Super Learning in Joint Models

Third report: Super learning in multivariate joint models for longitudinal and time-to-event data for optimizing dynamic predictions. A comparison with functional principal component analysis-based methods.

Arnau Garcia Fernandez

Facultat de Matemàtiques i Estadística
Universitat Politècnica de Catalunya
Date: 19/Dec/2024
Director: Dimitris Rizopoulos
Co-Director: Guadalupe Gómez

Contents

1 Introduction

In this report we focus on the next specific framework: we have data with time to event data, as well as longitudinal data. In addition, we have more than one longitudinal variable. In particular, let us assume that we have L longitudinal biomarkers. Our main goal is to compute subject-specific dynamic predictions for survival outcomes. Under the Bayesian paradigm, which is the one we use in this work, the computation of that dynamic prediction is derived by means of the posterior distribution of the joint model. Nonetheless, this joint model is a multivariate joint model with L longitudinal outcomes. In practice, fitting a multivariate joint model is computationally expensive. Taking into account that our main objective is to compute a predictive metric, we wonder if we can use super learning (SL) [7] and [12], to calculate this prediction with good enough accuracy, and avoid fitting a multivariate joint model.

From now on, we will be using the notation in [10], adding the detail (also treated in the previous reference) that we have L longitudinal biomarkers. Assume we have the data $\mathcal{D}_n = \{T_i, \delta_i, \boldsymbol{y}_{li}; i = 1, \ldots, n, l = 1, \ldots, L\}$. Where T_i^* denotes true time to true event of interest ε for the i-th subject, C_i the censoring time, $T_i = min(T_i^*, C_i)$ is the corresponding observed time to the event, and $\delta_i = \mathbf{1}(T_i^* \leq C_i)$ is the event indicator. Moreover, $\boldsymbol{y}_{li}$ is the $n_{li} \times 1$ longitudinal response vector for the i-th subject and the l-th longitudinal response, with element $y_{li}(t_{ij})$ being the value of the longitudinal outcome taken at time point t_{ij}, $j = 1, \ldots, n_{li}$ (n_{li} is the number of measurements for the i-th subject, l-th longitudinal response). We assume that the response vector $\boldsymbol{y}_i$ conditional on the random effects $\boldsymbol{b}_i$ has distribution $\mathcal{F}_\Psi$ within the exponential family of distributions. Thus, we are assuming a less general framework than the one postulated in [10], this is because we are interested in write the likelihood of the model, which is easier taking the assumption that the longitudinal outcomes are generalized linear mixed models, (GLMM's). The mean of the distribution of the longitudinal outcomes conditional on the random effects has the form

$$g_l\left[E(y_{li}(t)|\boldsymbol{b}_{li})\right] = m_{li}(t) = \boldsymbol{x}_{li}^\top(t)\boldsymbol{\beta}_l + \boldsymbol{z}_{li}^\top(t)\boldsymbol{b}_{li}, \quad l = 1, \ldots, L, \tag{1}$$

where $g_l(\cdot)$ denotes a known one-to-one monotonic link function for the lth longitudinal outcome, and $y_{li}(t)$ denotes the value of the lth longitudinal outcome for the ith subject at time point t, $\boldsymbol{x}_{li}(t)$ and $\boldsymbol{z}_{li}(t)$ denote the time-dependent design vectors for the fixed-effects $\boldsymbol{\beta}$ and for the random effects $\boldsymbol{b}_{li}$, respectively. We let ϕ denote the scale parameter of $\mathcal{F}_\Psi$, so $\boldsymbol{\Psi} = (\boldsymbol{\beta}, \phi)$. Notice the dependence of the previous components on the longitudinal outcome, we are assuming that we have L outcomes. To account for the association between the multiple longitudinal outcomes, we link their corresponding random effects. More specifically, the complete vector of random effects $\boldsymbol{b}_i = (\boldsymbol{b}_{1i}^\top, \boldsymbol{b}_{2i}^\top, \ldots, \boldsymbol{b}_{Li}^\top)^\top$ is assumed to follow a multivariate normal distribution with mean zero and variance-covariance matrix $\boldsymbol{D}$.

For the survival process, we assume that the risk of an event depends on a function of the subject-specific linear predictor $m_{li}(y)$ and the random effects. More specifically, we have

$$h_i\left(t \mid \mathcal{Y}_i(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\gamma^\top \boldsymbol{w}_i + \sum_{l=1}^{L} f_l(t, \mathcal{Y}_{li}(t), \boldsymbol{b}_{li}, \boldsymbol{\alpha}_l)\right\}, \quad t > 0, \tag{2}$$

where $\mathcal{Y}_{li}(t) = \{m_{li}(s), 0 \leq s < t\}$ denotes the history of the underlying lth longitudinal process up to t, $h_o(\cdot)$ denotes the baseline hazard function, $\boldsymbol{w}_i$ is a vector of baseline covariates with corresponding regression coefficients $\boldsymbol{\gamma}$ and $\boldsymbol{\alpha}_l$ is the association parameter (quantifies the effect of the underlying longitudinal outcome to the risk for an event). The reader can observe that the expression above is similar to the typical one for joint models with just one longitudinal outcome, but expanded to accommodate for all longitudinal outcomes. Since we are working under the Bayesian approach, we should complete the specification of the joint model and choose how to model the baseline hazard function. We use a B-splines approach,

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

where $B_q(t, \boldsymbol{v})$ denotes the q-th basis function of a B-spline with knots $v_1, \ldots, v_Q$ and γ_{h_0} the vector of spline coefficients. Q is the number of knots we use.

Since we are working under the Bayesian paradigm, we have to specify the prior distribution for all the parameters involved in our model. Let

$$\boldsymbol{\theta} = \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_L, \phi_1, \ldots, \phi_L, \gamma_{h_0}, \gamma, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L, vech(\boldsymbol{D}), \tau\} \tag{4}$$

denote the vector of all model parameters, where $vech(\boldsymbol{D})$ denotes the unique elements of the variance-covariance matrix $\boldsymbol{D}$. Inferences are drawn using the joint posterior distribution $\{\boldsymbol{\theta}, \boldsymbol{b} | \boldsymbol{Y}, \boldsymbol{T}, \delta\}$. Standard priors will be used for $\boldsymbol{\theta}$, normal priors for all the regression coefficients, and inverse-gamma priors for $\phi_1, \ldots, \phi_L$ and the diagonal elements of $\boldsymbol{D}_1, \ldots, \boldsymbol{D}_L$, and LKJ prior for the correlation matrices of the random effects. To ensure smoothness of the baseline hazard function $h_0(t)$ we use a penalized prior distribution for the regression coefficients γ_{h_0},

$$p(\gamma_{h_0} | \tau) \propto \tau^{\rho(K)/2} \exp\big(-\frac{\tau}{2} \gamma_{h_0}^\top \boldsymbol{K} \gamma_{h_0}\big),$$

where τ is the smoothing parameter that takes a Gamma$(5, 0.05)$ hyper-prior to ensure a proper posterior for γ_{h_0}, $\boldsymbol{K} = \Delta_r^\top \Delta_r$, Δ_r denoting the rth difference penalty matrix, and $\rho(\boldsymbol{K})$ denotes the rank of $\boldsymbol{K}$. Observe that the smoothing paramter τ is also included in the parameters of the model $\boldsymbol{\theta}$. Markov Chain Monte Carlo (MCMC) is the method used to obtain samples from the posterior distribution for all model parameters and the random effects.

Having introduced the problem in a proper manner, we should remember that our main goal is to compute the following dynamic individualized prediction

$$\begin{aligned}
\pi_j(u \mid t) &= \int \mathrm{P}\left(T_j^* \geq u \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta}\right) p\left(\boldsymbol{\theta} \mid \mathcal{D}_n\right) d\boldsymbol{\theta} \\
&= \int \left(\int \mathrm{P}\left(T_j^* \geq u \mid T_j^* > t, \boldsymbol{b}_j, \boldsymbol{\theta}\right) p\left(\boldsymbol{b}_j \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta}\right) d\boldsymbol{b}_j \right) p\left(\boldsymbol{\theta} \mid \mathcal{D}_n\right) d\boldsymbol{\theta} \\
&= \int \int \frac{S_j\left\{u \mid \mathcal{Y}_j\left(u, \boldsymbol{b}_j\right), \boldsymbol{\theta}\right\}}{S_j\left\{t \mid \mathcal{Y}_j\left(t, \boldsymbol{b}_j\right), \boldsymbol{\theta}\right\}} p\left(\boldsymbol{b}_j \mid T_j^* > t, \mathcal{Y}_j(t), \boldsymbol{\theta}\right) p\left(\boldsymbol{\theta} \mid \mathcal{D}_n\right) d\boldsymbol{\theta} d\boldsymbol{b}_j.
\end{aligned} \tag{5}$$

Clearly, the quantity above is computed using the posterior distribution of the model. Due to the fact that this posterior is coming from the multivariate joint model (2), where all the L longitudinal outcomes are used, and that this model is not easy to fit, we are interested in another way of calculating this metric. Super Learning (SL) [12] is the chosen method to compute the prediction avoiding to fit the multivariate joint model.

Let us assume that we have the library $\mathcal{L} = \{M_1, \dots, M_L\}$ consisting of the following L univariare joint models:

$$\begin{aligned}
M1 : \quad & h_i\left(t \mid \mathcal{Y}_{1i}(t), \boldsymbol{w}_i\right) = h_{0,1}(t) \exp\left\{\boldsymbol{\gamma_1}^\top \boldsymbol{w}_i + f_1(t, \mathcal{Y}_{1i}(t), \boldsymbol{b}_{1i}, \alpha_1)\right\}, \\
M2 : \quad & h_i\left(t \mid \mathcal{Y}_{2i}(t), \boldsymbol{w}_i\right) = h_{0,2}(t) \exp\left\{\boldsymbol{\gamma_2}^\top \boldsymbol{w}_i + f_2(t, \mathcal{Y}_{2i}(t), \boldsymbol{b}_{2i}, \alpha_2)\right\}, \\
\dots \quad & \\
M_L : \quad & h_i\left(t \mid \mathcal{Y}_{Li}(t), \boldsymbol{w}_i\right) = h_{0,L}(t) \exp\left\{\boldsymbol{\gamma_L}^\top \boldsymbol{w}_i + f_L(t, \mathcal{Y}_{Li}(t), \boldsymbol{b}_{Li}, \alpha_L)\right\}.
\end{aligned} \tag{6}$$

As the reader can observe, the library of models above is built by decomposing the multivariate model (2) into L univariate joint model, each one with one of the L longitudinal outcomes. Another important observation is that we assume fixed $\boldsymbol{w}_i$ in all the L models. Moreover, although we have different regression coefficients per univariate model $(\gamma_1, \dots, \gamma_L)$, we assume exactly the same prior distribution for all these parameters, which is the same assumed for the regression coefficient γ in the multivariate joint model (2). Thus, we have

$$p(\gamma) = p(\gamma_1) = p(\gamma_2) = \cdots = p(\gamma_L). \tag{7}$$

Furthermore, since we are dealing with L univariate joint models separately, we have L different vectors of random effects accounting for each model. These random effects are assumed to follow a multivariate normal distribution with mean zero and variance-covariance matrix D_l, for $l = 1, \dots, L$. In the baseline hazard functions in (6) we have added a subindex indicating the corresponding univariate joint model, because regression coefficients used in the B-spline approach are also different for each model.

The main goal of this work is to see if applying the SL procedure to the library $\mathcal{L}$ one can obtain a good enough prediction to be used instead of the multivariate model. With this objective in mind, first of all, we will try to establish a relationship between the posterior of the multivariate joint model (2) and the posteriors of the univariate models (6).

2 On the relationship between the posteriors

In this section we will simplify some parts of our problem in order to write as detailed as possible the posteriors of the models. When working under the Bayesian framework, the common used strategy to establish relationships between posteriors is by comparing them. It is well-known that the posterior distribution is the compromise between the likelihood and the prior distribution. Let us assume prior independence, this is, the prior distribution is the product of the corresponding marginal prior distributions. It is,

$$p(\boldsymbol{\theta}) = p(\boldsymbol{\beta}_1)\cdots p(\boldsymbol{\beta}_L)p(\phi_1)\cdots p(\phi_L)p(\boldsymbol{\alpha}_1)\cdots p(\boldsymbol{\alpha}_L)p(vech(\boldsymbol{D}))p(\tau)p(\gamma_{h_0})p(\gamma). \tag{8}$$

To specify the likelihood of the multivariate model (2), some assumptions have to be made. Besides the conditional independence between the longitudinal and the event outcomes, and the repeated measurements of the longitudinal process, on the random effects $\boldsymbol{b}_i$, when working with more than one longitudinal outcome we should add the conditional independence of the L longitudinal outcomes based, again, on the random effects [8]. Then, we assume that

$$\begin{aligned} p\left(\boldsymbol{y}_i, T_i, \delta_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}\right) &= p\left(\boldsymbol{y}_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}\right) p\left(T_i, \delta_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}\right) \\ p\left(\boldsymbol{y}_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}\right) &= \prod_l p\left(y_{li} \mid \boldsymbol{b}_{li}, \boldsymbol{\theta}_l\right), \\ p(\boldsymbol{y}_{li} | \boldsymbol{b}_{li}, \boldsymbol{\theta}_l) &= \prod_j p\left(y_{li}(t_{ij}) | \boldsymbol{b}_{li}, \boldsymbol{\theta}_l\right), \end{aligned} \tag{9}$$

where $\boldsymbol{y}_i = (\boldsymbol{y}_{1i}, \ldots, \boldsymbol{y}_{li}), \boldsymbol{y}_{li} = (y_{li}(t_{i1}), \ldots, y_{li}(t_{in_{li}}))$ and $\boldsymbol{b}_i = (\boldsymbol{b}_{1i}, \ldots, \boldsymbol{b}_{li})$.

Let us also assume that we have right-censored data and non-informative censoring. In addition, to simplify the algebra, we assume that the functional forms we are using in the joint models are the simplest ones: $f_l(t, \mathcal{Y}_{li}(t), \boldsymbol{b}_{li}) = \boldsymbol{\alpha}_l m_{li}(t)$. Now, using that the survival outcome contribution in the likelihood under right-censoring is

$$L = \prod_i h(t_i^*)^{\delta_i} S(t_i^*).$$

Assuming all of the above, we have that the likelihood of the multivariate joint model (2) is

$$\begin{aligned}
&p\left(\boldsymbol{y}_i, T_i, \delta_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}\right) = p\left(\boldsymbol{y}_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}\right) p\left(T_i, \delta_i \mid \boldsymbol{b}_i, \boldsymbol{\theta}\right)\\
&= \Big[\prod_{l=1}^{L}\prod_{j=1}^{n_{li}} p\left(y_{li}(t_{ij}) | \boldsymbol{b}_{li}, \boldsymbol{\theta}\right)\Big] p(\boldsymbol{b}_i|\boldsymbol{\theta}) p(T_i, \delta_i|\boldsymbol{b}_i, \boldsymbol{\theta})\\
&\propto \exp\left\{\sum_{l=1}^{L}\left(\frac{\sum_{j=1}^{n_{li}}(y_{li}(t_{ij})m_{li}(t_{ij}) - A(m_{li}(t_{ij})))}{\phi_l}\right) + \sum_l\sum_j C(y_{li}(t_{ij}), \phi_l)\right\}\\
&\quad\times \det(\boldsymbol{D})^{-1/2}\exp\left(-\boldsymbol{b}_i^\top \boldsymbol{D}^{-1}\boldsymbol{b}_i/2\right)\\
&\quad\times\left[\exp\left\{\sum_q \gamma_{h_0,q}B_q\left(T_i, \boldsymbol{v}\right) + \boldsymbol{\gamma}^\top \boldsymbol{w}_i + \sum_{l=1}^{L}\alpha_l m_{li}\left(T_i\right)\right\}\right]^{\delta_i}\\
&\quad\times\exp\left[-\exp\left(\boldsymbol{\gamma}^\top\boldsymbol{w}_i\right)\int_0^{T_i}\exp\left\{\sum_q \gamma_{h_0,q}B_q(s, \boldsymbol{v}) + \sum_{l=1}^{L}\alpha_l m_{li}(s)\right\}ds\right].
\end{aligned} \tag{10}$$

The reader can observe the following components in the likelihood above: in the first line after the proportionality symbol, we have the contribution of the longitudinal outcomes (or outcome). Having assumed that we model the longitudinal processes as GLMM's, one can use the general expression of the exponential family of distributions to write down the contribution of the longitudinal outcomes to the likelihood. Following, in the second line we have the contribution of the random effects. Note that in the multivariate joint model, the random effects act jointly. Then, in the third and forth line the reader can observe the contribution of the event outcomes.

Now, let us write the likelihood for the lth univariant model in (6). In this case, since there is only one longitudinal outcome, we only have to take into account the conditional independence between the longitudinal and the event outcomes, and the repeated measurements of the longitudinal process, both based on the random effects.

$$\begin{aligned}
&p\left(\boldsymbol{y}_{li}, T_i, \delta_i \mid \boldsymbol{\theta}, \boldsymbol{b}_{li}\right) = \Big[\prod_{j=1}^{n_{li}} p\left(y_{li}(t_{ij}) | \boldsymbol{b}_{li}, \boldsymbol{\theta}\right)\Big] p(\boldsymbol{b}_{li}|\boldsymbol{\theta}) p(T_i, \delta_i|\boldsymbol{b}_{li}, \boldsymbol{\theta})\\
&\propto \exp\left\{\frac{\sum_{j=1}^{n_{li}}(y_{li}(t_{ij})m_{li}(t_{ij}) - A(m_{li}(t_{ij})))}{\phi_l} + \sum_{j=1}^{n_{li}} C(y_{li}(t_{ij}), \phi_l)\right\}\\
&\times \det(\boldsymbol{D}_l)^{-1/2}\exp\left(-\boldsymbol{b}_{li}^\top \boldsymbol{D}_l^{-1}\boldsymbol{b}_{li}/2\right)\\
&\times\left[\exp\left\{\sum_q \gamma_{h_0,q,l}B_{q,l}\left(T_i, \boldsymbol{v}\right) + \boldsymbol{\gamma_l}^\top \boldsymbol{w}_i + \alpha_l m_{li}\left(T_i\right)\right\}\right]^{\delta_i}\\
&\times\exp\left[-\exp\left(\boldsymbol{\gamma_l}^\top\boldsymbol{w}_i\right)\int_0^{T_i}\exp\left\{\sum_q \gamma_{h_0,q,l}B_{q,l}(s, \boldsymbol{v}) + \alpha_l m_{li}(s)\right\}ds\right].
\end{aligned} \tag{11}$$

In the likelihood in (11), only the lth longitudinal outcome plays a role. Thus, just the random effects $\boldsymbol{b}_{li}$ contribute to the likelihood.

Now, we wonder if in any way the information provided by the posteriors of the univariate models resembles the information provided by the posterior of the multivariate model. To study this, we will see if we can find any relationship between the likelihoods. The reader can observe that by making the product of the likelihoods of the L univariate models we obtain:

$$
\begin{aligned}
&\prod_{l=1}^{L} p\left(\boldsymbol{y}_{li}, T_i, \delta_i \mid \boldsymbol{\theta}, \boldsymbol{b}_{li}\right) = \prod_{l=1}^{L} \Big[\prod_{j=1}^{n_{li}} p\left(y_{li}(t_{ij}) | \boldsymbol{b}_{li}, \boldsymbol{\theta}\right)\Big] p(\boldsymbol{b}_{li}|\boldsymbol{\theta}) p(T_i, \delta_i | \boldsymbol{b}_{li}, \boldsymbol{\theta}) \\
&= \left(\prod_{l=1}^{L} \Big[\prod_{j=1}^{n_{li}} p\left(y_{li}(t_{ij}) | \boldsymbol{b}_{li}, \boldsymbol{\theta}\right)\Big]\right) \left(\prod_{l=1}^{L} p(\boldsymbol{b}_{li}|\boldsymbol{\theta})\right) \left(\prod_{l=1}^{L} p(T_i, \delta_i | \boldsymbol{b}_{li}, \boldsymbol{\theta})\right) \\
&\propto \exp\left\{\sum_{l=1}^{L} \left(\frac{\sum_{j=1}^{n_{li}} (y_{li}(t_{ij}) m_{li}(t_{ij}) - A(m_{li}(t_{ij})))}{\phi_l}\right) + \sum_{l}\sum_{j} C(y_{li}(t_{ij}), \phi_l)\right\} \\
&\quad \times \prod_{l=1}^{L} \det(\boldsymbol{D}_l)^{-1/2} \exp\left(-\sum_{l=1}^{L} \boldsymbol{b}_{li}^{\top} \boldsymbol{D}_l^{-1} \boldsymbol{b}_{li}/2\right) \\
&\quad \times \left[\exp\left\{\sum_{l=1}^{L}\left(\sum_{q} \gamma_{h_0,q,l} B_{q,l}\left(T_i, \boldsymbol{v}\right)\right) + \left(\sum_{l=1}^{L} \boldsymbol{\gamma}_l^{\top}\right) \boldsymbol{w}_i + \sum_{l=1}^{L} \alpha_l m_{li}\left(T_i\right)\right\}\right]^{\delta_i} \\
&\quad \times \exp\left[-\sum_{l=1}^{L} \exp\left(\boldsymbol{\gamma}_l{}^{\top} \boldsymbol{w}_i\right) \int_0^{T_i} \exp\left(\sum_{q} \gamma_{h_0,q,l} B_{q,l}(s, \boldsymbol{v})\right) \left[\sum_{l=1}^{L} \exp\left(\alpha_l m_{li}(s)\right)\right] ds\right].
\end{aligned} \tag{12}
$$

The main goal in this section is to compare the posterior distribution of the multivariate joint model with the product of the L posterior distributions of the L univariate joint models. To achieve our objective, we start doing a comparison between (10) and (12). Observe that in (12) we have the product of the L univariate model likelihoods. We have that:

- The contribution of the longitudinal outcomes is identical. Notice that we have

$$
\Big[\prod_{l=1}^{L}\prod_{j=1}^{n_{li}} p\left(y_{li}(t_{ij}) | \boldsymbol{b}_{li}, \boldsymbol{\theta}\right)\Big] \quad = \quad \left(\prod_{l=1}^{L} \Big[\prod_{j=1}^{n_{li}} p\left(y_{li}(t_{ij}) | \boldsymbol{b}_{li}, \boldsymbol{\theta}\right)\Big]\right),
$$

  where we have the contribution of the longitudinal outcomes to the likelihood in (10) and (12), on the left and the right part, respectively.

- The contribution of the random effects is different. While in (10) the random effects are modeled jointly, accounting for the association between the L longitudinal outcomes, in (12) we have the product of the random effects. The latter contribution is because we are doing the product of the L likelihoods, so we are assuming the random effects to be mutually independent. More specifically we have

$$
p(\boldsymbol{b}_i|\boldsymbol{\theta}) \quad \neq \quad \left(\prod_{l=1}^{L} p(\boldsymbol{b}_{li}|\boldsymbol{\theta})\right),
$$

  having the contribution for the multivariate joint model on the left part, as well as the contribution for the product of the L likelihoods on the right part.

- The contribution of the survival outcomes is different. In this case we have different gamma parameters arising in the product of the L likelihoods, because for each univariate joint model the γ_l parameter comes from the posterior $[T_i, \delta_i | \boldsymbol{b}_{li}, \theta]$ (and the same for the coefficients in the baseline hazard function). Thus, the random effects to which we condition are different in each univariate model. We can see easily the difference when comparing:
$$p(T_i, \delta_i | \boldsymbol{b}_i, \boldsymbol{\theta}) \quad \neq \quad \left(\prod_{l=1}^{L} p(T_i, \delta_i | \boldsymbol{b}_{li}, \boldsymbol{\theta}) \right),$$
where the contribution in (10) is on the left part, and the contribution in (12) is on the right part.

Let us study now the differences between the priors and how they affect in the posterior. The prior distribution for the multivariate model is (8), while the prior for the lth model is

$$p(\boldsymbol{\theta}_l) = p(\boldsymbol{\beta}_l)p(\phi_l)p(\boldsymbol{\alpha}_l)p(vech(\boldsymbol{D}_l))p(\tau_l)p(\gamma_{h_{0,l}})p(\gamma_l). \tag{13}$$

Thus, when doing the product of the priors of the L models, we obtain the following

$$\begin{aligned}\prod_{l=1}^{L} p(\boldsymbol{\theta}_l) &= p(\boldsymbol{\beta}_1)\cdots p(\boldsymbol{\beta}_L)p(\phi_1)\cdots p(\phi_L)p(\boldsymbol{\alpha}_1)\cdots p(\boldsymbol{\alpha}_L)p(vech(\boldsymbol{D}_1))\cdots p(vech(\boldsymbol{D}_L)) \\ &\times p(\tau_1)\cdots p(\tau_L)p(\gamma_{h_{0,1}})\cdots p(\gamma_{h_{0,L}})p(\gamma_1)\cdots p(\gamma_L) \\ &\propto p(\boldsymbol{\beta}_1)\cdots p(\boldsymbol{\beta}_L)p(\phi_1)\cdots p(\phi_L)p(\boldsymbol{\alpha}_1)\cdots p(\boldsymbol{\alpha}_L) \\ &\times p(vech(\boldsymbol{D}_1))\cdots p(vech(\boldsymbol{D}_L))p(\tau)^L p(\gamma_{h_0})^L p(\gamma)^L.\end{aligned} \tag{14}$$

We have used that we use the same prior for all the gamma parameters, and then we can rewrite it as we did it, adding these L factors. Nonetheless, taking into account that we are using non-informative priors, the impact of these a prior distributions on the posterior distributions is not expected to be very significant. Thus, the differences between likelihoods will be much more important when comparing between posteriors.

3 Simulation study: more results, different scenarios

Based on the simulation study presented in [10], as well as in the supplementary material of the previous paper, a simulation study has been carried out to evaluate the performance of the SL algorithm and to assess whether SL can be useful to compute predictions and avoid the use of a multivariate joint model. To code the simulation I have used *Vignette of JMbayes2 R package* as a benchmark.

For the longitudinal outcomes, we consider $L = 5$ longitudinal processes. Three of them will be linear mixed models, and the remaining two models will be generalized linear mixed models (see model specification below). For the survival outcome, we consider the hazard model of the multivariate joint model (see model specification below). For this simulation study, several scenarios has been planned. Future simulations will explore adding more scenarios that can add robustness to the results. In all the scenarios we use the multivariate joint model as the data-generating model. Moreover, in a common way, in all scenarios we have simulated training and testing data, both containing 300 subjects. Let us expose the scenarios we have designed:

- **Scenario I:** We have used non-independent random effects within longitudinal processes, but independent random effects between longitudinal outcomes. Random censoring has been the mechanism used in this case.
- **Scenario II:** We have used non-independent random effects within and between longitudinal processes, as well as Type I censoring.

In both scenarios, 100 data sets have been simulated. For each simulated dataset, we implemented three-fold cross-validation, fitting the $L = 5$ univariate joint models (composed for the longitudinal sub-models separated) and the multivariate joint model. The cross-validated predictions from each model were used to estimate the model weights to optimize predictive performance via the super learning procedure. The discrete super learner was also compared with the ensemble super learner. Integrated Brier score and EPCE are the chosen predictive performance measures. We have assessed the predictive performance in the interval $(t, t + \Delta] = (4, 5.5]$. Let us expose the model specification and the parameter values used, the longitudinal models used are:

$$
\begin{aligned}
y_{1i}(t_{ij}) &= m_{1i}(t_{ij}) + \varepsilon_{1i}(t_{ij}) \\
&= (\beta_0^1 + b_{0i}^1) + (\beta_1^1 + b_{1i})t_{ij} + \beta_2^1 \texttt{sex}_i + \beta_3^1 \texttt{sex}_i t_{ij} + \varepsilon_{1i}(t_{ij}), \\
y_{2i}(t_{ij}) &= m_{2i}(t_{ij}) + \varepsilon_{2i}(t_{ij}) \\
&= (\beta_0^2 + b_{0i}^2) + (\beta_1^2 + b_{1i}^2)t_{ij} + \beta_2^2 \texttt{sex}_i + \varepsilon_{2i}(t_{ij}), \\
y_{3i}(t_{ij}) &= m_{3i}(t_{ij}) + \varepsilon_{3i}(t_{ij}) \\
&= (\beta_0^3 + b_{0i}^3) + (\beta_1^3 + b_{1i}^3)t_{ij} + \varepsilon_{3i}(t_{ij}), \\
\log\left(\frac{p(y_{4i}(t_{ij})=1)}{1-p(y_{4i}(t_{ij})=1)}\right) &= m_{4i}(t_{ij}) + \varepsilon_{4i}(t_{ij}) \\
&= (\beta_0^4 + b_{0i}^4) + (\beta_1^4 + b_{1i}^4)t_{ij} + \beta_2^4 \texttt{sex}_i + \varepsilon_{4i}(t_{ij}), \\
\log\left(\frac{p(y_{i5}(t_{ij})=1)}{1-p(y_{i5}(t_{ij})=1)}\right) &= m_{5i}(t_{ij}) + \varepsilon_{5i}(t_{ij}) \\
&= (\beta_0^5 + b_{0i}^5) + (\beta_1^5 + b_{1i}^5)t_{ij} + \varepsilon_{5i}(t_{ij}).
\end{aligned}
$$

Where the super-indices in the parameters denote the number of the sub-longitudinal model. As the reader can see, we have defined three linear mixed models, and two generalized linear mixed models. The values chosen for the fixed effects, in both scenarios, were $(\beta_0^1, \beta_1^1, \beta_2^1, \beta_3^1) = (-2.2, -0.25, 1.24, -0.05)$, $(\beta_0^2, \beta_1^2, \beta_2^2) = (-1.8, -0.06, 0.5)$, $(\beta_0^3, \beta_1^3) = (-2.5, 0.0333)$, $(\beta_0^4, \beta_1^4, \beta_2^4) = (0.01, 0.5, -0.31416)$ and $(\beta_0^5, \beta_1^5) = (1, 0.155)$. For the error terms of the first three models we assumed $\varepsilon_{li} \sim \mathcal{N}(0, \sigma_l^2)$, where $\sigma_1^2 = 0.125, \sigma_2^2 = 0.25, \sigma_3^2 = 0.25$. In the first Scenario the random effects were assumed to follow a multivariate normal distribution with mean zero and the following covariance matrices:

$$
D_1 = \begin{pmatrix} 1 & 0.00008 \\ 0.01478 & 0.5 \end{pmatrix}, D_2 = \begin{pmatrix} 1.2 & 0.0054 \\ 0.0000987 & 0.25 \end{pmatrix}, D_3 = \begin{pmatrix} 0.15 & 0.0005 \\ 0.07 & 0.05 \end{pmatrix},
$$
$$
D_4 = \begin{pmatrix} 0.212 & 0.01879 \\ 0.0000333 & 0.0125 \end{pmatrix}, D_5 = \begin{pmatrix} 0.0121 & 0.0098 \\ 0.0001212 & 0.0754 \end{pmatrix}.
$$

Then, as the reader can observe in the covariance matrices above, we assumed that random effects non-independent within longitudinal process. For Scenario II, we have used the following covariance matrix:

$$
D = \Sigma\Sigma^t,
$$

where Σ is the 10×10 matrix shown in the appendix.

From each model and for each subject we simulated longitudinal responses at time zero and then at 14 randomly selected time points from $U(0, 20)$. For the event time outcome, we simulated event times from the following hazard model by means of the inverse transform sampling method,

$$
h_i\left(t \mid \mathcal{Y}_i(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \sum_{l=1}^{5} \alpha_l m_{li}(t)\right\}, \quad t > 0, \tag{15}
$$

where $h_0(t) = \phi t^{\phi - 1}$ (thus, we are working with the Weibull distribution for the baseline hazard, the other parameter is γ_0, presented as an intercept inside the exponential function), with $\phi = 6.5$, $\gamma_0 = -15$, $\gamma_1 = -0.5$ and $\texttt{sex}_i$ denotes a binary sex indicator. In addition, the values for the association parameters are $\alpha_1 = 0.8, \alpha_2 = 0.61, \alpha_3 = 0.38, \alpha_4 = 0.222, \alpha_5 = 0.74$. All the previous parameters are the same in both scenarios. Regarding the censored mechanism used, in Scenario I, where we use random censoring, we have assumed censoring times are normally distributed with mean 6.5, standard deviation 1. In Scenario II it was fixed Type I, all event times greater than 6 were censored. Moreover, the longitudinal measurements that were taken after the observed time were dropped.

Furthermore, in order to make the reader aware of the library of univariate joint models $\mathcal{L} = \{M_1, M_2, M_3, M_4, M_5\}$ that we will use to carry out the super learning procedure, we write the models as follows:

$$\begin{aligned}
M1: &\quad h_i\left(t \mid \mathcal{Y}_{1i}(t), \texttt{sex}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_1 m_{1i}(t)\right\}, \\
M2: &\quad h_i\left(t \mid \mathcal{Y}_{2i}(t), \texttt{sex}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_2 m_{2i}(t)\right\}, \\
M3: &\quad h_i\left(t \mid \mathcal{Y}_{3i}(t), \texttt{sex}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_3 m_{3i}(t)\right\}, \\
M4: &\quad h_i\left(t \mid \mathcal{Y}_{4i}(t), \texttt{sex}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_4 m_{4i}(t)\right\}, \\
M5: &\quad h_i\left(t \mid \mathcal{Y}_{5i}(t), \texttt{sex}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_5 m_{5i}(t)\right\}.
\end{aligned}$$

Now, let us expose the results of the simulation study. In Scenario I, 38.32% and 36.81% of censoring has been obtained on average for training and testing data respectively. The results of Scenario I are shown in Figure 1. We can see for IBS that the multivariate joint model performs better than eSL and dSL in the training data. Nonetheless, it seems that both eSL and dSL perform better in testing data. This also indicates that the super learning procedure is responding well to over-fitting by showing a smaller performance difference between train and test data. The latter can also be observed in the right panel of Figure 1, where we have the results for the EPCE. eSL and dSL show similar results in training and testing data, while multivariate joint model performance undergoes more noticeable changes. Moreover, the EPCE values of the multivariate joint model show higher variability than the super learning values. However, multivariate joint model seems to perform better since EPCE values are smaller. The discrete super learner has been Model 1 (first univariate joint model) in all the cases for the EPCE, and in 98% of cases for IBS.

Results for Scenario II are shown in Figure 2. The reader can observe that the results obtained in this case are similar than above. Thus, the conclusions drawn are also similar. In the second scenario, we have obtained, on average, 42.05% and 41.58% of censoring for the training and testing data, respectively. In addition, the discrete super learner has been Model 2 (second univariate model) in all the 100 data sets for IBS, and in 93% of the data sets for EPCE.

Now, we would like to take a closer look at the differences between models. First, in order to analyze the possible effect of over-fitting on our models, we have calculated, for each of the simulated data sets, the difference between the proper scoring rule obtained by analyzing the

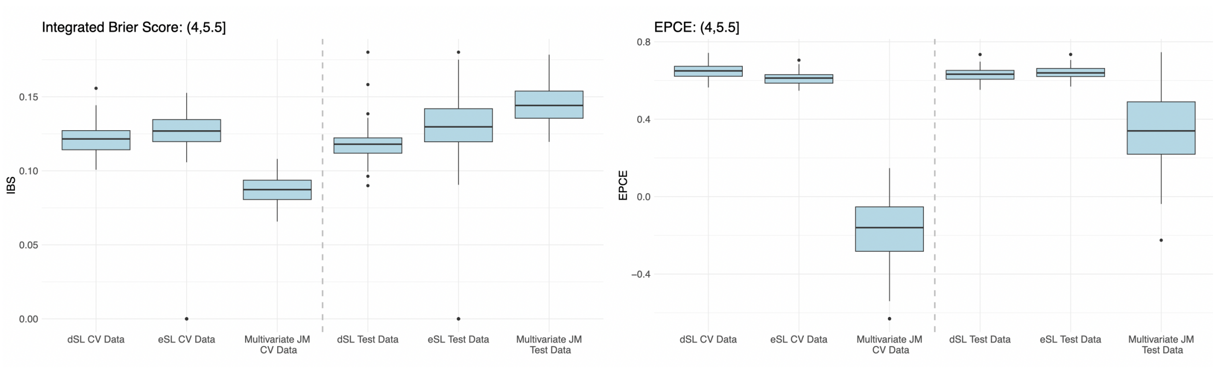

Figure 1: Simulation results for the IBS (left panel) and EPCE (right panel) under Scenario I for the time interval $(4, 5.5]$. In each panel, six boxplots are shown summarizing the results over the 100 simulated datasets; in the left part of the dashed vertical line there are metrics computed with the training data, in the right part with the test data. Each simulated training and testing dataset contains 300 subjects. Results are based on 3-fold CV.

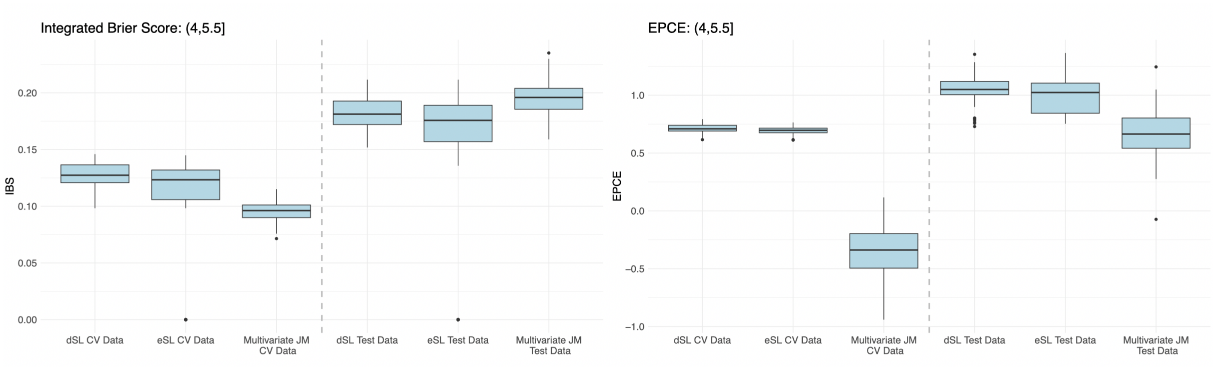

Figure 2: Simulation results for the IBS (left panel) and EPCE (right panel) under Scenario II for the time interval $(4, 5.5]$. In each panel, six boxplots are shown summarizing the results over the 100 simulated datasets; in the left part of the dashed vertical line there are metrics computed with the training data, in the right part with the test data. Each simulated training and testing dataset contains 300 subjects. Results are based on 3-fold CV.

testing data and the training data. We show the mean of those differences in Table 1. Positive values indicate that the scoring rules are, on average, bigger in the testing data than in training data. The latter is to be expected in general. Nonetheless, in Scenario 1, it seems that for both scoring rules dSL is improving the performance in testing data. The most remarkable thing about this table is that the difference between model performance in train and test data is larger in the multivariate joint model than in dSL and eSL. Hence, it seems that the super learning procedure appropriately accounts for over-fitting. Furthermore, the reader can observe that differences are bigger in the EPCE compared with IBS.

Our next objective is to study how big is the difference between the multivariate joint model and the dynamic predictions resulting from the super learning procedure. For this purpose, we have calculated for each simulated data set the difference between the accuracy of the dSL and the eSL and that of the multivariate joint model. The average of these differences

	Scenario 1			Scenario II		
	dSL	eSL	Multi JM	dSL	eSL	Multi JM
IBS	-0.0037	0.0041	0.0581	0.0547	0.0425	0.0993
EPCE	-0.0242	0.0306	0.4994	0.3242	0.2834	0.94521

Table 1: Mean of the differences, in Scenario I and Scenario II, between testing and training data for the dSL, eSL and multivariate joint model, as well as for the two different scoring rules IBS and EPCE.

can be found in Table 2 (for Scenario I) and Table 3 (for Scenario II). In this context, negative values means that, on average, the proper scoring rule is smaller (and then better) for either dSL or eSL than for the multivariate joint model. We can see that in both scenarios in the testing data, and for the IBS, the super-learning-based predictions are better than the multivariate joint model predictions. This result is surprising, considering that the multivariate joint model is the model we have used to simulate the data, i.e. the true model. Although for the EPCE we are not improving the multivariate joint model predictions, we can see that the gap between the dSL (idem eSL) and the multivariate joint model decreases considerably when using testing data.

	Train		Test	
	$dSL - mJM$	$eSL - mJM$	$dSL - mJM$	$eSL - mJM$
IBS	0.0345	0.0349	-0.0272	-0.0191
EPCE	0.8015	0.7627	0.2779	0.2938

Table 2: Mean of the differences, in Scenario I, between models within train and testing data respectively. mJM denotes multivariate joint model.

	Train		Test	
	$dSL - mJM$	$eSL - mJM$	$dSL - mJM$	$eSL - mJM$
IBS	0.0316	0.0031	-0.0130	-0.0536
EPCE	1.0004	0.9772	0.3795	0.3153

Table 3: Mean of the differences, in Scenario II, between models within train and testing data respectively. mJM denotes multivariate joint model.

Finally, I would like to point out that, in general, the IBS results have been more stable than those of EPCE. In fact, in several data sets (32 and 55, in first and second scenario, respectively) when calculating the EPCE for the multivariate joint model on the test data we have obtained an `Inf`. Moreover, as can be seen in Figure 1 and Figure 2, there is much more variability in the EPCE of the multivariate joint model than in the other models.

4 Comparison between super learning and other methods for calculating dynamic predictions

Super learning, an ensemble learning method, has gained attention in the literature for its capacity to optimally combine predictions from a diverse set of candidate algorithms, potentially offering improved predictive performance over traditional approaches. In order to validate the efficacy of the super learning procedure, this section will provide a comparative analysis against other widely used models for calculating dynamic predictions when working under the paradigm of having several longitudinal outcomes.

Among these alternative methods, functional principal component analysis (FPCA) [13] has been a prominent tool for handling functional data, offering a flexible framework to capture dynamic changes over time. By comparing super learning with models based on FPCA, we aim to assess the predictive performance across a variety of scenarios. Our objective is to demonstrate that the Super Learning approach can consistently outperform these methods.

The structure of this section, as well the models that will be used, is mainly based on [6]. A detailed study of the models we are going to use has been carried out through the different papers where these models are presented. These references will be shared throughout this section.

4.1 Functional principal component analysis-based methods

Functional data analysis provides a statistical framework that is flexible and well suited to model sparsely sampled longitudinal data, and time-to-event outcomes. Nevertheless, we are interested in the situation where we have a large number of longitudinal outcomes. Hence, to deal with our problem we will use multivariate functional principal component analysis (MFPCA) [2]. In this section we will explain the theory behind this method of summarizing longitudinal outcomes.

In order to work within the framework of the MFPCA, we must make some restrictions on the assumptions we made above. The most important restriction is that we are going to assume throughout this section that the response vector $\boldsymbol{y}_i$ conditional on the random effects $\boldsymbol{b}_i$ has a continuous distribution $\mathcal{F}_\Psi$. Although a normality assumption is not required, the longitudinal outcomes must be continuous.

To model continuous longitudinal outcomes, we assume that the observed data $\boldsymbol{y}_{li}$ is a noisy measurement of the latent outcome process $X_{li}(t)$, where time $t \in \mathcal{T} = [0, \tau]$ and $\tau = max\{T_i^* : i = 1, \dots, n\}$. This is, we end up with $y_{li}(t_{ij}) = X_{li}(t_{ij}) + \varepsilon_{li}(t_{ij})$, where $\varepsilon_{li}(t_{ij})$ are independent errors centered and with variance $\sigma^2_{\varepsilon_l}$.

Let $\boldsymbol{X}_i(t) = \{X_{li}(t)\}_{l=1,\dots,L}$ be the multivariate longitudinal processes of the ith subject. Let us explain how to model $\boldsymbol{X}_i(t)$ by means of MFPCA. A two-step procedure will be used. In the first step, we let $\mu_l(t)$ be the unknown smoothed mean function of $X_{li}(t)$ (the longitudinal process of the ith subject and lth longitudinal outcome). Moreover, we let $\Sigma_l(t, t') = \texttt{cov}\{X_{li}(t), X_{li}(t')\}$ be the covariance function that models the correlation of outcome l's trajectory between time points t and t'. By the Mercer's theorem (see the Mercer's Theorem), the spectral decomposition of the covariance function is

$$\Sigma_l(t,t') = \mathtt{cov}\{X_{li}(t), X_{li}(t')\} = \sum_{r=1}^{\infty} \lambda_{lr}\phi_{lr}(t)\phi_{lr}(t'),$$

where $\{\lambda_{lr}\}_{r=1,\dots,\infty}$ are non-increasing eigenvalues and $\{\phi_{lr}(t)\}_{r=1,\dots,\infty}$ are the corresponding orthonormal eigenfunctions. Furthermore, the Karhunen Loève (see KL theorem) expansion of the stochastic process $X_{li}(t)$ is given by

$$X_{li}(t) = \mu_l(t) + \sum_{r=1}^{\infty} \xi_{ilr}\phi_{lr}(t), \tag{16}$$

where the so-called FPC scores $\{\xi_{ilr}\}_{r=1,\dots,\infty}$ are uncorrelated random variables with mean zero and variance λ_{lr}. It is, we have that $E(\xi_{ilr}) = 0$, for all $r \in \mathbb{N}$, as well as $E(\xi_{ilr_1}\xi_{ilr_2}) = \mathbb{1}(r_1 = r_2)\lambda_{lr_1}$, for all $r_1, r_2 \in \mathbb{N}$. Moreover, we can think in $\phi_{lr}(t)$ as a changing pattern of the longitudinal process, and the random variable ξ_{ilr} is describing how strongly the data from the ith subject follow this pattern. We assume that the longitudinal process can be approximated by the first R_l eigenfunctions. Thus, the truncated version of the individual trajectory, $X_{li}(t) \approx \mu_l(t) + \sum_{r=1}^{R_l} \xi_{ilr}\phi_{lr}(t)$, will be the quantity used. A prespecified percentage of variance explained (PVE) will be used to determine the number of components R_l. Say $\text{PVE} = 80\%, 90\%, 95\%$.

In our framework, observed outcomes of $\boldsymbol{y}_{li}$ are available only at discrete random times t_{ij}, as well as missing for $t > T_i^*$. The FPCA is conducted via principal analysis by conditional estimation (PACE) algorithm [14]. The PACE method has been proven to be versatile and powerful when applied to sparse and irregularly measured longitudinal data contaminated with measurements errors. The PACE algorithm is applied to all the observed data for the lth outcome, generating the following estimated quantities: mean function $\hat{\mu}_l(t)$, error variance $\hat{\sigma}_{\varepsilon_l}$, covariance function $\hat{\Sigma}_l(t,t')$, eigenvalues $\hat{\lambda}_{lr}$, and eigenfunctions $\hat{\phi}_{lr}(t)$. Using these quantities we can estimate the FPC scores of subject i as follows

$$\hat{\xi}_{ilr} = \hat{\lambda}_{lr}(\hat{\boldsymbol{\phi}}_{ilr})^\top \hat{\Sigma}_{\boldsymbol{y}_{li}}^{-1}(\boldsymbol{y}_{li} - \hat{\boldsymbol{\mu}}_{li}) \tag{17}$$

where we have the vectors $\hat{\boldsymbol{\mu}}_{li} = (\hat{\mu}_l(t_{11}), \dots, \hat{\mu}_l(t_{1n_{li}}))$, $\hat{\boldsymbol{\phi}}_{ilr} = (\hat{\phi}_{lr}(t_{11}), \dots, \hat{\phi}_{lr}(t_{1n_{li}}))$ and $\hat{\Sigma}_{\boldsymbol{y}_{li}}$ is a $n_{li} \times n_{li}$ matrix with entries

$$\left(\hat{\Sigma}_{\boldsymbol{y}_{li}}\right)_{j,j'} = \hat{\Sigma}_l(t_{ij}, t_{ij'}) + \mathbb{1}(j = j')\hat{\sigma}^2_{\varepsilon_l}.$$

We apply the PACE algorithm on all L longitudinal outcomes and estimate the eigenfunctions $\hat{\boldsymbol{\phi}}(t) = (\hat{\phi}_{l1}(t), \dots, \hat{\phi}_{lR_l}(t))$ and the FPC scores $\hat{\boldsymbol{\xi}}_{li} = (\hat{\xi}_{il1}, \dots, \hat{\xi}_{ilR_l})$, where R_l is a proper truncation for the lth longitudinal outcome. Let us denote the vector of estimated FPC scores across all longitudinal outcomes for the ith subject as $\hat{\boldsymbol{\xi}}_i = (\hat{\boldsymbol{\xi}}_{1i}, \dots, \hat{\boldsymbol{\xi}}_{li})$, the length of the previous vector is denoted as $R_+ = \sum_{l=1}^{L} R_l$.

Since each longitudinal outcome is associated with the disease progression, there may exist nonnegligible correlation among the FPC scores derived from multiple longitudinal outcomes. In the second step of the procedure used, MFPCA implicitly models the correlations among outcomes by means of the correlations among the FPC scores. Let V be an $n \times R_+$ matrix such that whose ith row is $\hat{\boldsymbol{\xi}}_i^\top$. Now, we consider the $R_+ \times R_+$ matrix $H = (n-1)^{-1}V^\top V$. A matrix eigenanalysis is performed on H, resulting into estimated eigenvalues $\{\hat{\nu}_k\}_{k=1,\dots,R_+}$ and orthonormal eigenvectors $\{\hat{\boldsymbol{c}}_k\}_{k=1,\dots,R_+}$. Estimates for the multivariate eigenfunctions for the lth outcome are given by

$$\hat{\Psi}_{lk}(t) = \sum_{r=1}^{R_l} [\hat{\boldsymbol{c}}_k]_r^{(l)} \hat{\phi}_{lr}(t),$$

where $[\hat{\boldsymbol{c}}_k]_r^{(l)}$ is denoting the lth block of the orthonormal eigenvector $\hat{\boldsymbol{c}}_k$. The set of multivariate eigenfunctions $\hat{\boldsymbol{\Psi}}_k = \{\hat{\Psi}_{lk}(t)\}_{k=1,\dots,R_+}$ characterizes the kth changing pattern of the multivariate longitudinal processes $\boldsymbol{X}_i(t)$. Estimates for the MFPC scores of subject i can be computed as

$$\hat{\rho}_{ik} = \sum_{l=1}^{L} \sum_{r=1}^{R_l} [\hat{\boldsymbol{c}}_k]_r^{(l)} \hat{\xi}_{ilr}. \tag{18}$$

Lastly, the lth longitudinal outcome, $X_{li}(t)$, can be sufficiently approximated by selecting the first $R^* \le R_+$ scores and eigenfunctions based on PVE, and computing

$$E(y_{li}(t)) = \hat{X}_{li}(t) \approx \hat{\mu}_l(t) + \sum_{k=1}^{R^*} \hat{\rho}_{ik} \hat{\Psi}_{lk}(t). \tag{19}$$

Implementation of MFPCA can be found in the R package `MFPCA` [2]. An important observation is that the current implementation requires the longitudinal data to lie on a grid with fixed intervals. Nonetheless, missing data is allowed. A consequence of using a fixed grid, is that we lose flexibility compared with our initial framework. When using FPCA-based methods in this work, the observed longitudinal data $y_{li}(t_{ij})$ is first rounded to the nearest time corresponding to the fixed grid. A $n \times J \times L$ input matrix will be used to deal with the longitudinal data, where J is the grid length needed to accommodate the largest observation time.

4.1.1 MFPCA-Cox model

The main goal of our work is to make dynamic predictions. Therefore, MFPC scores $\hat{\boldsymbol{\rho}}_i$, which is computed using the follow-up data, can be used as predictors in modeling the relationship between time-to-event and the patterns of longitudinal outcomes. Cox proportional hazards model will be used to model the hazard function of ith subject,

$$h_i\left(t \mid \mathcal{Y}_i(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\boldsymbol{\gamma}^\top \boldsymbol{w}_i + \boldsymbol{\beta}^\top \hat{\boldsymbol{\rho}}_i\right\}, \tag{20}$$

where $h_0(t)$ is an unspecified baseline hazard function and $\boldsymbol{w}_i$ is a vector of baseline covariates with corresponding regression coefficients $\boldsymbol{\gamma}$. Moreover, $\boldsymbol{\beta}$ (not be confused with the β's used previously to model the longitudinal outcomes via GLMs or LMMs) is the vector of regression coefficients for the multivariate longitudinal predictors through the R^* estimated

MFPC scores. This framwork is referred from now on as MFPCA-Cox, and it was firstly proposed in [5].

Furthermore, let us explain how personalized dynamic predictions are computed when using MFPCA-Cox. In summary, what we do is to fit the model (20) in the training data set and use the estimated parameters to compute dynamic predictions for new patients. Specifically, when fitting the model in training data, the first step is to obtain estimated parameters in functional data analysis and scores $\hat{\boldsymbol{\rho}}_i$ using all the longitudinal measurements in the training data set. Then, regression coefficients $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are estimated by fitting (20) using the estimated MFPC scores, time-independent covariates, and survival information of subjects in training data set. Finally, the baseline hazard function is estimated $\hat{h}_0(t)$ via the Breslow estimator.

Now, assume we are interested in computing dynamic predictions for a new subject i^* who is event-free until time t. We compute the FPC scores of the lth outcome, based on data up to time t, for i^*th subject by means of (17). The quantities needed in (17) are derived via the PACE algorithm. We repeat the latter with all the L longitudinal outcomes. Having all the FPC scores from the different L longitudinal outcomes, we calculate the MFPC score $\hat{\boldsymbol{\rho}}_{i^*}$ by means of (18). To predict the risk of an event not occurring within a time window $(t, u]$, we use the conditional probability of event-free at any time u after t

$$\pi_j(u \mid t) = P\left(T_{i^*}^* \geq u \mid T_{i^*}^* > t, \mathcal{Y}_j(t), \boldsymbol{\gamma}, \boldsymbol{\beta}, \hat{\boldsymbol{\rho}}_i\right) = \left(\frac{\hat{S}_0(u)}{\hat{S}_0(t)}\right)^{\exp\{\hat{\boldsymbol{\gamma}}^\top \boldsymbol{w}_i + \hat{\boldsymbol{\beta}}^\top \hat{\boldsymbol{\rho}}_i\}}, \tag{21}$$

where $\hat{S}_0(t) = \exp\{-\int_0^s \hat{h}_0(t)dt\}$ is the baseline survival function. Of course, these risk predictions can be updated dynamically. Note that (21) is the same quantity exposed in (5) but for the MFPCA-Cox model, instead of the multivariate joint model. Once we have the prediction computed, predictive performance can be assessed as we did above for the multivariate joint models and the super learning procedure predictions.

4.1.2 MFPCA-DeepSurv model

The same methodology exposed above has been applied to incorporate longitudinal data information into the machine learning (ML) model DeepSurv [3]. DeepSurv is a feed-forward neural network which predicts the effects of a patient's covariates on their hazard rate parameterized by the weights of the network $\boldsymbol{\theta}$. Observe that here we use $\boldsymbol{\theta}$ to denote the weights of the network, while in previous sections $\boldsymbol{\theta}$ was denoting the vector of all the parameters involved in the model. Thus, the choice of $\boldsymbol{\theta}$ in this ML context is meaningful. This is because the weights of the network will be playing the role, in some manner, of the parameters of the model.

In the Cox model, a subject's log-risk of experiencing the event is modeled as a linear combination of the subject's covariates, let us denote it as $\hat{g}(w) = \boldsymbol{\gamma}^\top \boldsymbol{w}_i$. Nonlinear effects such as interactions between covariates have to be added and depend on the knowledge on the problem at hand. Nonetheless, DeepSurv is able to learn complex and nonlinear relationships between patient's covariates and their event risk.

While in the original DeepSurv model [3] the only input to the network is a patient's baseline data $\boldsymbol{w}_i$, in the so-called MFPCA-DeepSurv model [6] the input used is $\boldsymbol{w}_i$ and the estimated MFPC scores $\hat{\boldsymbol{\rho}}_i$. Then, we have the hidden layers of the network consisting of a fully-connected layer of nodes, followed by a dropout layer. The latter is used to prevent our feed-forward neural network from over-fitting [11]. The output of the network $\hat{g}_{\boldsymbol{\theta}}(w)$ is a single node with a linear activation which estimates the log-hazard function in the following Cox model

$$h(t|x) = h_0(t)\exp\{g(w)\}. \tag{22}$$

Notice that the output of the network $\hat{g}_{\boldsymbol{\theta}}(x)$ is not necessarily a linear combination of the covariates. In addition, the output accounts for nonlinear patterns by using multiple hidden layers and activation functions. DeepSurv is trained on a loss function (or objective function) based on the Cox partial likelihood function [1]. In particular, the model minimizes the average negative log partial likelihood

$$l(\boldsymbol{\theta}) = -\frac{1}{N_{\delta=1}} \sum_{i|\delta_i=1} \left(\hat{g}_{\boldsymbol{\theta}}(w_i) - \log \left(\sum_{j\in\mathcal{R}(T_i^*)} \exp(\hat{g}_{\boldsymbol{\theta}}(w_j)) \right) \right), \tag{23}$$

where $N_{\delta=1}$ is the total number of subjects who had suffered the event and $\mathcal{R}(t)$ is the set of patients at risk at time t. Gradient descent optimization is used to find the weights of the network minimizing (23).

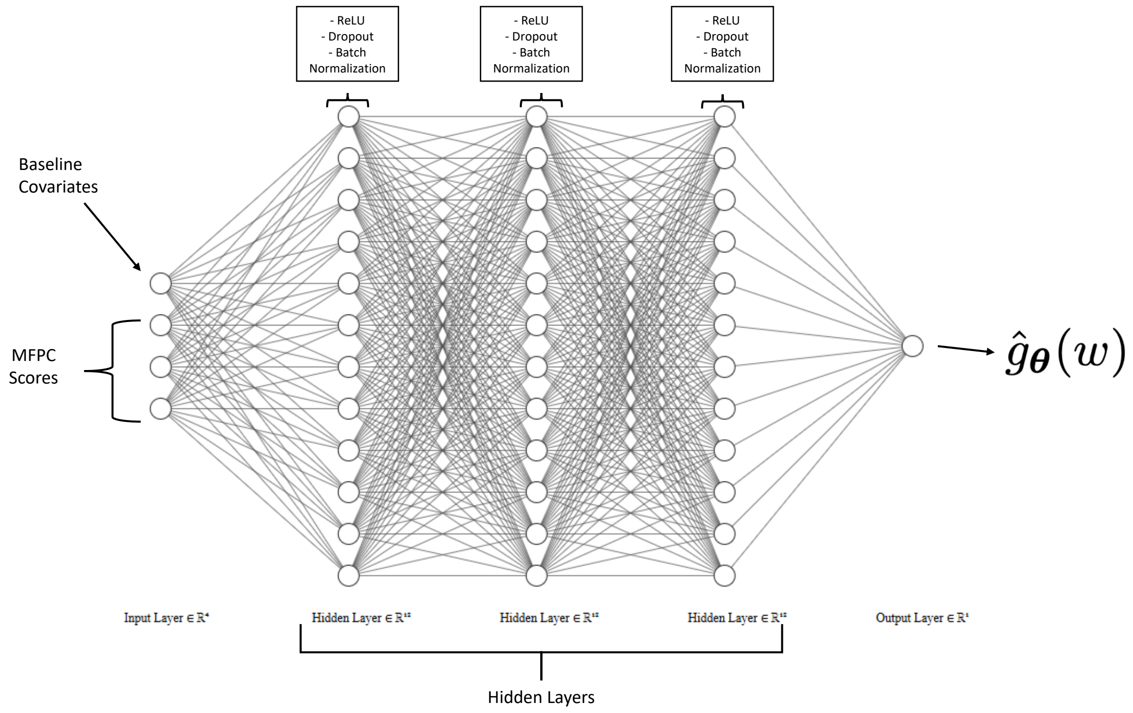

Figure 3: Outline of MFPCA-DeepSurv neural network architecture, with three hidden layers, each one with 12 nodes.

The outline of the MFPCA-DeepSurv model is presented in Figure 3. Several deep learning techniques are used to optimize the training of the network. Each linear layer in the feed-forward neural network is followed by an activation function. In the implementation used in [6], rectified linear unit (ReLU) is the activation function chosen. Where $ReLU(x) = max(0, x)$. The latter is followed by the previously mentioned dropout layer. Finally, batch normalization is used in order to re-center and re-scale the layer output to allow for faster and more stable training. Besides the aforementioned techniques, adaptive moment estimation (Adam) [4] is used for the gradient descent algorithm.

Finally, we have to refer to an important part of machine learning models, which are the hyper-parameters. Although a Random hyper-parameter optimization search is used in [3], we will be following the workflow used in [6] and a simple grid search will be applied.

Furthermore, once we have the output of the feed-forward neural network $\hat{g}_{\boldsymbol{\theta}}(w)$, we can see that in the Cox model (22) only remains to be estimated the baseline hazard function $\hat{h}_0(t)$. The latter is done by means of the Breslow estimator, as in the MFPCA-Cox model. Thus, individualized dynamic predictions can be computed using (21), taking into account that $\exp\{\hat{\boldsymbol{\gamma}}^\top \boldsymbol{w}_i + \hat{\boldsymbol{\beta}}^\top \hat{\boldsymbol{\rho}}_i\}$ should be replaced by $\exp\{\hat{g}_{\boldsymbol{\theta}}(w)\}$ in (21).

4.2 Simulation study

A simulation study has been performed to compare the performance of the MFPCA-Cox and MFPCA-DeepSurv models with the super learning algorithm in the joint modeling context. It is important to note that these simulations have been made taking into account the restrictions regarding the initial framework we have made in order to be able to use the models presented in this section. The main restriction is that in this simulation study the longitudinal outcomes used are all continuous.

A total amount of $L = 5$ longitudinal outcomes are considered in the simulation study. All of them are generated via linear mixed models (see model specification below). For the survival outcome, we consider the hazard model of the multivariate joint model (see model specification below). Similarly than in the simulation study conducted above, we consider only one data-generating model, which is the multivariate joint model. Two different scenarios have been planned to test the performance of dynamic predictions in different situations. In both scenarios we have simulated training and testing data independently, both containing 300 subjects. The two scenarios planned are:

- **Scenario I:** Random censoring has been used.
- **Scenario II:** Type I (administrative) censoring has been used.

For both scenarios 100 data sets have been simulated (100 for testing data and another 100 for testing data). We have used non-independent random effects within and between longitudinal outcomes. For each simulated dataset, we implemented three-fold cross-validation, fitting the $L = 5$ univariate joint models (composed for the longitudinal sub-models separated) and the multivariate joint model. The cross-validated predictions from each model were used to estimate the model weights to optimize predictive performance via the super learning procedure. The discrete super learner was also compared with the ensemble super learner. Moreover, for each simulated data set, MFPCA-Cox and MFPCA-DeepSurv models have been used to derive dynamic predictions. Integrated Brier score using inverse probability of censoring weights (IPCW) is the chosen predictive performance measure. IPCW is used to appropriately account for patients who were censored in the interval $(t, t + \Delta t]$ [10]. Until now in this work we had used model-based weights instead of IPCW. Furthermore, we have assessed the predictive performance in the interval $(t, t + \Delta t] = (2.5, 3.5]$.

Let us expose the model specification and the parameter values used, the longitudinal models used are:

$$\begin{aligned}
y_{1i}(t_{ij}) &= m_{1i}(t_{ij}) + \varepsilon_{1i}(t_{ij}) \\
&= (\beta_0^1 + b_{0i}^1) + (\beta_1^1 + b_{1i})t_{ij} + \beta_2^1 \texttt{sex}_i + \beta_3^1 \texttt{sex}_i t_{ij} + \varepsilon_{1i}(t_{ij}), \\
y_{2i}(t_{ij}) &= m_{2i}(t_{ij}) + \varepsilon_{2i}(t_{ij}) \\
&= (\beta_0^2 + b_{0i}^2) + (\beta_1^2 + b_{1i})t_{ij} + \beta_2^2 \texttt{treatment}_i + \beta_3^2 \texttt{treatment}_i t_{ij} + \varepsilon_{2i}(t_{ij}), \\
y_{3i}(t_{ij}) &= m_{i3}(t_{ij}) + \varepsilon_{3i}(t_{ij}) \\
&= (\beta_0^3 + b_{0i}^3) + (\beta_1^3 + b_{1i}^3)t_{ij} + \varepsilon_{3i}(t_{ij}), \\
y_{4i}(t_{ij}) &= m_{4i}(t_{ij}) + \varepsilon_{4i}(t_{ij}) \\
&= (\beta_0^4 + b_{0i}^4) + (\beta_1^4 + b_{1i}^4)t_{ij} + \beta_2^4 \texttt{sex}_i + \varepsilon_{4i}(t_{ij}), \\
y_{5i}(t_{ij}) &= m_{5i}(t_{ij}) + \varepsilon_{5i}(t_{ij}) \\
&= (\beta_0^5 + b_{0i}^5) + (\beta_1^5 + b_{1i}^5)t_{ij} + \beta_2^5 \texttt{treatment}_i + \varepsilon_{5i}(t_{ij}).
\end{aligned}$$

Where the super-indices in the parameters denote the number of the longitudinal model, $\texttt{sex}_i$ denotes a binary sex indicator and $\texttt{treatment}_i$ denotes a binary treatment indicator. Since all the longitudinal outcomes have to be continuous to apply MFPCA, all the longitudinal processes are modeled via linear mixed models, as the reader can see in the expressions above. The values chosen for the fixed effects were $(\beta_0^1, \beta_1^1, \beta_2^1, \beta_3^1) = (-2.2, -0.25, 1.24, -0.05)$, $(\beta_0^2, \beta_1^2, \beta_2^2, \beta_3^2) = (-1.8, -0.06, 0.5, 0.06)$, $(\beta_0^3, \beta_1^3) = (-2.5, 0.0333)$, $(\beta_0^4, \beta_1^4, \beta_2^4) = (0.01, 0.5, -0.31416)$ and $(\beta_0^5, \beta_1^5, \beta_2^5) = (1, 0.155, 0.12345)$. For the error terms we assumed $\varepsilon_{li} \sim \mathcal{N}(0, \sigma_l^2)$, where $\sigma_1^2 = 0.125, \sigma_2^2 = \sigma_3^2 = \sigma_4^2 = \sigma_5^2 = 0.25$. The random effects were assumed to follow a multivariate normal distribution with mean zero and the following covariance matrix:

$$D = \Sigma \Sigma^t,$$

where Σ the 10×10 matrix shown in the appendix of this work.

From each model and for each subject we simulated longitudinal responses at time zero and then at 16 randomly selected time points coming from $U(0, 10)$. In addition, a constraint has been used when selecting the time points, a minimum distance of 0.5 between randomly selected points is required. The latter is because when using the grid of fixed time intervals to compute MFPC scores, a grid with a distance of 0.5 between time points is used. Thus, if two time points are at a distance less than 0.5, the model will use only the first time point and the other one will be ignored. This causes that we do not have enough data to fit the MFPCA-based models. Of course, this constraint is not necessary when using the super learning algorithm due to the flexibility of the procedure to deal with different data structures.

For the event time outcome, we simulated event times from the following hazard model by means of the inverse transform sampling method,

$$h_i\left(t \mid \mathcal{Y}_i(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \sum_{l=1}^{5} \alpha_l m_{li}(t)\right\}, \quad t > 0, \tag{24}$$

where $h_0(t) = \phi t^{\phi-1}$ (thus, we are working with the Weibull distribution for the baseline hazard, the other parameter is γ_0, presented as an intercept inside the exponential function), with $\phi = 3.5$, $\gamma_0 = -7.5$ and $\gamma_1 = -0.5$. In addition, the values for the association parameters are $\alpha_1 = 0.8, \alpha_2 = 0.61, \alpha_3 = 0.38, \alpha_4 = 0.222, \alpha_5 = 0.74$.

Moreover, in order to make the reader aware of the library of univariate joint models $\mathcal{L} = \{M_1, M_2, M_3, M_4, M_5\}$ that will be used to conduct the super learning algorithm, we write the models as follows:

$$\begin{aligned}
M1: &\quad h_i\left(t \mid \mathcal{Y}_{1i}(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_1 m_{1i}(t)\right\}, \\
M2: &\quad h_i\left(t \mid \mathcal{Y}_{2i}(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_2 m_{2i}(t)\right\}, \\
M3: &\quad h_i\left(t \mid \mathcal{Y}_{3i}(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_3 m_{3i}(t)\right\}, \\
M4: &\quad h_i\left(t \mid \mathcal{Y}_{4i}(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_4 m_{4i}(t)\right\}, \\
M5: &\quad h_i\left(t \mid \mathcal{Y}_{5i}(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\gamma_0 + \gamma_1 \texttt{sex}_i + \alpha_5 m_{5i}(t)\right\}.
\end{aligned}$$

Furthermore, model specification for the MFPCA-based models must be reported. Firstly, in order to compute the MFPCA, a grid with fixed time intervals has been used. These time intervals occurs every 0.5 units of time. An 80% of PVE has been used. The latter is used to compute the number of principal components to be computed. For the MFPCA-DeepSurv model, the hyperparameters of the feed-forward neural network used are: 2 hidden layers, 64 nodes per hidden layer, 0.2 dropout probability, 12 number of epochs and a batch size of 32. In both MFPCA-based models, Breslow estimator has been used to estimate the baseline hazard function.

All the model specifications above are common for both scenarios. However, differences arise for the censoring mechanism. In Scenario I random censoring has been used, the distribution of censored times follows a exponential distribution with rate 1/10. On average, we have obtained 41.12% and 41.94% of censoring in training and testing data, respectively. In Scenario II, Type I censoring has been used. Event times greater than 5 were censored. On average 40.48% and 40.82% of censoring have been obtained in training and testing data, respectively.

Figures 4 and 5 show the results for the integrated Brier score (using IPCW) under the two scenarios, respectively. In both scenarios we can see similar results. While in the training data all models, except the multivariate joint model, give similar performance, in the testing data we see that eSL and dSL produce better results than MFPCA-Cox and MFPCA-DeepSurv. In both scenarios eSL is giving the most accurate prediction in testing data. As we could see in the previous simulation study, predictions based on the super learning algorithm seem to better account for over-fitting. Furthermore, as in the first simulation study, it appears that dSL and eSL perform better in testing data than the data-generating model (the multivariate joint model). Although not as expected, this may be due to the good performance with the over-fitting of the predictions based on the super learning algorithm. In Scenario I, in 93% of simulated datasets the discrete super learner is the univariate Model 2, in the remaining data sets the discrete super learners is either the univariate Model 5 (6%) or Model 1 (1%). In Scenario II, in the 94% of data sets the dSL is the univariate joint Model 2, in the rest of cases the univariate Model 5.

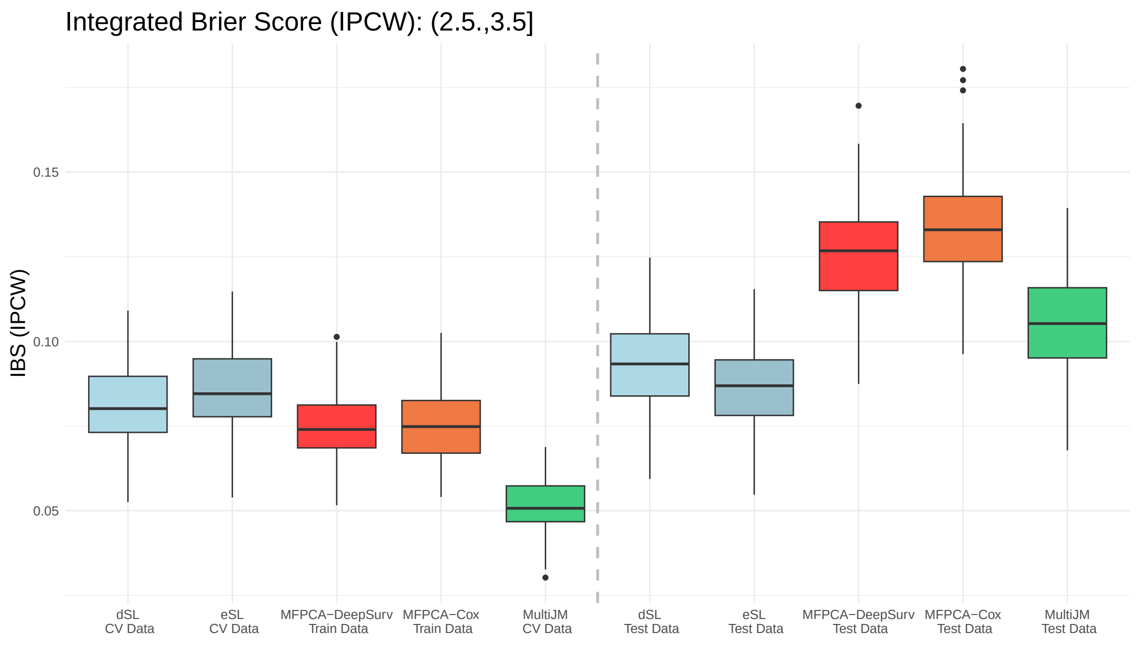

Figure 4: Simulation results for the IBS (IPCW) under Scenario I for the time interval $(2.5, 3.5]$. Ten boxplots are shown summarizing the results over the 100 simulated datasets; in the left part of the dashed vertical line there are metrics computed with the training data, in the right part with the testing data. Each simulated training and testing dataset contains 300 subjects. Results related with the SL procedure are based on 3-fold CV.

Nonetheless, a remarkable advantage of the MFPCA-based models with respect to super learning procedures is the computation time. Almost two entire days are necessary to fit the multivariate joint models, as well as carry out the super learning procedures and to compute the proper scoring rule for each scenario (Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz and 8.0 GB RAM with Microsoft Windows 11 Education operating system). On the other hand, for the MFPCA-Cox model 4.42 and 8.56 minutes are the running times for Scenario I and Scenario II, respectively. In addition, for the MFPCA-DeepSurv model, 4.68 and 5.07 minutes are the running times. For the MFPCA-based methods we have used Python instead of R. Moreover, for the latter models a laptop with an Intel Core i5 @2.3GHz with dual-core, with macOS Ventura 13.6.7 as operating system. These times that we have presented are not entirely comparable, since the running times of the super learning algorithm also take into account the time it takes

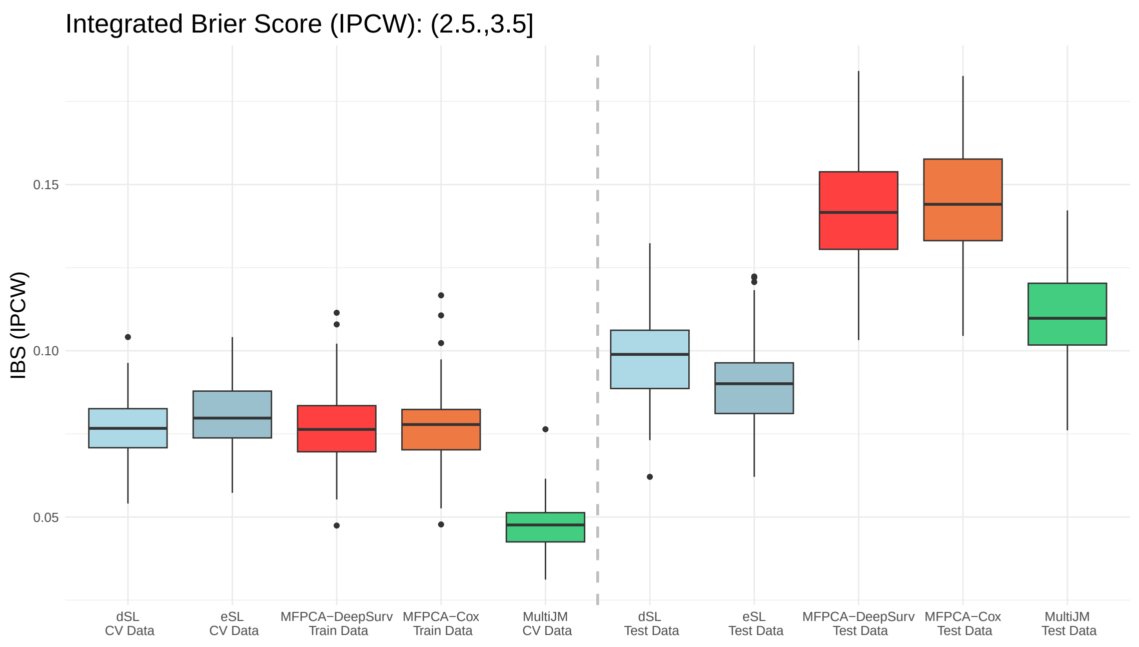

Figure 5: Simulation results for the IBS (IPCW) under Scenario II for the time interval $(2.5, 3.5]$. Ten boxplots are shown summarizing the results over the 100 simulated datasets; in the left part of the dashed vertical line there are metrics computed with the training data, in the right part with the testing data. Each simulated training and testing dataset contains 300 subjects. Results related with the SL procedure are based on 3-fold CV.

to adjust the multivariate model, a time that we know to be high. Moreover, running time for MFPCA-DeepSurv model is not taking into account the hyperparameter tuning running time. However, it seems clear that the MFPCA-based models, implemented via Python, have a lower computation time.

5 Further comments and details

To finish this report, I would like to add a short section with some details about the progress I have made since the last report that I think are important.

5.1 About Bayesian model averaging

In order to compare the super learning algorithm with some other method of weighting different models, I thought that using Bayesian model averaging (BMA) could be an attractive option. Carrying out a BMA is by no means trivial, especially when dealing with complex models with a considerable number of parameters such as ours. I was studying the best way to carry out the BMA, after probing several possibilities I saw that using Laplace approximations was perhaps the best option. Using [9], and its supplementary material as a benchmark, I tried to conduct a BMA with a simplified library of univariate joint models. Nevertheless, after filling several sheets with derivatives and thinking of an optimal and generic way to use Laplace approximation in our model (deriving likelihood and priors in general in our model), I saw that it was probably not worth it, considering that the BMA had given worse results before. Later, taking into account the use of numerical derivatives, the problem became much simpler and more feasible. However, by that time I had already burned out with the BMA approach and preferred to change strategy (using other types of machine learning related models to compare with SL).

However, it should be noted that the effort was not in vain since it allowed me to review in depth the multivariate joint model as a Bayesian model, all its different components (parameters, priors, likelihood, posteriors, etc.) which has given me a much broader knowledge of the model.

With all this in mind, one approach that I came across while studying the possibility of implementing the BMA is the 'indicator approach'. I think the latter might be more interesting to use. That is why I would like to present it briefly in this section. In addition, I have already carried out an implementation of this approach in Stan, so I know that this is feasible, although complicated in Stan by some details that I will discuss later.

For the indicator approach I have used *Getting started with Bayesian variable selection using JAGS and rjags* as a starting point. A widely used approach in Bayesian analysis to do variable selection, is to introduce an indicator random variable I_i for each covariate, and introduce these into the model in order to 'zero out' inactive covariates. The latter can be used also with BMA. In particular, in our case we can write the the multivariate joint model (2) as

$$h_i\left(t \mid \mathcal{Y}_i(t), \boldsymbol{w}_i\right) = h_0(t) \exp\left\{\boldsymbol{\gamma}^\top \boldsymbol{w}_i + \sum_{l=1}^{L} I_l f_l(t, \mathcal{Y}_{li}(t), \boldsymbol{b}_{li}, \boldsymbol{\alpha}_l)\right\}, \quad t > 0, \tag{25}$$

where $I_l, l = 1, \ldots, L$ are indicator random variables, with prior distribution *Bernoulli*(p), where $p \sim$ *Beta*(a, b), a, b known parameters. Thus, what we do is use MCMC to fit 25 and get the posterior distributions. Having the posteriors of the L indicator random variables we can compute the weights for the BMA. There are several points to be commented:

- Indicators approach is not feasible to do BMA in our case, since we want to avoid fitting the multivariate joint model. With this approach we not only adjust the multivariate model, but we also add L parameters to be taken into account, which increases the computational cost even more.

- The approach above can be useful to do a comparison with the super learning algorithm with small L. We have to take into account that in (25) we are taking into account 2^L models (all the possible combinations of models with the L different longitudinal submodels). Thus, one could expect that (25) will be the model making the best predictions. Compare the results of super learning algorithm with this model could be interesting.
- I am not sure how having the association parameter α_l (which takes into account the effect of the lth longitudinal outcome on the risk of suffering the event), can affect the new indicator random variables.

Furthermore, there are several difficulties on the implementation of (25). I have an implementation done in Stan, for a simple multivariate joint model with just $L = 2$ longitudinal outcomes. The main problem is that Stan does not allow for non-continuous random variables. Then, we cannot define a random variable with Bernoulli prior in Stan. The way to overcome this issue is via marginalization (see Stan User guide). This strategy becomes more and more complicated as L grows.

In conclusion, I may invest some more time in the indicator approach in the future. For the moment I think that exploring other models, for example machine learning-based models, that are used in the literature, to see if SL still gives better results may be more interesting to validate SL against other models.

5.2 About MFPCA-based models implementation

This last section is intended to briefly explain how the implementation of the MFPCA-based models has been carried out. To implement the models of the fourth section of this work we have used Python and R. The simulation of the data sets has been carried out via R (as in previous simulation studies). In order to carry out the implementation we have relied on [6], as well as its supplementary material and the GitHub repository provided there (`https://github.com/reylined/TransformerJM`). The code in the latter repository is pretty complete and has been useful to implement the models. Nonetheless, several adaptations have been made in order to make the code useful in our framework.

In [6] predictions are done using several landmark times. However, we are interested in predictions for an specific time interval. Moreover, although in [6] integrated Brier score is used, the metric the authors used is not the same used in [10]. Thus, adaptations have been made to calculate the same metric as in [10]. Simpson's rule has been used to compute the IBS. Furthermore, in the fourth section we have used inverse probability of censoring because it was the method implemented by the authors of [6]. Since this method is also implemented in `JMbayes2`, it was easier to use it instead of model based-weights.

Another point to keep in mind is that in order to use the same data sets both in R to carry out the super learning procedures and in Python to fit the MFPCA-based models, what we have done is to save in an RData files the 100 data sets simulated via R. These RData files can be read both in R and in Python.

References

[1] David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

[2] Clara Happ and Sonja Greven. Multivariate functional principal component analysis for data observed on different (dimensional) domains. *Journal of the American Statistical Association*, 113(522):649–659, 2018.

[3] Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18:1–12, 2018.

[4] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Kan Li and Sheng Luo. Dynamic prediction of alzheimer's disease progression using features of multiple longitudinal outcomes and time-to-event data. *Statistics in medicine*, 38(24):4804–4818, 2019.

[6] Jeffrey Lin and Sheng Luo. Deep learning for the dynamic prediction of multivariate longitudinal and survival data. *Statistics in medicine*, 41(15):2894–2907, 2022.

[7] Eric C Polley and Mark J Van der Laan. Super learner in prediction. *Statistical applications in genetics and molecular biology*, 2010.

[8] Dimitris Rizopoulos. *Joint models for longitudinal and time-to-event data: With applications in R*. CRC press, 2012.

[9] Dimitris Rizopoulos, Laura A Hatfield, Bradley P Carlin, and Johanna JM Takkenberg. Combining dynamic predictions from joint models for longitudinal and time-to-event data using bayesian model averaging. *Journal of the American Statistical Association*, 109(508):1385–1397, 2014.

[10] Dimitris Rizopoulos and Jeremy MG Taylor. Optimizing dynamic predictions from joint models using super learning. *Statistics in Medicine*, 43(7):1315–1328, 2024.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[12] Mark J Van der Laan, Eric C Polley, and Alan E Hubbard. Super learner. *Statistical applications in genetics and molecular biology*, 6(1), 2007.

[13] Fang Yao. Functional principal component analysis for longitudinal and survival data. *Statistica Sinica*, pages 965–983, 2007.

[14] Fang Yao, Hans-Georg Müller, and Jane-Ling Wang. Functional data analysis for sparse longitudinal data. *Journal of the American statistical association*, 100(470):577–590, 2005.

A Annex

The 10×10 Σ matrix used to compute the covariance matrix for the Scenario II in the first simulation study, and for both scenarios in the second simulation study is:

$$\begin{pmatrix}
1.0814 & -0.0047 & -5e-04 & 0.0029 & 0.0028 & 0.0046 & 0.0029 & 0.0019 & 0.0037 & 0.0043 \\
0.0038 & 0.2286 & -0.0017 & -0.0049 & -7e-04 & 0.0033 & -0.0024 & 1e-04 & 1e-04 & 0.0031 \\
0.0026 & 0.0024 & 1.4481 & -0.0031 & 0.0043 & -0.0018 & 0.0049 & -0.0013 & -0.0049 & -0.0042 \\
0.0039 & -0.005 & 0.0021 & 1.0228 & 0.0027 & -0.0029 & 0.0026 & -0.0016 & -0.0048 & 0.001 \\
-4e-04 & -0.0011 & 0.0014 & -0.0013 & 0.3895 & 0.0023 & 0.0048 & -0.0045 & -0.0036 & 0.0021 \\
-0.0033 & -4e-04 & -0.0011 & -0.0014 & -0.0018 & 0.7489 & -0.0028 & 0.0012 & -0.0019 & 1e-04 \\
-0.0017 & -0.0011 & 0.002 & 0.0037 & -0.0044 & 0.0023 & 1.4231 & 0.0046 & 0.0033 & 0.0022 \\
1e-04 & -0.001 & 4e-04 & 0.004 & -0.0046 & -0.0042 & -0.0035 & 0.9824 & 0 & 0.0025 \\
0.0023 & -0.0032 & -0.0027 & 0.0012 & -0.0044 & -6e-04 & 0.001 & 1e-04 & 1.2054 & -0.004 \\
0.0049 & 0.0045 & -2e-04 & -0.0037 & 0.0013 & -0.0026 & 0.0045 & -0.0035 & -0.0044 & 0.5967
\end{pmatrix}$$