Joint longitudinal and time-to-event models via Stan

Sam Brilleman[*]*, Michael Crowther, Margarita Moreno-Betancur, Jacqueline Buros Novik, Rory Wolfe*

Abstract

The joint modelling of longitudinal and time-to-event data has received much attention in the biostatistical literature in recent years. In this notebook, we describe the implementation of a shared parameter joint model for longitudinal and time-to-event data in Stan. The methods described in the notebook are a simplified version of those underpinning the `stan_jm` modelling function that has recently been contributed to the **rstanarm** R package. This notebook will proceed as follows. In Section 1 we provide an introduction to the joint modelling of longitudinal and time-to-event data, including briefly describing the potential motivations for using such an approach. In Section 2 we describe the formulation of a multivariate shared parameter joint model and introduce it's log likelihood function. In Section 3 we describe some of the more important features of the Stan code required to implement the model. In Section 4 we present a short applied example to demonstrate estimation of the model and the type of inferences that can be obtained. In Section 5 we close with a discussion.

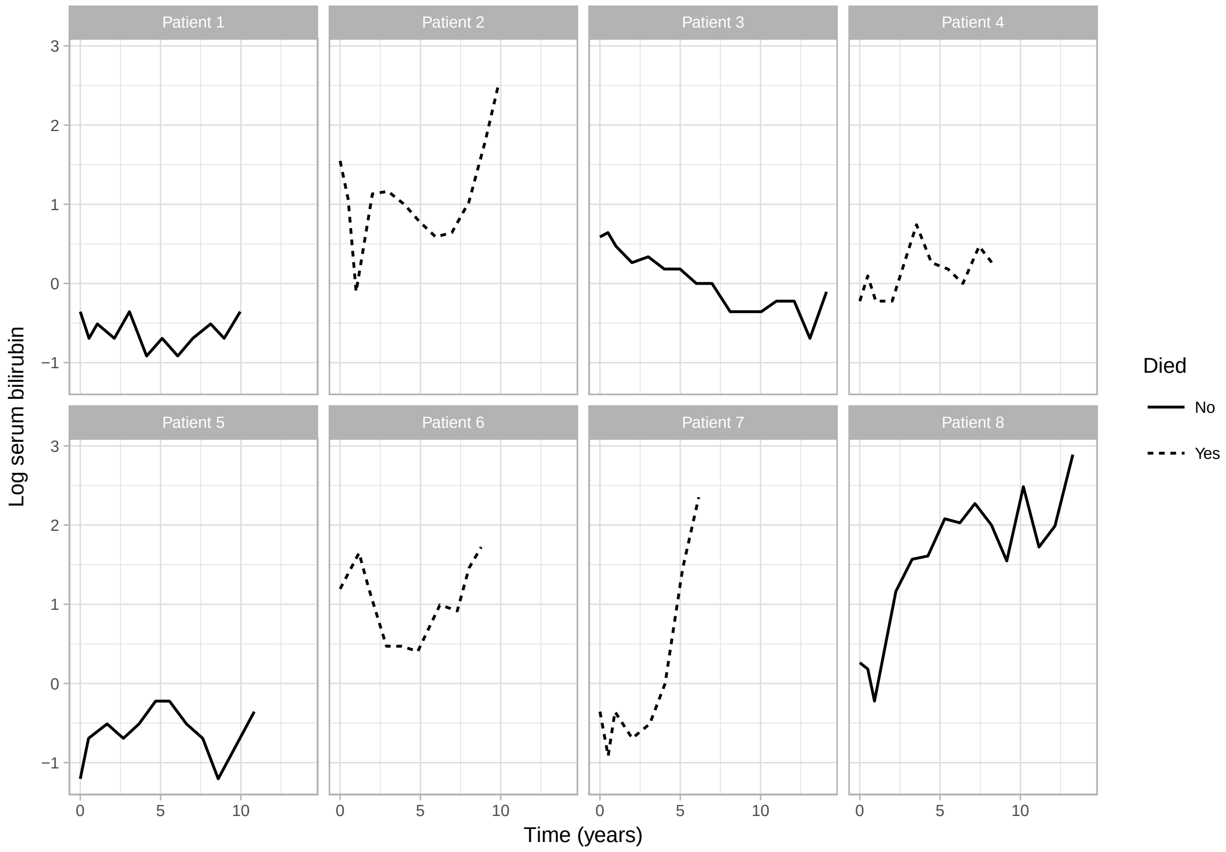
Date this notebook was compiled: 29 November 2017.

1 Introduction

Joint modelling can be broadly defined as the simultaneous estimation of two or more statistical models which traditionally would have been separately estimated. When we refer to a shared parameter joint model for longitudinal and time-to-event data, we generally mean the joint estimation of: 1) a longitudinal mixed effects model which analyses patterns of change in an outcome variable that has been measured repeatedly over time (for example, a clinical biomarker) and 2) a survival or time-to-event model which analyses the time until an event of interest occurs (for example, death or disease progression). Joint estimation of these so-called "submodels" is achieved by assuming they are correlated via individual-specific parameters (i.e. individual-level random effects).

Over the last two decades the joint modelling of longitudinal and time-to-event data has received a significant amount of attention [1-5]. Methodological developments in the area have been motivated by a growing awareness of the benefits that a joint modelling approach can provide. In clinical or epidemiological research it is common for a clinical biomarker to be repeatedly measured over time on a given patient. In addition, it is common for time-to-event data, such as the patient-specific time from a defined origin (e.g. time of diagnosis of a disease) until a terminating clinical event such as death or disease progression to also be collected. The figure below shows observed longitudinal measurements (i.e. observed "trajectories") of log serum bilirubin for a small sample of patients with primary biliary cirrhosis. Solid lines are used for those patients who were still alive at the end of follow up, while dashed lines are used for those patients who died. From the plots, we can observe between-patient variation in the longitudinal trajectories for log serum bilirubin, with some patients showing an increase in the biomarker over time, others decreasing, and some remaining stable. Moreover, there is variation between patients in terms of the frequency and timing of the longitudinal measurements.

[*]Corresponding author: sam.brilleman@monash.edu.



From the perspective of clinical risk prediction, we may be interested in asking whether the between-patient variation in the log serum bilirubin trajectories provides meaningful prognostic information that can help us differentiate patients with regard to some clinical event of interest, such as death. Alternatively, from an epidemiological perspective we may wish to explore the potential for etiological associations between changes in log serum bilirubin and mortality. Joint modelling approaches provide us with a framework under which we can begin to answer these types of clinical and epidemiological questions.

More formally, the motivations for undertaking a joint modelling analysis of longitudinal and time-to-event data might include one or more of the following:

- One may be interested in how *underlying changes in the biomarker influence the occurrence of the event.* However, including the observed biomarker measurements directly into a time-to-event model as time-varying covariates poses several problems. For example, if the widely used Cox proportional hazards model is assumed for the time-to-event model then biomarker measurements need to be available for all patients at all failure times, which is unlikely to be the case [3]. If simple methods of imputation are used, such as the "last observation carried forward" method, then these are likely to induce bias [6]. Furthermore, the observed biomarker measurements may be subject to measurement error and therefore their inclusion as time-varying covariates may result in biased and inefficient estimates. In most cases, the measurement error will result in parameter estimates which are shrunk towards the null [7]. On the other hand, joint modelling approaches allow us to estimate the association between the biomarker (or some function of the biomarker trajectory, such as rate of change in the biomarker) and the risk of the event, whilst allowing for both the discrete time and measurement-error aspects of the observed biomarker.
- One may be interested primarily in the evolution of the clinical biomarker but *may wish to account for what is known as informative dropout.* If the value of future (unobserved) biomarker measurements are related to the occurrence of the terminating event, then those unobserved biomarker measurements will be "missing not at random" [8,9]. In other words, biomarker measurements for patients who have an

event will differ from those who do not have an event. Under these circumstances, inference based solely on observed measurements of the biomarker will be subject to bias. A joint modelling approach can help to adjust for informative dropout and has been shown to reduce bias in the estimated parameters associated with longitudinal changes in the biomarker [1,9,10].

- Joint models are naturally suited to the task of *dynamic risk prediction*. For example, joint modelling approaches have been used to develop prognostic models where predictions of event risk can be updated as new longitudinal biomarker measurements become available. Taylor et al. [11] jointly modelled longitudinal measurements of the prostate specific antigen (PSA) and time to clinical recurrence of prostate cancer. The joint model was then used to develop a web-based calculator which could provide real-time predictions of the probability of recurrence based on a patient's up to date PSA measurements.

In this notebook, we describe the implementation of a shared parameter joint model for longitudinal and time-to-event data in Stan. In Section 2 we describe the formulation for a multivariate joint model, that is, a joint model for multiple (i.e. more than one) longitudinal biomarkers and the time to a terminating event. In Section 3 we describe the important features of the Stan code required to fit the model. In Section 4 we present a brief applied example to demonstrate estimation of the model and the type of inferences that can be obtained. Note that the methods and code described in this paper are a simplified version of the `stan_jm` modelling function that is being contributed to the **rstanarm** R package [12,13], see https://github.com/stan-dev/rstanarm or https://github.com/sambrilleman/rstanarm.

2 Model formulation

A shared parameter joint model consists of related submodels which are specified separately for each of the longitudinal and time-to-event outcomes. These are therefore commonly referred to as the *longitudinal submodel(s)* and the *event submodel*. The longitudinal and event submodels are linked using shared individual-specific parameters, which can be parameterised in a number of ways. We describe each of these submodels below.

2.1 Longitudinal submodel(s)

We assume $y_{ijm}(t) = y_{im}(t_{ij})$ corresponds to the observed value of the m^{th} $(m = 1, ..., M)$ biomarker for individual i $(i = 1, ..., N)$ taken at time point t_{ij}, $j = 1, ..., n_i$. We specify a (multivariate) generalised linear mixed model that assumes $y_{ijm}(t)$ follows a distribution in the exponential family with mean $\mu_{ijm}(t)$ and linear predictor

$$\eta_{ijm}(t) = g_m(\mu_{ijm}(t)) = \boldsymbol{x}^T_{ijm}(t)\boldsymbol{\beta}_m + \boldsymbol{z}^T_{ijm}(t)\boldsymbol{b}_{im} \tag{1}$$

where $\boldsymbol{x}^T_{ijm}(t)$ and $\boldsymbol{z}^T_{ijm}(t)$ are both row-vectors of covariates (which likely include some function of time, for example a linear slope, cubic splines, or polynomial terms) with associated vectors of fixed and individual-specific parameters $\boldsymbol{\beta}_m$ and $\boldsymbol{b}_{im}$, respectively, and g_m is some known link function.

The distribution and link function are allowed to differ over the M longitudinal submodels. We assume that the dependence across the different longitudinal submodel (i.e. the correlation between the different longitudinal biomarkers) is captured through a shared multivariate normal distribution for the individual-specific parameters; that is, we assume

$$\begin{pmatrix} \boldsymbol{b}_{i1} \\ \vdots \\ \boldsymbol{b}_{iM} \end{pmatrix} = \boldsymbol{b}_i \sim \mathsf{Normal}\left(0, \boldsymbol{\Sigma}\right) \tag{2}$$

for some unstructured variance-covariance matrix $\mathbf{\Sigma}$.

2.2 Event submodel

We assume that we also observe an event time $T_i = \mathsf{min}\,(T_i^*, C_i)$ where T_i^* denotes the so-called "true" event time for individual i (potentially unobserved) and C_i denotes the censoring time. We define an event indicator $d_i = I(T_i^* \leq C_i)$. We then model the hazard of the event using a parametric proportional hazards regression model of the form

$$h_i(t) = h_0(t)\mathsf{exp}\left(\boldsymbol{w}_i^T(t)\boldsymbol{\gamma} + \sum_{m=1}^{M}\sum_{q=1}^{Q_m}\alpha_{mq}f_{mq}(\boldsymbol{\beta}_m, \boldsymbol{b}_{im}; t)\right) \tag{3}$$

where $h_i(t)$ is the hazard of the event for individual i at time t, $h_0(t)$ is the baseline hazard at time t, $\boldsymbol{w}_i^T(t)$ is a row-vector of individual-specific covariates (possibly time-dependent) with an associated vector of regression coefficients $\boldsymbol{\gamma}$ (log hazard ratios), and the α_{mq} are also coefficients (log hazard ratios).

The longitudinal and event submodels are assumed to related via an "association structure" based on shared individual-specific parameters and captured via the $\sum_{m=1}^{M}\sum_{q=1}^{Q_m}\alpha_{mq}f_{mq}(\boldsymbol{\beta}_m, \boldsymbol{b}_{im}; t)$ term in the linear predictor of equation (3). The coefficients α_{mq} are referred to as the "association parameters" since they quantify the strength of the association between the longitudinal and event processes, while the $f_{mq}(\boldsymbol{\beta}_m, \boldsymbol{b}_{im}; t)$ (for some functions $f_{mq}(.)$) can be referred to as the "association terms" and can be specified in a variety of ways which we describe in the next section.

We assume that the baseline hazard $h_0(t)$ is modelled parametrically. For simplicity, in the formulation of the joint model presented in this notebook we will restrict ourselves to modelling the log baseline hazard using B-splines. Note however that in the **rstanarm** package's `stan_jm` modelling function the baseline hazard can be specified as either an approximation using B-splines (the default), a Weibull distribution, or a piecewise constant baseline hazard (sometimes referred to as piecewise exponential). In the case of the piecewise constant or B-splines baseline hazard, the user can control the flexibility by explicitly specifying the knot points or degrees of freedom.

2.3 Association structures

As mentioned in the previous section, the dependence between the longitudinal and event submodels is captured through the association structure, which can be specified in a number of ways. In this notebook, we focus on the simplest association structure

$$f_{mq}(\boldsymbol{\beta}_m, \boldsymbol{b}_{im}; t) = \eta_{im}(t) \tag{4}$$

This is often referred to as a *current value* association structure since it assumes that the log hazard of the event at time t is linearly associated with the value of the longitudinal submodel's linear predictor also evaluated at time t. This is the most common association structure used in the joint modelling literature to date. In the situation where the longitudinal submodel is based on an identity link function and normal error distribution (i.e. a linear mixed model) the *current value* association structure can be viewed as a method for including the underlying "true" value of the biomarker as a time-varying covariate in the event submodel.[1]

[1]By "true" value of the biomarker, we mean the value of the biomarker which is not subject to measurement error or discrete time observation. Of course, for the expected value from the longitudinal submodel to be considered the so-called "true" underlying biomarker value, we would need to have specified the longitudinal submodel appropriately!

However, there are a variety of other association structures that could be specified. For example, we could assume the log hazard of the event is linearly associated with the *current slope* (i.e. rate of change) of the longitudinal submodel's linear predictor, that is

$$f_{mq}(\boldsymbol{\beta}_m, \boldsymbol{b}_{im}; t) = \frac{d\eta_{im}(t)}{dt} \tag{5}$$

Moreover, there are a whole range of possible association structures, many of which have been discussed in the literature [14-16]. Also note that the full set of association structures that are accommodated in the **rstanarm** package's `stan_jm` modelling function are not described here but are discussed in the documentation for the `stan_jm` function itself.

2.4 Conditional independence assumption

A key assumption of the multivariate shared parameter joint model is that the observed longitudinal measurements are independent of one another (both across the M biomarkers and across the n_i time points), as well as independent of the event time, conditional on the individual-specific parameters $\boldsymbol{b}_i$. That is, we assume

$$\text{Cov}\Big(y_{im}(t), y_{im'}(t)|\boldsymbol{b}_i\Big) = 0 \tag{6}$$

$$\text{Cov}\Big(y_{im}(t), y_{im}(t')|\boldsymbol{b}_i\Big) = 0 \tag{7}$$

$$\text{Cov}\Big(y_{im}(t), T_i|\boldsymbol{b}_i\Big) = 0 \tag{8}$$

for some $m \neq m'$ and $t \neq t'$.

Although this may be considered a strong assumption, it is useful in that it allows the full likelihood for joint model to be factorised into the likelihoods for each of the component parts (i.e. the likelihoods for each of the submodels and the likelihood for the distribution of the individual-specific parameters).

2.5 Log posterior distribution

Under the conditional independence assumption, the log posterior for the i^{th} individual can be specified as

$$p(\boldsymbol{\theta}, \boldsymbol{b}_i|\boldsymbol{y}_i, T_i, d_i) \propto \log\left[\left(\prod_{m=1}^{M}\prod_{j=1}^{n_i} p(y_{ijm}|\boldsymbol{b}_i, \boldsymbol{\theta})\right)p(T_i, d_i|\boldsymbol{b}_i, \boldsymbol{\theta})p(\boldsymbol{b}_i|\boldsymbol{\theta})p(\boldsymbol{\theta})\right] \tag{9}$$

which we can rewrite as

$$p(\boldsymbol{\theta}, \boldsymbol{b}_i|\boldsymbol{y}_i, T_i, d_i) \propto \left(\sum_{m=1}^{M}\sum_{j=1}^{n_i} \log p(y_{ijm}|\boldsymbol{b}_i, \boldsymbol{\theta})\right) + \log p(T_i, d_i|\boldsymbol{b}_i, \boldsymbol{\theta}) + \log p(\boldsymbol{b}_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \tag{10}$$

where $\sum_{j=1}^{n_i} \log p(y_{ijm}|\boldsymbol{b}_i, \boldsymbol{\theta})$ is the log likelihood for the m^{th} longitudinal submodel, $\log p(T_i, d_i|\boldsymbol{b}_i, \boldsymbol{\theta})$ is the log likelihood for the event submodel, $\log p(\boldsymbol{b}_i|\boldsymbol{\theta})$ is the log likelihood for the distribution of the group-specific parameters (i.e. random effects), and $\log p(\boldsymbol{\theta})$ represents the log likelihood for the joint prior distribution across all remaining unknown parameters.[2]

We can rewrite the log likelihood for the event submodel as

$$\log p(T_i, d_i|\boldsymbol{b}_i, \boldsymbol{\theta}) = d_i * \log h_i(T_i) - \int_0^{T_i} h_i(s)ds \tag{11}$$

and then use Gauss-Kronrod quadrature with Q nodes to approximate $\int_0^{T_i} h_i(s)ds$, such that

$$\int_0^{T_i} h_i(s)ds \approx \frac{T_i}{2} \sum_{q=1}^{Q} w_q h_i \left(\frac{T_i(1+s_q)}{2} \right) \tag{12}$$

where w_q and s_q, respectively, are the standardised weights and locations ("abscissa") for quadrature node q ($q = 1, ..., Q$) [17]. In this notebook we choose to use $Q = 15$ quadrature nodes.[3]

Therefore, once we have an individual's event time T_i we can evaluate the design matrices for the event submodel and longitudinal submodels at the $Q+1$ necessary time points (which are the event time T_i and the quadrature points $\frac{T_i(1+s_q)}{2}$ for $q = 1, ..., Q$) and then pass these to Stan's data block. We can then evaluate the log likelihood for the event submodel by simply calculating the hazard $h_i(t)$ at those $Q+1$ time points and summing the quantities appropriately. This calculation will need to be performed each time we iterate through Stan's model block. The Stan code required to evaluate this log posterior is described in the next section.

3 Stan code

Here we describe the most important features of the Stan code used to estimate the joint model. The full Stan code is provided in the separate `jm.stan` file supplied with this notebook. To simplify things for the reader, we have limited ourselves to the situation in which $M = 2$ (i.e. we have two longitudinal biomarkers) and each of those longitudinal outcomes is modelled using a linear mixed model (i.e. identity link, normal distribution).

3.1 Data and transformed data blocks

The data block includes dimensions of the model, outcome vectors (e.g. observed biomarker values and event times), design matrices for the different submodels, and hyperparameters for the prior distributions. We do not discuss the data or transformed data blocks here in any detail.

[2]In this notebook we assume normal prior distributions for all unbounded parameters (e.g. regression coefficients) and half-normals for all lower-bounded parameters (e.g. standard deviations). However, in the **rstanarm** package there is a variety of prior distributions available to the user. For the prior distribution for the variance-covariance matrix of the group-specific parameters (i.e. the variance-covariance matrix for the individual-level random effects) we use a decomposition of the variance-covariance matrix into a vector of standard deviations and a correlation matrix. We then place a half-Cauchy prior distribution on each of the standard deviations, and use the LKJ correlation matrix distribution (parameterised in terms of it's Cholesky factor) as the prior for the correlation matrix of the group-specific parameters. Further details of this prior distribution are described in the Stan User Manual and the implementation can be seen in the `jm.stan` file included with this notebook. Importantly however, when using the `stan_jm` modelling function in the **rstanarm** package, the user can control the hyperparameters related to this prior distribution. Moreover, the user can instead choose to place a prior on a further decomposed version of the variance-covariance matrix, whereby the vector of standard deviations are further decomposed into a trace and a simplex vector. This latter option is taken directly from the prior distribution described for variance-covariance matrices in the **rstanarm** package's `stan_glmer` modelling, and we refer the reader to the documentation of that package for further details.

[3]The `stan_jm` modelling function in the **rstanarm** package allows the user to choose between $Q = 15$ (the default), 11, or 7 quadrature nodes.

3.2 Parameters block

Most of the parameters defined in the parameters block are "primitive" or "unscaled", meaning that they will be given a prior distribution with mean 0 and scale 1 and then converted into the actual parameters used in the regression equation via the transformed parameters block. Our parameters block therefore includes:

- `y1_gamma`, `y2_gamma`: the intercept for each of the longitudinal submodels. These intercept parameters are unbounded, given that each longitudinal submodel in our application consists of a linear mixed model (i.e. in our application we assume an identity link function and normal error distribution for each longitudinal biomarker).
- `y1_z_beta`, `y2_z_beta`: the primitive coefficients for each of the longitudinal submodels.
- `e_z_beta`, `a_z_beta`: the primitive coefficients and primitive association parameters for the event submodel.
- `y1_aux_unscaled`, `y2_aux_unscaled`: the unscaled standard deviations (SD) of the residual errors for each of the longitudinal submodels, combined into a single vector.
- `e_aux_unscaled`: the unscaled coefficients for the B-spline terms used in the baseline hazard.

The parameters block also includes the unscaled group-specific parameters (i.e. unscaled individual-level random effects). We specify these as a matrix, with the number of rows in the matrix equal to the total number of group-specific terms in the model, and the number of columns in the matrix equal to the total number of patients in the data (i.e. the total number of "groups"). We declare a parameter vector that contains the standard deviations for each of the group-specific parameters and a lower triangular matrix that corresponds to the Cholesky factor of the correlation matrix for the group-specific terms. The latter is declared using Stan's `cholesky_factor_corr` data type.

```
parameters {
  real y1_gamma; // intercepts in long. submodels
  real y2_gamma;
  vector[y_K[1]] y1_z_beta; // primitive coefs in long. submodels
  vector[y_K[2]] y2_z_beta;
  vector[e_K] e_z_beta; // primitive coefs in event submodel (log hazard ratios)
  vector[a_K] a_z_beta; // primitive assoc params (log hazard ratios)
  real<lower=0> y1_aux_unscaled; // unscaled residual error SDs
  real<lower=0> y2_aux_unscaled;
  vector[basehaz_df] e_aux_unscaled; // unscaled coefs for baseline hazard

  // group level params
  vector<lower=0>[b_K] b_sd; // group level sds
  matrix[b_K,b_N] z_b_mat;   // unscaled group level params
  cholesky_factor_corr[b_K > 1 ? b_K : 0]
    b_cholesky;              // cholesky factor of corr matrix
}
```

3.3 Transformed parameters block

The transformed parameters block includes code to alter the location and scale of the "primitive" or "unscaled" parameters, in order to obtain the actual parameters used in the regression submodels.

Note that in the code below `b_K` is the number of group-specific parameters in the model, so if `b_K > 1` then we will be estimating a correlation matrix for the group-specific parameters and, hence, we must transform the primitive group-specific parameters using `b_cholesky` and `b_sd`, rather than `b_sd` alone. If there was only one group-specific parameter in the model then there would be no correlation matrix (i.e. no b_cholesky parameter). Also note that for any *multivariate* joint model (i.e. more than one longitudinal outcome) we will have b_K > 1.

```
transformed parameters {
  ...
  // coefs for long. submodels
  y1_beta = y1_z_beta .* y1_prior_scale + y1_prior_mean;
  y2_beta = y2_z_beta .* y2_prior_scale + y2_prior_mean;

  // coefs for event submodel (incl. association parameters)
  e_beta = e_z_beta .* e_prior_scale + e_prior_mean;
  a_beta = a_z_beta .* a_prior_scale + a_prior_mean;

  // residual error SDs for long. submodels
  y1_aux = y1_aux_unscaled * y_prior_scale_for_aux[1] + y_prior_mean_for_aux[1];
  y2_aux = y2_aux_unscaled * y_prior_scale_for_aux[2] + y_prior_mean_for_aux[2];

  // b-spline coefs for baseline hazard
  e_aux = e_aux_unscaled .* e_prior_scale_for_aux + e_prior_mean_for_aux;

  // group level params
  if (b_K == 1)
    b_mat = (b_sd[1] * z_b_mat)';
  else if (b_K > 1)
    b_mat = (diag_pre_multiply(b_sd, b_cholesky) * z_b_mat)';
}
```

3.4 Model block

The model block consists of several distinct parts. We describe each of these separately.

In the first part of the model block, we evaluate the linear predictor for each of the M longitudinal submodels at the respective observation times. We then increment the target with the resulting likelihood. To evaluate the linear predictor we call a user-defined function which is defined in the functions {} block at the start of the jm.stan file. This function takes the form:

```
  /**
  * Evaluate the linear predictor for the glmer submodel
  *
  * @param X Design matrix for fe
  * @param Z Design matrix for re, for a single grouping factor
  * @param Z_id Group indexing for Z
  * @param gamma The intercept parameter
  * @param beta Vector of population level parameters
  * @param bMat Matrix of group level params
  * @param shift Number of columns in bMat
  *   that correpond to group level params from prior glmer submodels
  * @return A vector containing the linear predictor for the glmer submodel
  */
  vector evaluate_eta(matrix X, vector[] Z, int[] Z_id, real gamma,
                      vector beta, matrix bMat, int shift) {
    int N = rows(X);    // num rows in design matrix
    int K = rows(beta); // num predictors
    int p = size(Z);    // num group level params
    vector[N] eta;

    if (K > 0) eta = X * beta;
    else eta = rep_vector(0.0, N);

    for (k in 1:p)
      for (n in 1:N)
        eta[n] = eta[n] + (bMat[Z_id[n], k + shift]) * Z[k,n];

    return eta;
  }
```

Such that the code in our model block is the following:

```
model {
  //---- Log-lik for longitudinal submodels
  {
    // declare linear predictors
    vector[y_N[1]] y1_eta;
    vector[y_N[2]] y2_eta;

    // evaluate linear predictor for each long. submodel
    y1_eta = evaluate_eta(y1_X, y1_Z, y1_Z_id, y1_gamma, y1_beta, b_mat, 0);
    y2_eta = evaluate_eta(y2_X, y2_Z, y2_Z_id, y2_gamma, y2_beta, b_mat, b_KM[1]);

    // increment the target with the log-lik
    target += normal_lpdf(y1 | y1_eta, y1_aux);
    target += normal_lpdf(y2 | y2_eta, y2_aux);
  }
  ...
```

To evaluate the event submodel likelihood we must evaluate $h_i(T_i)$ for individuals who experienced the event (i.e. $d_i = 1$) (i.e. the hazard at their event time) as well as the cumulative hazard $\int_0^{T_i} h_i(s)ds$ for all individuals. Since we are going to evaluate the cumulative hazard using Gauss-Kronrod quadrature, this means calculating the hazard $h_i(t)$ at 15 quadrature points between 0 and T_i for each individual i. To do this, we have constructed the design matrices in R evaulated at the necessary times; these are passed to Stan's data block (not shown here) as `e_Xq`, `y1_Xq`, `y2_Xq` etc. In the code below there are several steps:

- In **Step 1** we use the event submodel design matrices to evaluate the $\boldsymbol{w}_i^T(t)\boldsymbol{\gamma}$ part of the event submodel's linear predictor at the observed event times and the 15 quadrature points between 0 and T_i.
- The remainder of the event submodel's linear predictor consists of the term corresponding to the association structure: $\sum_{m=1}^{M} \alpha_m \eta_{im}(t)$. This involves the current value of the longitudinal submodel's linear predictor, so we must also evaluate the longitudinal submodel's linear predictor at the event times and the 15 quadrature points between 0 and T_i. This is shown in **Step 2** of the code below.
- In **Step 3** we evaluate the log baseline hazard at the event times and the 15 quadrature points between 0 and T_i.
- In **Step 4** we combine the log baseline hazard with the event submodel linear predictor, that is, we evaluate

$$\log h_i(t) = \log h_0(t) + \left(\boldsymbol{w}_i^T(t)\boldsymbol{\gamma} + \sum_{m=1}^{M} \alpha_m \eta_{im}(t) \right)$$

- In **Steps 5** and **6** the hazard evaluated at the event times is separated out from the hazard evaluated at each of the quadrature points. The latter will be used in **Step 7** to evaluate the approximate cumulative hazard at the event time via the Gauss-Kronrod quadrature rule described in equation (12).

- In **Step 7** we evaluate the log likelihood for the event submodel as

$$\log p(T_i, d_i | \boldsymbol{b}_i, \boldsymbol{\theta}) = d_i * \log h_i(T_i) - \int_0^{T_i} h_i(s) ds$$

  The first term in **Step 7** is the log hazard contribution to the log likelihood for the event submodel. The second term is the log survival contribution to the log likelihood for the event submodel. The latter is obtained by summing over the quadrature points to get the approximate integral (i.e. cumulative hazard). Note that the `qwts` vector already incorporates the necessary scaling such that the integral is evaluated over limits $(0, T_i)$ rather than $(-1, +1)$. We increment the target with the resulting log likelihood.

```
//----- Log-lik for event submodel (Gauss-Kronrod quadrature)
{
  vector[nrow_y_Xq[1]] y1_eta_q;
  vector[nrow_y_Xq[2]] y2_eta_q;
  vector[nrow_e_Xq] e_eta_q;
  vector[nrow_e_Xq] log_basehaz;
  vector[nrow_e_Xq] ll_haz_q;
  vector[Nevents] log_haz_etimes;
  vector[Npat_times_qnodes] log_haz_qtimes;

  // Step 1: event submodel linear predictor at event time and quadrature points
  e_eta_q = e_Xq * e_beta;

  // Step 2: long. submodel linear predictor at event time and quadrature points
  y1_eta_q = evaluate_eta(y1_Xq, y1_Zq, y1_Zq_id, y1_gamma, y1_beta, b_mat, 0);
  y2_eta_q = evaluate_eta(y2_Xq, y2_Zq, y2_Zq_id, y2_gamma, y2_beta, b_mat, b_KM[1]);

  // Step 2 (continued): add on contribution from association structure to
  // the event submodel linear predictor at event time and quadrature points
  e_eta_q = e_eta_q + a_beta[1] * y1_eta_q + a_beta[2] * y2_eta_q;

  // Step 3: log baseline hazard at event time and quadrature points
  log_basehaz = basehaz_X * e_aux;

  // Step 4: log hazard at event time and quadrature points
  ll_haz_q = log_basehaz + e_eta_q;

  // Step 5: log hazard at event times only
  // (i.e. log hazard contribution to the likelihood)
  log_haz_etimes = head(log_haz_q, Nevents);

  // Step 6: log hazard at quadrature points only
  log_haz_qtimes = tail(log_haz_q, Npat_times_qnodes);

  // Step 7: log likelihood for event submodel
  target += sum(log_haz_etimes) - dot_product(qwts, exp(log_haz_qtimes));
}
```

We then increment the target with the log priors for each of the intercepts, coefficients, auxiliary parameters (including coefficients for the B-splines baseline hazard), and group-specific terms (i.e. individual-level random effects):

```
  //----- Log-priors

    // intercepts for long. submodels
    target += normal_lpdf(y1_gamma |
      y_prior_mean_for_intercept[1], y_prior_scale_for_intercept[1]);
    target += normal_lpdf(y2_gamma |
      y_prior_mean_for_intercept[2], y_prior_scale_for_intercept[2]);

    // coefficients for long. submodels
    target += normal_lpdf(y1_z_beta | 0, 1);
    target += normal_lpdf(y2_z_beta | 0, 1);

    // coefficients for event submodel
    target += normal_lpdf(e_z_beta | 0, 1);
    target += normal_lpdf(a_z_beta | 0, 1);

    // residual error SDs for long. submodels
    target += normal_lpdf(y1_aux_unscaled | 0, 1);
    target += normal_lpdf(y2_aux_unscaled | 0, 1);

    // b-spline coefs for baseline hazard
    target += normal_lpdf(e_aux_unscaled | 0, 1);

    // group level terms
      // sds
      target += student_t_lpdf(b_sd | b_prior_df, 0, b_prior_scale);
      // primitive coefs
      target += normal_lpdf(to_vector(z_b_mat) | 0, 1);
      // corr matrix
      if (b_K > 1)
        target += lkj_corr_cholesky_lpdf(b_cholesky | b_prior_regularization);
}
```

4 Application

4.1 Data

In order to make this notebook freely available we use a motivating example based on a publically accessible dataset. The Mayo Clinic's widely used primary biliary cirrhosis (PBC) data contains 312 individuals with primary biliary cirrhosis, who participated in a randomised placebo controlled trial of D-penicillamine conducted at the Mayo Clinic between 1974 and 1984 [18]. In our secondary analysis of this trial data, our primary research is *not* concerned with the efficacy of the randomised treatment but rather understanding how the clinical biomarker histories for these patients are associated with their overall survival. Specifically, we focus on the associations between two repeatedly measured clinical biomarkers, log serum bilirubin and serum albumin, and the risk of death. Given that the joint modelling methods are computationally intensive we restrict our analyses to a small random subset of just 40 patients from the PBC dataset. This ensures that the computation time for the joint models described in later sections are kept to a minimum and therefore this notebook can be compiled in a relatively short time. However, this also means that the clinical findings from this analysis should not to be overinterpreted. Rather, this notebook aims to simply demonstrate the joint modelling framework and describe how these models can be estimated using Stan.

The PBC data are contained in two separate data frames, each saved as an RDS object. The first data frame (saved as "Data/pbcLong.rds"), contains multiple-row per patient longitudinal biomarker information, as shown in

```
head(pbcLong)
```

```
##   id      age sex trt      year     logBili albumin platelet
## 1  1 58.76523   f   1 0.0000000  2.67414865    2.60      190
## 2  1 58.76523   f   1 0.5256674  3.05870707    2.94      183
## 3  2 56.44627   f   1 0.0000000  0.09531018    4.14      221
## 4  2 56.44627   f   1 0.4982888 -0.22314355    3.60      188
## 5  2 56.44627   f   1 0.9993155  0.00000000    3.55      161
## 6  2 56.44627   f   1 2.1026694  0.64185389    3.92      122
```

while the second data frame (saved as "Data/pbcSurv.rds"), contains single-row per patient survival information, as shown in

```
head(pbcSurv)
```

```
##    id      age sex trt futimeYears status death
## 1   1 58.76523   f   1    1.095140      2     1
## 3   2 56.44627   f   1   14.151951      0     0
## 12  3 70.07255   m   1    2.770705      2     1
## 16  4 54.74059   f   1    5.270363      2     1
## 23  5 38.10541   f   0    4.120465      1     0
## 29  6 66.25873   f   0    6.852841      2     1
```

The variables included across the two datasets can be defined as follows:

- `age` in years
- `albumin` serum albumin (g/dl)
- `logBili` logarithm of serum bilirubin
- `death` indicator of death at endpoint
- `futimeYears` time (in years) between baseline and the earliest of death, transplantion or censoring
- `id` numeric ID unique to each individual
- `platelet` platelet count
- `sex` gender (m = male, f = female)
- `status` status at endpoint (0 = censored, 1 = transplant, 2 = dead)
- `trt` binary treatment code (0 = placebo, 1 = D-penicillamine)
- `year` time (in years) of the longitudinal measurements, taken as time since baseline)

4.2 Estimation using the simplified jm.stan file

We fit a multivariate joint model to the two longitudinal biomarkers, log serum bilirubin and serum albumin, and time-to-death. Note that patients are censored if they had a transplant prior to death (here we ignore the fact that this is likely to be informative censoring). We fit a linear mixed model (identity link, normal distribution) for each biomarker with a patient-specific intercept and linear slope but no other covariates. In the event submodel we include gender (`sex`) and treatment (`trt`) as baseline covariates. Each biomarker is assumed to be associated with the log hazard of death at time t via it's expected value at time t (i.e. a *current value* association structure).

To save needing to carry out any data manipulation steps we instead used the `stan_jm` modelling function in **rstanarm** to generate the R list for passing to **rstan**. This data is saved as an RDS object and supplied with the notebook ("Stan/standata.rds"). In addition, a function to generate a list of initial values has also been supplied as an RDS object with the notebook ("Stan/staninit.rds"). Of course, the stan file containing the model is also supplied ("Stan/jm.stan"). We can therefore estimate this model using the **rstan** package:

```
standata <- readRDS("Stan/standata.rds")
staninit <- readRDS("Stan/staninit.rds")
mod1 <- with_filecache(
  stan(
    file = "Stan/jm.stan",
    data = standata,
    init = function() staninit,
    chains = 2, seed = 12345),
  filename = "mod1.rds")
```

Since our primary interest is in the association between the current value of each of the biomarkers (log serum bilirubin and serum albumin) and the hazard of death, we focus on the estimated association parameters. The summary of the posterior distribution for each of the association parameters follows:

```
print(mod1, pars = "a_beta")
```

```
## Inference for Stan model: jm.
## 2 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=2000.
##
##             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## a_beta[1]  0.72    0.01 0.29  0.17  0.53  0.72  0.92  1.29  2000    1
## a_beta[2] -3.23    0.03 0.75 -4.77 -3.66 -3.17 -2.72 -1.90   752    1
##
## Samples were drawn using NUTS(diag_e) at Wed Nov 29 18:17:42 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

We see that a one unit increase in log serum bilirubin is associated with a 0.72 (95% CrI: 0.17 to 1.29) unit increase in the log hazard of death, equivalent to a 2.05-fold (95% CrI: 1.19 to 3.63) *increase* in the hazard of death. Similarly, a one unit increase in serum albumin is associated with a 3.23 (95% CrI: 1.90 to 4.77) unit *decrease* in the log hazard of death. These estimates are broadly in line with what we would expect from a clinical perspective; that is, that higher serum bilirubin is associated with *worse* patient outcomes (i.e. higher risk of mortality), whilst higher serum albumin is associated with *better* patient outcomes (i.e. lower risk of mortality). However, recall that we have estimated this model with a very small dataset only used for demonstration purposes. Moreover, the number of mortality events ($N = 29$) is even less than the number of patients since some patients are censored.

4.3 Estimation using the joint modelling functionality in rstanarm

The `jm.stan` file provided with this notebook is a simplified version of the Stan code underlying the `stan_jm` modelling function in the **rstanarm** package. However, estimating the model using the **rstanarm** provides us with much nicer output (for example, meaningful variable names!) as well as a broad range of post-estimation functionality, including model diagnostics, posterior predictions, dynamic predictions and more.

To see this, we can use the development version of **rstanarm** with joint modelling functionality to refit our model, this time using `stan_jm` with the customary R formula syntax and data frames:

```
mod2 <- with_filecache(
  stan_jm(
    formulaLong = list(
        logBili ~ year + (year | id),
        albumin ~ year + (year | id)),
    formulaEvent = survival::Surv(futimeYears, death) ~ sex + trt,
    dataLong = pbcLong, dataEvent = pbcSurv,
    time_var = "year", assoc = "etavalue", basehaz = "bs",
    chains = 2, seed = 12345),
  filename = "mod2.rds")
```

We can now examine the output from the fitted model, for example

```
print(mod2)
```

```
## stan_jm
##  formula (Long1): logBili ~ year + (year | id)
##  family  (Long1): gaussian [identity]
##  formula (Long2): albumin ~ year + (year | id)
##  family  (Long2): gaussian [identity]
##  formula (Event): survival::Surv(futimeYears, death) ~ sex + trt
##  baseline hazard: bs
##  assoc:           etavalue (Long1), etavalue (Long2)
## ------
##
## Longitudinal submodel 1: logBili
##             Median MAD_SD
## (Intercept) 0.669  0.180
## year        0.227  0.044
## sigma       0.354  0.017
##
## Longitudinal submodel 2: albumin
##             Median MAD_SD
## (Intercept)  3.518  0.085
## year        -0.160  0.025
## sigma        0.290  0.013
##
## Event submodel:
##                 Median  MAD_SD  exp(Median)
## (Intercept)       6.751   2.887 855.218
## sexf             -0.148   0.680   0.863
## trt              -0.503   0.484   0.605
## Long1|etavalue    0.798   0.295   2.221
## Long2|etavalue   -3.065   0.899   0.047
## b-splines-coef1  -0.889   1.070      NA
## b-splines-coef2   0.547   0.925      NA
## b-splines-coef3  -2.623   1.240      NA
## b-splines-coef4  -0.441   1.751      NA
## b-splines-coef5  -1.226   1.777      NA
## b-splines-coef6  -2.576   1.906      NA
##
## Group-level error terms:
##  Groups Name              Std.Dev. Corr
##  id     Long1|(Intercept) 1.2605
##         Long1|year        0.1939    0.51
##         Long2|(Intercept) 0.5044   -0.64 -0.52
##         Long2|year        0.1028   -0.60 -0.82  0.47
## Num. levels: id 40
##
## Sample avg. posterior predictive distribution
## of longitudinal outcomes:
##                Median MAD_SD
## Long1|mean_PPD 0.588  0.030
## Long2|mean_PPD 3.343  0.023
##
## ------
## For info on the priors used see help('prior_summary.stanreg').
```

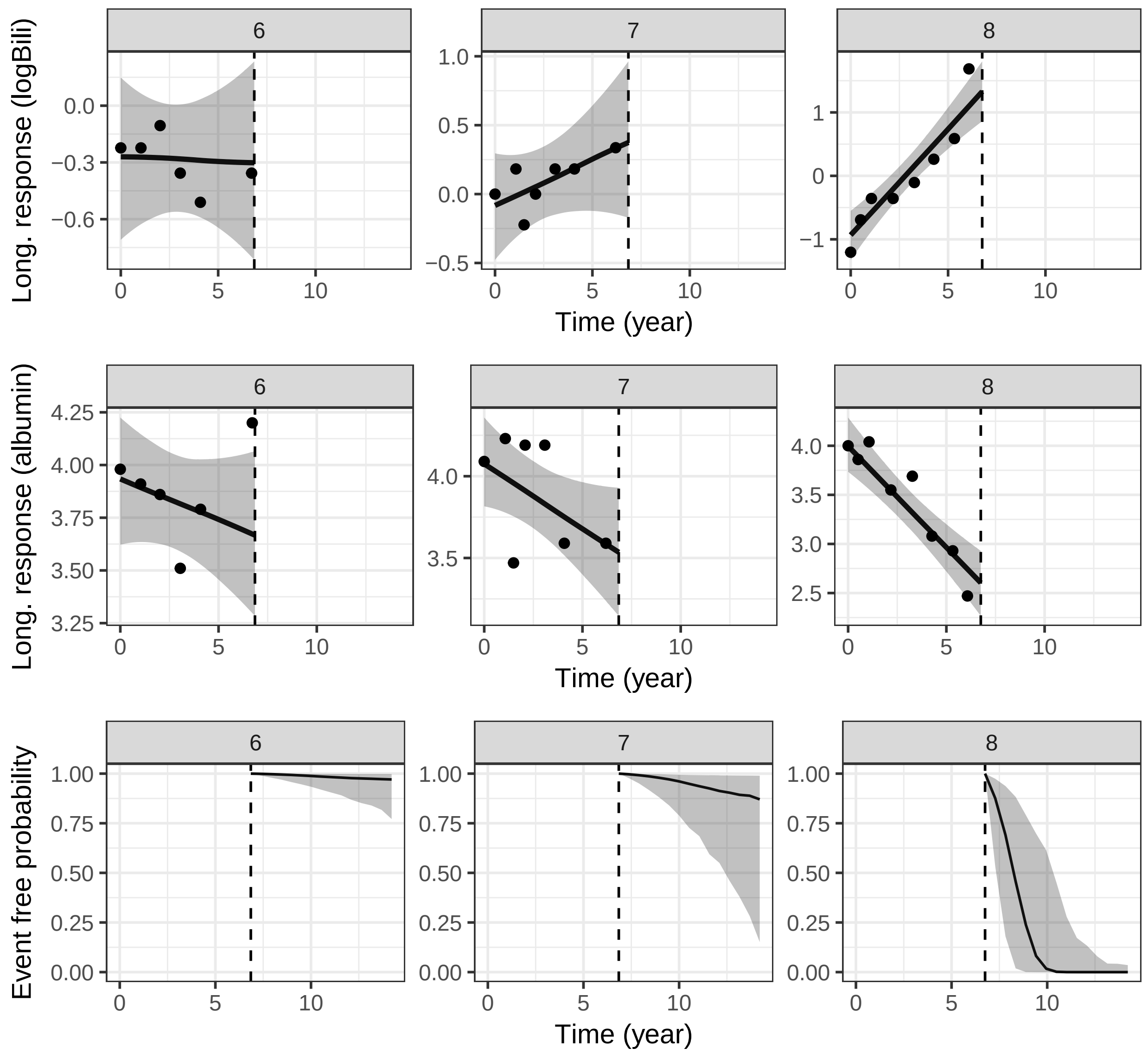
or we can examine the summary output for the association parameters alone:

```r
summary(mod2, pars = "assoc")
```

```
##
## Model Info:
##
##  function:        stan_jm
##  formula (Long1): logBili ~ year + (year | id)
##  family  (Long1): gaussian [identity]
##  formula (Long2): albumin ~ year + (year | id)
##  family  (Long2): gaussian [identity]
##  formula (Event): survival::Surv(futimeYears, death) ~ sex + trt
##  baseline hazard: bs
##  assoc:           etavalue (Long1), etavalue (Long2)
##  algorithm:       sampling
##  priors:          see help('prior_summary')
##  sample:          2000 (posterior sample size)
##  num obs:         304 (Long1), 304 (Long2)
##  num subjects:    40
##  num events:      29 (72.5%)
##  groups:          id (40)
##  runtime:         1.1 mins
##
## Estimates:
##                         mean   sd     2.5%   25%    50%    75%    97.5%
## Assoc|Long1|etavalue  0.801  0.301  0.214  0.595  0.798  0.992  1.416
## Assoc|Long2|etavalue -3.128  0.898 -5.114 -3.709 -3.065 -2.505 -1.515
##
## Diagnostics:
##                       mcse  Rhat  n_eff
## Assoc|Long1|etavalue 0.007 1.004 2000
## Assoc|Long2|etavalue 0.026 1.003 1238
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

We can see that the estimated association parameters are similar to those obtained from the model in the previous section. However, we can now also access a range of post-estimation functions (described in the `stan_jm` and related help documentation; see for example `help(posterior_traj)` or `help(posterior_survfit)`). As an example, let's plot the predicted trajectories for each biomarker and the predicted survival function for three selected individuals in the dataset using `stan_jm` post-estimation functions:

```
p1 <- posterior_traj(mod2, m = 1, ids = 6:8)
p2 <- posterior_traj(mod2, m = 2, ids = 6:8)
p3 <- posterior_survfit(mod2, ids = 6:8, draws = 200)
pp1 <- plot(p1, plot_observed = TRUE, vline = TRUE)
pp2 <- plot(p2, plot_observed = TRUE, vline = TRUE)
plot_stack_jm(yplot = list(pp1, pp2), survplot = plot(p3))
```



Here we can see the strong relationship between the underlying values of the biomarkers and mortality. Patient 8 who, relative to patients 6 and 7, has a higher underlying value for log serum bilirubin and a lower underlying value for serum albumin at the end of their follow up has a far worse predicted probability of survival.

5 Discussion

In this notebook we have introduced the formulation of a shared parameter joint model for longitudinal and time-to-event data. The formulation of the joint model can allow for multiple longitudinal biomarkers along with a terminating event. The association between the longitudinal and event processes can be parameterised in a variety of ways, but here we have focussed on the so-called *current value* association structure which serves as the simplest and natural starting point.

The aim of this notebook was to introduce some of the ideas underpinning the estimation of these joint models in Stan. One key feature of the Stan code that we have tried to describe in detail is the implementation of the Gauss-Kronrod quadrature rule. The Gauss-Kronrod quadrature rule is required to approximate the cumulative hazard in the likelihood of the event submodel. This aspect makes evaluating the log likelihood for the event submodel more computationally intensive than if there were a closed-form solution to the integral. In addition, the models are computationally intensive due to the relatively large number of group-specific parameters that often need to be estimated. Nonetheless, estimating joint models under a Bayesian framework can provide a number of benefits. The specification of complex association structures can be made much easier. Furthermore, a Bayesian approach can lead more naturally to dynamic predictions. For these, and other reasons, we believe it is of interest to try and optimise the estimation of these models in Stan. The hope is that by describing the Stan code in some detail as part of this notebook, those reading it will have the opportunity to provide guidance on how increases in speed, efficiency, or numerical stability might be achieved.

6 Acknowledgements

Much of the joint modelling functionality that has been contributed to the **rstanarm** package has been built upon code that was already included in that package, and that code was written primarily by Ben Goodrich and Jonah Gabry [12]. We are also grateful to them for their ongoing support in helping to get the joint modelling functionality up and running in **rstanarm**.

7 References

1. Henderson R, Diggle P, Dobson A. Joint modelling of longitudinal measurements and event time data. *Biostatistics* 2000;**1**(4):465-80.
2. Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics* 1997;**53**(1):330-9.
3. Tsiatis AA, Davidian M. Joint modeling of longitudinal and time-to-event data: An overview. *Stat Sinica* 2004;**14**(3):809-34.
4. Gould AL, Boye ME, Crowther MJ, Ibrahim JG, Quartey G, Micallef S, et al. Joint modeling of survival and longitudinal non-survival data: current methods and issues. Report of the DIA Bayesian joint modeling working group. *Stat Med.* 2015;**34**(14):2181-95.
5. Rizopoulos D. *Joint Models for Longitudinal and Time-to-Event Data: With Applications in R* CRC Press; 2012.
6. Liu G, Gould AL. Comparison of alternative strategies for analysis of longitudinal trials with dropouts. *J Biopharm Stat* 2002;**12**(2):207-26.
7. Prentice RL. Covariate Measurement Errors and Parameter-Estimation in a Failure Time Regression-Model. *Biometrika* 1982;**69**(2):331-42.
8. Baraldi AN, Enders CK. An introduction to modern missing data analyses. *J Sch Psychol* 2010;**48**(1):5-37.
9. Philipson PM, Ho WK, Henderson R. Comparative review of methods for handling drop-out in longitudinal studies. *Stat Med* 2008;**27**(30):6276-98.

10. Pantazis N, Touloumi G. Bivariate modelling of longitudinal measurements of two human immunodeficiency type 1 disease progression markers in the presence of informative drop-outs. *Applied Statistics* 2005;**54**:405-23.
11. Taylor JM, Park Y, Ankerst DP, et al. Real-time individual predictions of prostate cancer recurrence using joint models. *Biometrics* 2013;**69**(1):206-13.
12. Stan Development Team. *rstanarm: Bayesian applied regression modeling via Stan.* R package version 2.14.1. http://mc-stan.org/. 2016.
13. R Core Team. *R: A language and environment for statistical computing.* Vienna, Austria: R Foundation for Statistical Computing; 2015.
14. Crowther MJ, Lambert PC, Abrams KR. Adjusting for measurement error in baseline prognostic biomarkers included in a time-to-event analysis: a joint modelling approach. *BMC Med Res Methodol* 2013;**13**.
15. Hickey GL, Philipson P, Jorgensen A, Kolamunnage-Dona R. Joint modelling of time-to-event and multivariate longitudinal outcomes: recent developments and issues. *BMC Med Res Methodol* 2016;**16**(1):117.
16. Rizopoulos D, Ghosh P. A Bayesian semiparametric multivariate joint model for multiple longitudinal outcomes and a time-to-event. *Stat Med.* 2011;**30**(12):1366-80.
17. Laurie DP. Calculation of Gauss-Kronrod quadrature rules. *Math Comput* 1997;**66**(219):1133-45.
18. Therneau T, Grambsch P. *Modeling Survival Data: Extending the Cox Model* Springer-Verlag, New York; 2000. ISBN: 0-387-98784-3