM(jointFit.pbc15, newdata = pbc2, Tstart = 5, Dt = 2)
R> plot(roc.pbc15)
```

The area under this ROC curve is calculated with the similar call to `aucJM()`:

```
R> auc.pbc15 <- aucJM(jointFit.pbc15, newdata = pbc2, Tstart = 5, Dt = 2)
R> auc.pbc15

        Time-dependent AUC for the Joint Model jointFit.pbc15

Estimated AUC: 0.8428
At time: 7
Using information up to time: 5 (202 subjects still at risk)
```

[4]Instead of giving `Dt` the user may also opt to directly provide the horizon time $t + \Delta t$ in the argument `Thoriz`.

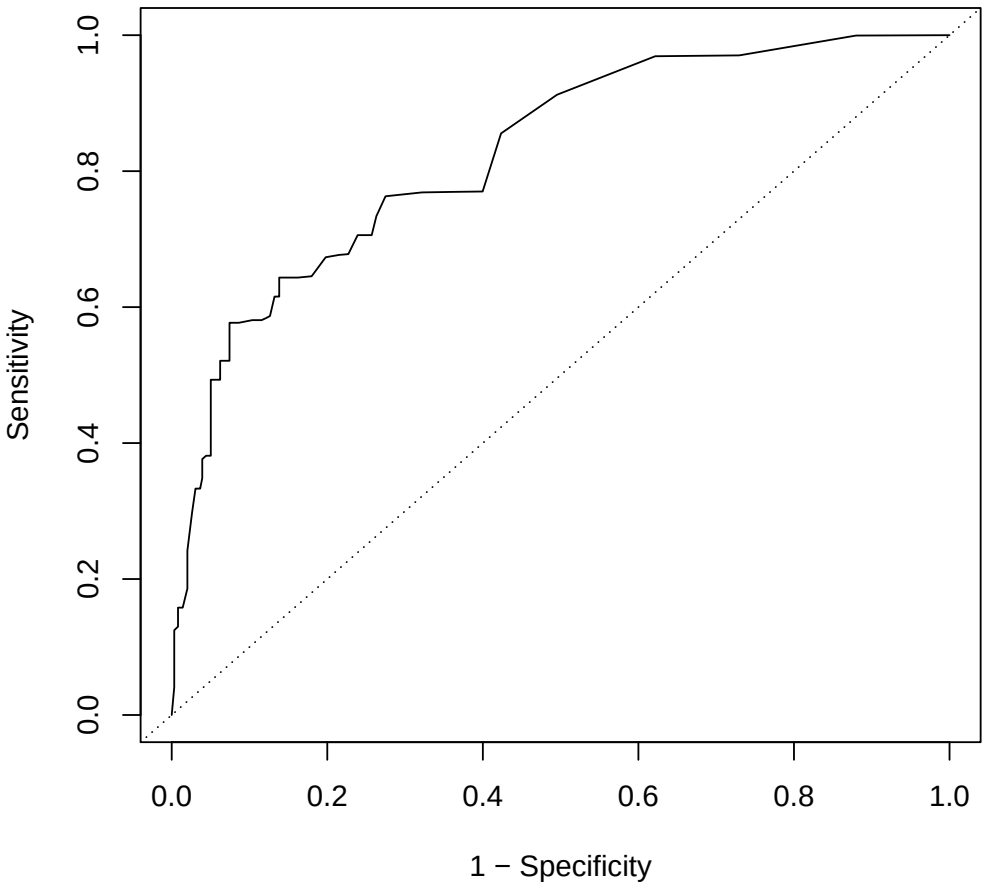

Figure 6: ROC curve at follow-up time $t = 5$ years, and time window $\Delta t = 2$ years based on joint model `jointFit.pbc15`.

We observe that using the first five year longitudinal measurements, serum bilirubin exhibits good discrimination capabilities for patients who are to die within a two-year time frame. To investigate if this is also the case during the whole follow-up period, we calculate the dynamic discrimination index for the same time window. The syntax of `dynCJM()` is (almost) identical to the one of `aucJM()`, i.e., (time to calculate 11 min),

```
R> dynC.pbc15 <- dynCJM(jointFit.pbc15, newdata = pbc2, Dt = 2)
R> dynC.pbc15

        Dynamic Discrimination Index for the Joint Model jointFit.pbc15

Estimated dynC: 0.8504
In the time interval: [0, 14.3057]
Length of time interval: 2
```

The estimate of $\mathrm{C}_{dyn}^{\Delta t=2}$ is almost identical to the one of $\mathrm{AUC}(t = 5, \Delta t = 2)$ indicating that serum bilirubin can discriminate well between patients during follow-up.

Prediction error

The assessment of the accuracy of predictions of survival models is typically based on the expected error of predicting future events. In our setting, and again taking into account the dynamic nature of the longitudinal outcome, it is of interest to predict the occurrence of events at $u > t$ given the information we have recorded up to time t. This gives rise to expected prediction error:

$$\mathrm{PE}(u \mid t) = E\big[L\{N_i(u) - \pi_i(u \mid t)\}\big],$$

where $N_i(t) = I(T_i^* > t)$ is the event status at time t, $L(\cdot)$ denotes a loss function, such as the absolute or square loss, and the expectation is taken with respect to the distribution of the event times. An estimate of $\mathrm{PE}(u \mid t)$ that accounts for censoring has been proposed by Henderson, Diggle, and Dobson (2002):

$$\begin{aligned}\widehat{\mathrm{PE}}(u \mid t) = \{n(t)\}^{-1} \sum_{i:T_i \geq t} & I(T_i \geq u) L\{1 - \hat{\pi}_i(u \mid t)\} + \delta_i I(T_i < u) L\{0 - \hat{\pi}_i(u \mid t)\} \\ & + (1 - \delta_i) I(T_i < u) \Big[\hat{\pi}_i(u \mid T_i) L\{1 - \hat{\pi}_i(u \mid t)\} + \{1 - \hat{\pi}_i(u \mid T_i)\} L\{0 - \hat{\pi}_i(u \mid t)\}\Big],\end{aligned}$$

where $n(t)$ denotes the number of subjects at risk at time t. The first two terms in the sum correspond to patients who were alive after time u and dead before u, respectively; the third term corresponds to patients who were censored in the interval $[t, u]$. Using the longitudinal information up to time t, $\mathrm{PE}(u \mid t)$ measures the predictive accuracy at the specific time point u. Alternatively, we could summarize the error of prediction in a specific interval of interest, say $[t, u]$, by calculating a weighted average of $\{\mathrm{PE}(s \mid t), t < s < u\}$ that corrects for censoring, similarly to $\mathrm{C}_{dyn}^{\Delta t}$. An estimator of this type for the integrated prediction error has been suggested by Schemper and Henderson (2000), which adapted to our time-dynamic setting takes the form

$$\widehat{\mathrm{IPE}}(u \mid t) = \frac{\sum_{i:t \leq T_i \leq u} \delta_i \big\{\widehat{S}_C(t)/\widehat{S}_C(T_i)\big\} \widehat{\mathrm{PE}}(T_i \mid t)}{\sum_{i:t \leq T_i \leq u} \delta_i \big\{\widehat{S}_C(t)/\widehat{S}_C(T_i)\big\}},$$

where $\widehat{S}_C(\cdot)$ denotes the Kaplan-Meier estimator of the censoring time distribution.

Both PE and IPE can be calculated for joint models fitted by `jointModelBayes()` using function `prederrJM()`. This has a similar syntax as function `aucJM()`, and requires a fitted joint model, a data frame based on which the prediction error will be calculated, and the time points t (argument `Tstart`) and u (argument `Thoriz`) that denotes up to which time point to use the longitudinal information and at which time point to make the prediction, respectively. For model `jointFit.pbc15` using the biomarker information during the first five years of follow-up the estimated prediction error at year seven is

```
R> pe.pbc15 <- prederrJM(jointFit.pbc15, pbc2, Tstart = 5, Thoriz = 7)
R> pe.pbc15

Prediction Error for the Joint Model jointFit.pbc15

Estimated prediction error: 0.1065
At time: 7
Using information up to time: 5 (202 subjects still at risk)
Loss function: square
```

By default the loss function is the square one (i.e., $L(x) = x^2$), but the user may specify the absolute loss or define her own loss function using argument `lossFun`. The integrated prediction error can be simply calculated by setting logical argument `interval` to `TRUE` in the call to `prederrJM()`; for example, for the same joint model and in the interval $[5, 9]$ the IPE is calculated with the code (time to calculate 42 min):

```
R> ipe.pbc15 <- prederrJM(jointFit.pbc15, pbc2, Tstart = 5,
+    Thoriz = 9, interval = TRUE)
R> ipe.pbc15

Prediction Error for the Joint Model jointFit.pbc15

Estimated prediction error: 0.0902
In the time interval: [5, 9]
Using information up to time: 5 (202 subjects still at risk)
Loss function: square
```

Validation

In the previous sections we have seen how the predictive performance of model `jointFit.pbc15` can be assessed in terms of discrimination and calibration on the PBC dataset. However, as it is know from the prognostic models literature (see e.g., Harrell 2001), these estimates of predictive performance may be over-optimistic because they do not account for the fact that the model was also fitted in the same dataset. One standard approach to obtain better, more objective, estimates of predictive ability is to utilize the cross-validation technique. The following code illustrates how we could implement 10-fold cross-validation using package **parallel**. First, we load the package and create 10 random splits of the PBC dataset:

```
R> library("parallel")
R> set.seed(123)
R> V <- 10
R> n <- nrow(pbc2.id)
R> splits <- split(seq_len(n), sample(rep(seq_len(V), length.out = n)))
```

Following we define a function that takes as argument the above defined splits, creates the training and testing datasets, fits joint model `jointFit.pbc15` in the training dataset, and calculates the AUC and the PE in the test dataset:

```
R> CrossValJM <- function (i) {
+    library("JMbayes")
+    pbc2$status2 <- as.numeric(pbc2$status != "alive")
+    pbc2.id$status2 <- as.numeric(pbc2.id$status != "alive")
+
+    trainingData <- pbc2[!pbc2$id %in% i, ]
+    trainingData.id <- trainingData[!duplicated(trainingData$id), ]
+    testingData <- pbc2[pbc2$id %in% i, ]
+
+    lmeFit.pbc1 <- lme(log(serBilir) ~ ns(year, 2), data = trainingData,
+      random = ~ ns(year, 2) | id)
+    coxFit.pbc1 <- coxph(Surv(years, status2) ~ drug * age,
+      data = trainingData.id, x = TRUE)
+
+    dForm <- list(fixed = ~ 0 + dns(year, 2),
+      random = ~ 0 + dns(year, 2),
+      indFixed = 2:3, indRandom = 2:3)
+    tf1 <- function (x, data) {
+      cbind(x, "^2" = x*x)
+    }
+    tf2 <- function (x, data) {
+      cbind(x, "drugD-penicil" = x * (data$drug == 'D-penicil'))
+    }
+    jointFit.pbc15 <-
+      jointModelBayes(lmeFit.pbc1, coxFit.pbc1, timeVar = "year",
+        param = "td-both", extraForm = dForm,
+        transFun = list(value = tf1, extra = tf2))
+
+    auc <- aucJM(jointFit.pbc15, newdata = testingData,
+      Tstart = 5, Thoriz = 7)
+    pe <- prederrJM(jointFit.pbc15, newdata = testingData,
+      Tstart = 5, Thoriz = 7)
+    list(auc = auc, pe = pe)
+  }
```

We run function `CrossValJM()` in parallel using five processors/cores by first creating the corresponding cluster and then using `parLapply()` (time to run 10.7 min):

```
R> cl <- makeCluster(5)
R> res <- parLapply(cl, splits, CrossValJM)
R> stopCluster(cl)
```

The averaged AUCs and PEs from the 10 random splits of the PBC dataset are calculated with the code:

```
R> mean(sapply(res, function (x) x$auc$auc))

[1] 0.8362527

R> mean(sapply(res, function (x) x$pe$prederr))

[1] 0.121802
```

We observe that the cross-validated estimate of the AUC is identical the one obtained in the original dataset, whereas for the prediction error there is a slight over-optimism.

6. Future plans

In this paper we have illustrated the capabilities of package **JMbayes** for fitting joint models for longitudinal and time-to-event data under a Bayesian approach. As we have seen, the current version of the package provides extensive options for fitting different types of joint models, but nonetheless several extensions are planned in the future to further expand on what is currently available. These include among others:

- Make the MCMC algorithm faster by implementing its time consuming pieces in C++.
- Extend functionality in the survival submodel to handle, competing risks, recurrent events and interval-censored event time data.
- The consideration of multiple longitudinal outcomes, while allowing for the various association structures we have presented in Sections 4.3 and 4.4.
- Update dynamic predictions to handle the aforementioned extensions.

References

Antolini L, Boracchi P, Biganzoli E (2005). "A time-dependent discrimination index for survival data." *Statistics in Medicine*, **24**, 3927–3944.

Brown E (2009). "Assessing the association between trends in a biomarker and risk of event with an application in pediatric HIV/AIDS." *The Annals of Applied Statistics*, **3**, 1163–1182.

Brown E, Ibrahim J, DeGruttola V (2005). "A flexible B-spline model for multiple longitudinal biomarkers and survival." *Biometrics*, **61**, 64–73.

Crowther M (2013). ***STJM**: Stata module to fit shared parameter joint models of longitudinal and survival data*. URL http://ideas.repec.org/c/boc/bocode/s457502.html.

Crowther M, Abrams K, Lambert P (2013). "Joint modeling of longitudinal and survival data." *The Stata Journal*, **13**, 165–184.

Dahl DB (2014). ***xtable**: Export tables to LaTeX or HTML*. R package version 1.7-3, URL http://CRAN.R-project.org/package=xtable.

Eilers P, Marx B (1996). "Flexible smoothing with B-splines and penalties." *Statistical Science*, **11**, 89–121.

Gerds T, Schumacher M (2006). "Consistent estimation of the expected Brier score in general survival models with right-censored event times." *Biometrical Journal*, **48**, 1029 – 1040.

Guo X, Carlin B (2004). "Separate and joint modeling of longitudinal and event time data using standard computer packages." *The American Statistician*, **58**, 16–24.

Harrell F (2001). *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer-Verlag, New York.

Harrell F, Kerry L, Mark D (1996). "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." *Statistics in Medicine*, **15**, 361–387.

Heagerty P, Zheng Y (2005). "Survival model predictive accuracy and ROC curves." *Biometrics*, **61**, 92–105.

Henderson R, Diggle P, Dobson A (2000). "Joint modelling of longitudinal measurements and event time data." *Biostatistics*, **1**, 465–480.

Henderson R, Diggle P, Dobson A (2002). "Identification and efficacy of longitudinal markers for survival." *Biostatistics*, **3**, 33–50.

Hoeting J, Madigan D, Raftery A, Volinsky C (1999). "Bayesian model averaging: A tutorial." *Statistical Science*, **14**, 382–417.

Ibrahim J, Chen M, Sinha D (2001). *Bayesian Survival Analysis*. Springer-Verlag, New York.

Kalbfleisch J, Prentice R (2002). *The Statistical Analysis of Failure Time Data*. 2nd edition. John Wiley & Sons, New York.

Lang S, Brezger A (2004). "Bayesian P-splines." *Journal of Computational and Graphical Statistics*, **13**, 183–212.

Little R, Rubin D (2002). *Statistical Analysis with Missing Data*. 2nd edition. John Wiley & Sons, New York.

Molenberghs G, Kenward M (2007). *Missing Data in Clinical Studies*. John Wiley & Sons, New York.

Murtaugh P, Dickson E, Van Dam G, Malincho M, Grambsch P, Langworthy A, Gips C (1994). "Primary biliary cirrhosis: prediction of short-term survival based on repeated patient visits." *Hepatology*, **20**, 126–134.

Pencina M, D'Agostino Sr R, D'Agostino Jr R, Vasan R (2008). "Evaluating the added predictive ability of a new marker: From area under the ROC curve to reclassification and beyond." *Statistics in Medicine*, **27**, 157–172.

Philipson P, Sousa I, Diggle P, Williamson P, Kolamunnage-Dona R, Henderson R (2012). ***joineR**: Joint modelling of repeated measurements and time-to-event data*. R package version 1.0-3, URL http://CRAN.R-project.org/package=joineR.

Pinheiro J, Bates D, DebRoy S, Sarkar D, R Development Core Team (2014). ***nlme**: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-115, URL http://CRAN.R-project.org/package=nlme.

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/.

Proust-Lima C, Philipps V, Diakite A, Liquet B (2013). ***lcmm**: Estimation of latent class mixed models, joint latent class mixed models and mixed models for curvilinear outcomes*. R package version 1.6.3, URL http://CRAN.R-project.org/package=lcmm.

Rizopoulos D (2010). "**JM**: An R package for the joint modelling of longitudinal and time-to-event data." *Journal of Statistical Software*, **35 (9)**, 1–33. URL http://www.jstatsoft.org/v35/i09/.

Rizopoulos D (2011). "Dynamic predictions and prospective accuracy in joint models for longitudinal and time-to-event data." *Biometrics*, **67**, 819–829.

Rizopoulos D (2012). *Joint Models for Longitudinal and Time-to-Event Data, with Applications in R*. Chapman & Hall/CRC, Boca Raton.

Rizopoulos D (2014). ***JM**: Shared parameter models for the joint modeling of longitudinal and time-to-event data*. R package version 1.3-0, URL http://CRAN.R-project.org/package=JM.

Rizopoulos D, Hatfield L, Carlin P, Takkenberg J (2014). "Combining dynamic predictions from joint Models for longitudinal and time-to-event data using Bayesian model averaging." *Journal of the American Statistical Association*, **109**, 1385–1397.

Rizopoulos D, Lesaffre E (2014). "Introduction to the special issue on joint modelling techniques." *Statistical Methods in Medical Research*, **23**, 3–10.

RStudio, Inc (2014). ***shiny**: Web Application Framework for R*. R package version 0.9.1, URL http://CRAN.R-project.org/package=shiny.

Sarkar D (2008). *Lattice: Multivariate Data Visualization with R*. Springer-Verlag, New York.

Schemper M, Henderson R (2000). "Predictive accuracy and explained variation in Cox regression." *Biometrics*, **56**, 249–255.

Sylvestre MP, Abrahamowicz M (2009). "Flexible modeling of the cumulative effects of time-dependent exposures on the hazard." *Statistics in Medicine*, **28**, 3437 – 3453.

Taylor J, Park Y, Ankerst D, Proust-Lima C, Williams S, Kestin L, Bae K, Pickles T, Sandler H (2013). "Real-time individual predictions of prostate cancer recurrence using joint models." *Biometrics*, **69**, 206–213.

Therneau T, Grambsch P (2000). *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, New York.

Therneau T, Lumley T (2014). ***survival**: Survival Analysis Including Penalised Likelihood*. R package version 2.37-7, URL http://CRAN.R-project.org/package=survival.

Tierney L, Kadane J (1986). "Accurate approximations for posterior moments and marginal densities." *Journal of the American Statistical Association*, **81**, 82–86.

Tsiatis A, Davidian M (2004). "Joint modeling of longitudinal and time-to-event data: An overview." *Statistica Sinica*, **14**, 809–834.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Fourth edition. Springer-Verlag, New York.

Ye W, Lin X, Taylor J (2008). "Semiparametric modeling of longitudinal measurements and time-to-event data – a two stage regression calibration approach." *Biometrics*, **64**, 1238–1246.

Yu M, Taylor J, Sandler H (2008). "Individualized prediction in prostate cancer studies using a joint longitudinal-survival-cure model." *Journal of the American Statistical Association*, **103**, 178–187.

A. Sensitivity analysis number of knots

In this appendix we investigate the impact the number of knots for the baseline hazard can have in the derived results. The number of knots is controlled via the control argument `lng.in.kn = 10L`. The following code refits model `jointFit.pbc1`, presented in Section 4.1, using 10 and 20 knots (time to fit: 4.5 min and 4.5 min, respectively):

```
R> jointFit.pbc1.10knots <- update(jointFit.pbc1, lng.in.kn = 10L)
R> jointFit.pbc1.20knots <- update(jointFit.pbc1, lng.in.kn = 20L)
```

Following using function `fixef()` we extract the posterior means for the fixed effects of the longitudinal submodel:

```
R> cbind("10 knots" = fixef(jointFit.pbc1.10knots),
+    "15 knots" = fixef(jointFit.pbc1),
+    "20 knots" = fixef(jointFit.pbc1.10knots))
```

```
             10 knots  15 knots  20 knots
(Intercept)  0.5148687 0.5114541 0.5148687
ns(year, 2)1 2.2961662 2.2840843 2.2961662
ns(year, 2)2 2.1082737 2.0897856 2.1082737
```

and the regression coefficients of the survival submodel:

```
R> cbind("10 knots" = fixef(jointFit.pbc1, process = "Event"),
+    "15 knots" = fixef(jointFit.pbc1.10knots, process = "Event"),
+    "20 knots" = fixef(jointFit.pbc1.10knots, process = "Event"))
```

```
                      10 knots    15 knots    20 knots
drugD-penicil     -0.77755764 -0.84470619 -0.84470619
age                0.04039257  0.03891005  0.03891005
drugD-penicil:age  0.01249318  0.01532151  0.01532151
Assoct             1.40989077  1.40819672  1.40819672
```

We observe small differences between the three models.

B. MCMC diagnostic plots

The `plot()` method for objects produced by `jointModelBayes()` produces diagnostic plots for the MCMC, namely trace, auto-correlation and kernel density estimated plots. In addition, the `plot()` method can be used to create the figure of the Conditional Predictive Ordinate (CPO). As an example, we produce trace and density plots for the joint model `jointFit.pbc1` that was fitted in Section 4.1. To avoid lengthy output we just illustrate how these plots are produced for the parameters of the longitudinal submodel, and the regression coefficients in the survival submodel. The relevant code is:

```
R> plot(jointFit.pbc1, param = c("betas", "sigma", "alphas", "gammas"))
R> plot(jointFit.pbc1, which = "density",
+    param = c("betas", "sigma", "alphas", "gammas"))
```

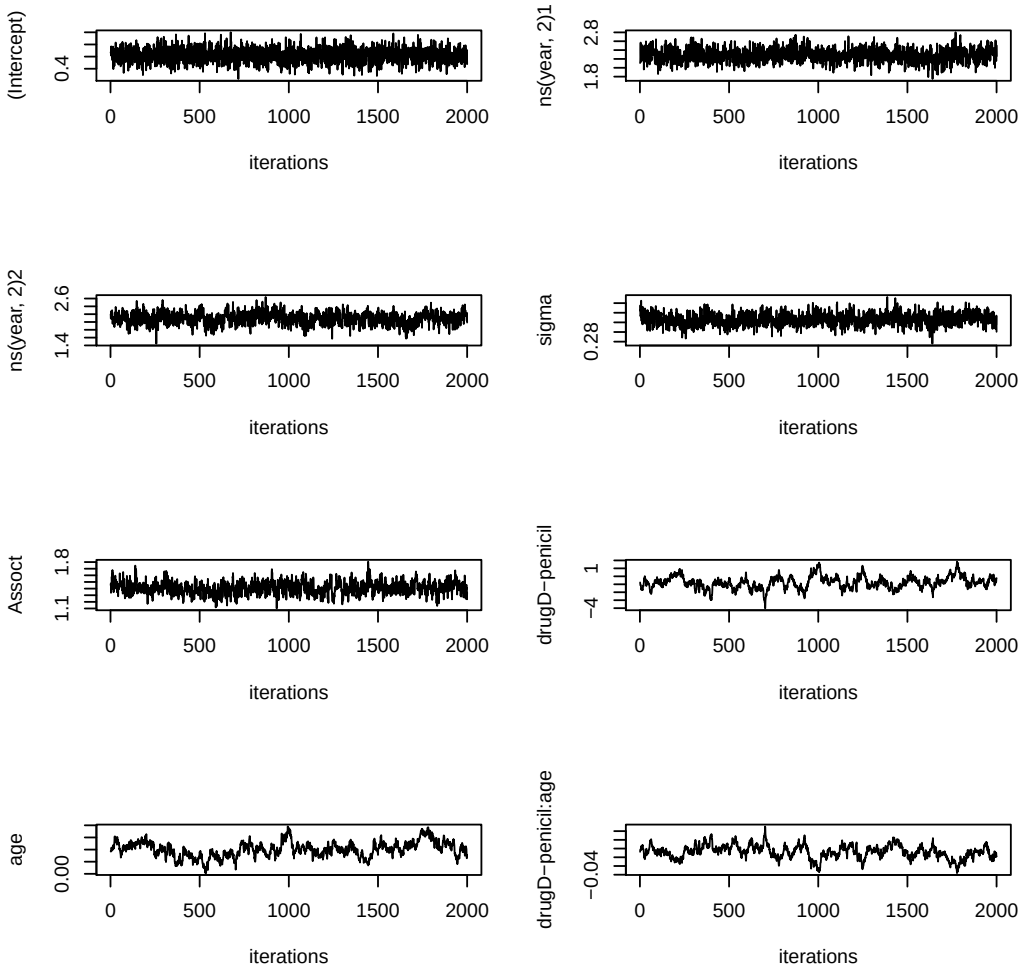

Figure 7: Traceplots for the parameters of the longitudinal (first four plots) and survival (last four plots) submodels from `jointFit.pbc1`.

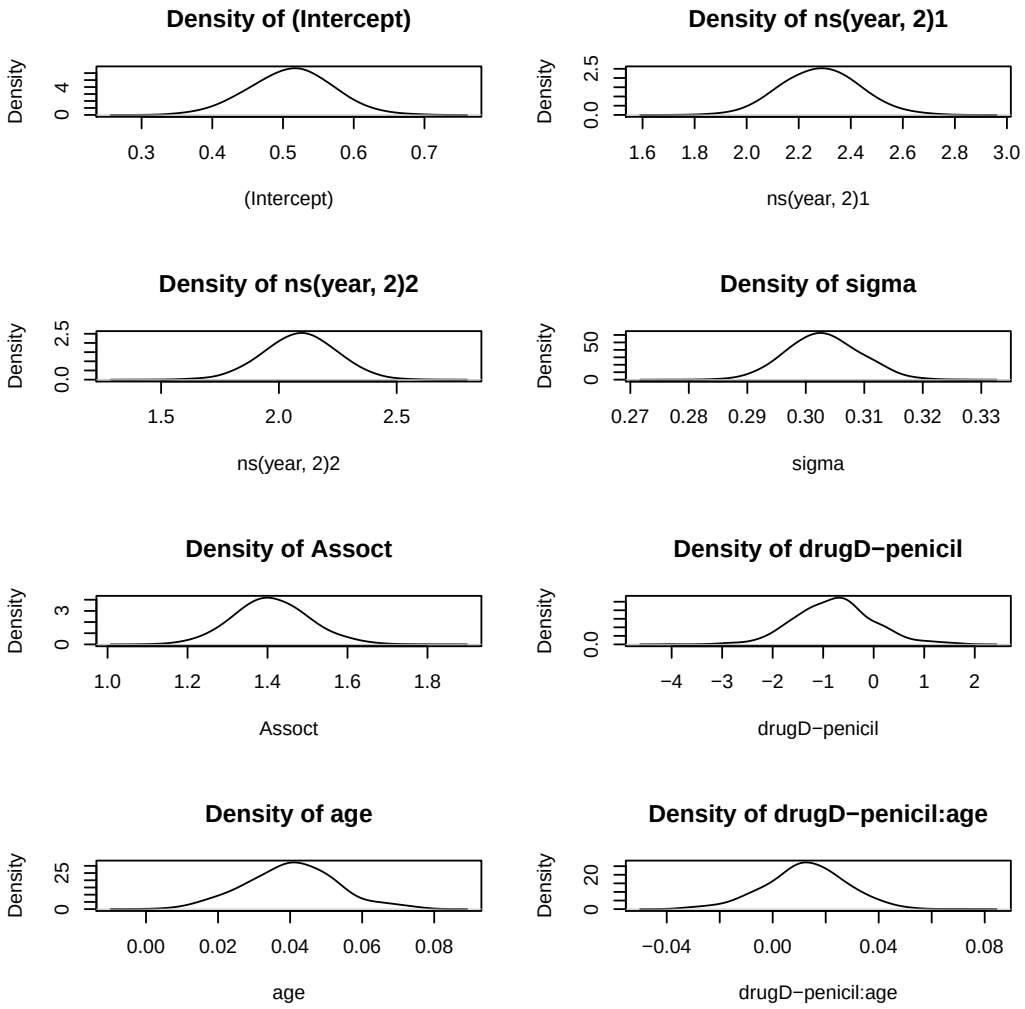

Figure 8: Kernel density estimation plots for the parameters of the longitudinal (first four plots) and survival (last four plots) submodels from `jointFit.pbc1`.

C. Joint models with left-truncation and exogenous time-varying covariates

In this appendix we illustrate how left-truncated data and additional exogenous time-varying covariates can be included in the survival submodel of a joint model. As an example we fit a joint model for the PBC dataset in which we also control for spiders. The first piece of code constructs the counting process `start`, `stop` variables that need to be supplied to the `Surv()` function.

```
> pbc <- pbc2[c("id", "serBilir", "drug", "year", "years",
+   "status2", "spiders")]
> pbc$start <- pbc$year
> splitID <- split(pbc[c("start", "years")], pbc$id)
> pbc$stop <- unlist(lapply(splitID,
+   function (d) c(d$start[-1], d$years[1]) ))
> pbc$event <- with(pbc, ave(status2, id,
+   FUN = function (x) c(rep(0, length(x)-1), x[1])))
> pbc <- pbc[!is.na(pbc$spiders), ]
```

We create artificial left-truncation by omitting the baseline measurement of each subject:

```
> pbc <- pbc[pbc$start != 0, ]
```

Then we proceed by fitting the linear mixed and time-varying Cox models. In the latter we need to use the dataset in the long format, and also include as cluster variable the subject id indicator:

```
> lmeFit.pbc <- lme(log(serBilir) ~ drug * ns(year, 2),
+   random = ~ ns(year, 2) | id, data = pbc)
> tdCox.pbc <- coxph(Surv(start, stop, event) ~ drug * spiders + cluster(id),
+   data = pbc, x = TRUE, model = TRUE)
```

The joint model is then fitted using a simple call to `jointModelBayes()`.

```
> jointFit.pbc <- jointModelBayes(lmeFit.pbc, tdCox.pbc, timeVar = "year")
> summary(jointFit.pbc)

Call:
jointModelBayes(lmeObject = lmeFit.pbc, survObject = tdCox.pbc,
    timeVar = "year")

Data Descriptives:
Longitudinal Process                Event Process
Number of Observations: 1575        Number of Events: 96 (33.7%)
Number of subjects: 285

Joint Model Summary:
Longitudinal Process: Linear mixed-effects model
Event Process: Relative risk model with penalized-spline-approximated
```

```
               baseline risk function
Parameterization: Time-dependent value

      LPML      DIC       pD
 -2527.554 4825.243 847.4221

Variance Components:
              StdDev    Corr
(Intercept)   1.0182  (Intr)  n(,2)1
ns(year, 2)1  1.9712  0.1354
ns(year, 2)2  1.6237 -0.0140  0.3377
Residual      0.2733

Coefficients:
Longitudinal Process
                               Value Std.Err Std.Dev    2.5%  97.5%      P
(Intercept)                   0.5100  0.0020  0.0876  0.3342 0.6783 <0.001
drugD-penicil                -0.1587  0.0028  0.1248 -0.3995 0.0796  0.200
ns(year, 2)1                  2.0087  0.0074  0.1906  1.6233 2.3890 <0.001
ns(year, 2)2                  1.1859  0.0098  0.1812  0.8303 1.5449 <0.001
drugD-penicil:ns(year, 2)1    0.0095  0.0084  0.2735 -0.5211 0.5479  0.980
drugD-penicil:ns(year, 2)2 -0.0420  0.0121  0.2495 -0.5216 0.4737  0.876

Event Process
                              Value Std.Err  Std.Dev    2.5%     97.5%      P
drugD-penicil              -0.3874  0.0232   0.3476 -1.0633    0.3017  0.280
spidersYes                  0.2947  0.0202   0.3069 -0.2888    0.9059  0.334
drugD-penicil:spidersYes    0.5240  0.0266   0.4524 -0.3557    1.4461  0.240
Assoct                      1.2690  0.0109   0.1156  1.0458    1.4882 <0.001
tauBs                     408.3443 64.3415 288.3054 79.2179 1130.6831     NA

MCMC summary:
iterations: 20000
adapt: 3000
burn-in: 3000
thinning: 10
time: 3.4 min
```

Affiliation:

Dimitris Rizopoulos
Department of Biostatistics
Erasmus Medical Center Rotterdam
PO Box 2040, 3000 CA Rotterdam, the Netherlands
E-mail: d.rizopoulos@erasmusmc.nl
URL: http://www.erasmusmc.nl/biostatistiek/People/Faculty/drizopoulos/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd