survivalSL: an R Package for Predicting Survival by a Super Learner

by Camille Sabathe and Yohann Foucher

Abstract The R package **survivalSL** contains a variety of functions to construct a super learner in the presence of censored times-to-event and to evaluate its prognostic capacities. Compared to the available packages, we propose additional learners, loss functions for the parameter estimations, and user-friendly functions for evaluating prognostic capacities and predicting survival curves from new observations. We performed simulations to describe the value of our proposal. We also detailed its usage by an application in multiple sclerosis. Because machine learning is increasingly being used in predictive studies with right-censoring, we believe that our solution can be useful for a large community of data analysts, beyond this clinical application .

1 Introduction

In clinical practice, the prediction of the probability that a subject will experience an event is often of interest. For instance, for patients with multiple sclerosis under first-line treatment, the prediction of disease progression would result in the early identification of non-responders and help in deciding the switch to second-line treatment. However, the time-to-disease progression is often not observable for all subjects because of a lack of follow-up, i.e., right-censoring.

Several regressions can be used for prediction from right-censored data. The most popular is probably the proportional hazards (PH) model (Cox, 1972). The corresponding baseline hazard function can be obtained by assuming parametric distributions (Andersen et al., 1985), nonparametric estimators (Lin, 2007) or other flexible approaches such as splines (Rosenberg, 1995). Penalized PH models have also been proposed, particularly useful for high-dimensional data (Goeman, 2009). The accelerated failure times (AFT) approach also constitutes an alternative to the PH assumption (Wei, 1992).

In parallel to these regression-based methods, machine learning is increasingly used. Support-vector machines (Shivaswamy et al., 2007), neural networks (Faraggi and Simon, 1995), or even random forests (Ishwaran et al., 2008) have been successfully developed for censored time-to-event data. Ensemble learning allows us to combine regressions and algorithms by minimizing the cross-validated loss function (Breiman, 1996). SL was first proposed by van der Laan et al. (2007) and van der Laan and Dudoit (2003) from the theory of unified loss-based estimation.

Polley et al. (2011) and Polley and van der Laan (2011) extended the SL to right-censored data. Two R-based solutions are available on GitHub repositories. The **SuperLearner-Survival** repository is related to the work by Golmakani and Polley (2020). It allows us to obtain the linear predictor of a PH regression. The **survSuperLearner** package was developed by Westling (2021) with additional learners: several parametric PH models (Exponential, Weibull, log-logistic, and piecewise constant hazard), generalized additive Cox regression, and random survival forest.

In this paper, we aim to extend these solutions by i) additional learners (accelerated failure times, neural networks, penalized regressions), ii) several loss functions for the estimation of the parameters (Brier score, negative binomial log-likelihood, etc.) and iii) user-friendly S3 methods for evaluating the predictive capacities, as well as predicting survival curves from new observations (Section 1). In the second section, we propose a simulation-based study to compare the performances of our proposal with that of the alternative solutions. In the third section, we detail its usage by an application in multiple sclerosis. Finally, we discuss the strengths, weaknesses and perspectives of our solution.

2 Methods

2.1 Notations

For a subject i in a sample of N independent subjects $(i = 1, ..., N)$, we denoted the time-to-event by $T_i^\star$. Right-censoring leads the observation of $T_i = \min(T_i^\star, C_i)$, where C_i is the censoring time. Let $D_i = \mathbb{1}\{T_i^\star \leq C_i\}$ be the event indicator. The survival function at time t for a subject with the characteristics Z_i at baseline is defined by $S(t \mid Z_i) = \mathbb{P}(T_i^\star > t \mid Z_i)$.

2.2 Definition of the super learner

Let $S_{sl}(.)$ be the survival function obtained by the SL such as $S_{sl}(t \mid Z_i) = \sum_{m=1}^{M} w_m \times S_m(t \mid Z_i)$, where $S_m(.)$ is the survival function obtained by its mth learner ($m = 1, ..., M$), and $w = (w_1, ..., w_M)$ are the corresponding weights with respect to $\sum_1^M w_m = 1$ and $0 \leq w_m \leq 1$. The estimations of the learners and the weights can be obtained by respecting the following steps (Polley and van der Laan, 2010):

1. Estimate the M models or algorithms on the entire sample as usual. The procedure of estimation can be different for each model. For instance, a parametric model may be fitted by maximizing the full likelihood, a Cox regression by maximizing the partial likelihood and estimating the baseline survival function with Breslow method, or a random survival forest by maximizing the between-node survival differences according to the log-rank statistic.
2. By using the M models or algorithms estimated in the step #1 ($m = 1, ..., M$), predict the values $\hat{S}_m(u \mid Z_i)$ for each subject $i = 1, ..., N$ and for a vector of U times ($u = u_1, ..., u_U$) and stack the values in a 3-dimensional array of size ($N \times U \times M$).
3. Split the sample into a training and validation sample according to a V-fold cross-validation, i.e., V-equal size and randomly selected groups. Let $T(v)$ be the vth training sample and $V(v)$ be the corresponding validation sample ($v = 1, .., V$).
4. For the vth fold, estimate the M models or algorithms from $T(v)$. The learners based on the estimation of hyperparameters should be tuned for each of the V folds. Note that this tunning step is often based on cross-validation and should be performed for each of the V folds. Regarding the corresponding computational burden, a compromise consists of using the hyperparameters estimated at the step #1 (Le Borgne et al., 2021).
5. Compute the corresponding predicted survival rates for the individuals of the validation sample $V(v)$ and the U times.
6. Stack the predictions $\tilde{S}_m(u \mid Z_i)$ obtained in the step #5 in a 3-dimensional array of size ($N \times U \times M$).
7. Determine the loss function $\mathbb{L}(w \mid \tau)$ for quantifying the gap of the predictions $\tilde{S}_{sl}(\tau \mid Z_i) = \sum_{m=1}^{M} w_m \tilde{S}_m(\tau \mid Z_i)$ and the observations (T_i, D_i) for a prognostic up to τ. The loss functions available in the package are listed in the next subsection.
8. Estimate the weights w by minimizing the loss function for a prognostic up to a time τ:

$$\hat{w} = \arg\min_{w} \mathbb{L}(w \mid \tau)$$

9. Combine $\hat{w}_m$ with $\hat{S}_m(u \mid Z_i)$ to obtain the final super learner:

$$\hat{S}_{sl}(t \mid Z_i) = \sum_{m=1}^{M} \hat{w}_m \hat{S}_m(t \mid Z_i)$$

2.3 Available loss functions for estimating the weights

Several loss functions $\mathbb{L}(w \mid \tau)$ are available in the package. The Brier score (BS) for right-censored data and a prediction at time τ is defined by:

$$BS(w \mid \tau) = \frac{1}{N} \sum_i \left[\frac{\tilde{S}_{sl}(\tau \mid Z_i)^2 \mathbb{1}\{T_i \leq \tau, D_i = 1\}}{\hat{G}(T_i)} + \frac{(1 - \tilde{S}_{sl}(\tau \mid Z_i))^2 \mathbb{1}\{T_i > \tau\}}{\hat{G}(\tau)} \right]$$

where $\hat{G}(u) = \mathbb{P}(C_i > u)$ is the survival function of the censoring time (van Houwelingen et al., 2012). The corresponding τ-restricted integrated BS is $RIBS(w \mid \tau) = \int_0^\tau BS(u)du$. The integrated BS equals $RIBS(w \mid \max(T_i^\star))$. We also considered the negative binomial log-likelihood (BLL), which is defined for a prognostic at time τ as follows:

$$BLL(w \mid \tau) = \frac{1}{N} \sum_i \left[\frac{\log(1 - \tilde{S}_{sl}(\tau \mid Z_i)) \mathbb{1}\{T_i \leq \tau, D_i = 1\}}{\hat{G}(T_i)} + \frac{\log(\tilde{S}_{sl}(\tau \mid Z_i)) \mathbb{1}\{T_i > \tau\}}{\hat{G}(\tau)} \right]$$

We also proposed the corresponding integrated and τ-restricted integrated versions. Additionally, we implemented the concordance index (CI) as defined by Uno et al. (2011):

$$CI(w \mid \tau) = \frac{\sum_i \sum_j \mathbb{1}\{T_j < T_i, \tilde{S}_{sl}(\tau \mid Z_j) > \tilde{S}_{sl}(\tau \mid Z_i)\} D_j}{\sum_i \sum_j \mathbb{1}\{T_j < T_i\} D_j}$$

Finally, we considered the area under the time-dependent ROC curve, i.e., the sensitivity (SE) against one minus the specificity (SP) at each threshold η (Hung and Chiang, 2010):

$$SE_\eta(w \mid \tau) = \frac{\sum_i D_i \mathbb{1}\{T_i \leq \tau, \tilde{S}_{sl}(\tau \mid Z_i) > \eta\}/(1-\hat{G}(T_i))}{\sum_i D_i \mathbb{1}\{T_i \leq \tau\}/(1-\hat{G}(T_i))}$$

$$SP_\eta(w \mid \tau) = \frac{\sum_i \mathbb{1}\{T_i > \tau, \tilde{S}_{sl}(\tau \mid Z_i) \leq \eta\}}{\sum_i \mathbb{1}\{T_i > \tau\}}$$

2.4 Available learners

Several models or algorithms are proposed in the **survivalSL** package:

- `Parametric AFT models.` They can be considered with Weibull, Gamma or generalized Gamma distributions (**flexsurv** package). The parameters are estimated by likelihood maximization.
- `Parametric PH models.` They can be considered with Exponential or Gompertz distributions (**flexsurv** package). The parameters are estimated by likelihood maximization.
- `Spline-based PH models.` We considered the spline-based survival model proposed by Royston and Parmar (2002). To estimate the baseline distribution, it involves one hyperparameter: the number of internal knots of the natural cubic spline, with a default grid search $k = \{1, 2, 3, 4\}$. The parameters are estimated by likelihood maximization (**flexsurv** package).
- `Semi-parametric PH models.` It corresponds to a PH model (**coxph** function of the **survival** package) with a non-parametric baseline hazard function estimated by using the Breslow estimator (Lin, 2007). This approach can be used with or without a forward selection of covariates by using the AIC.
- `Penalized PH models.` For high-dimensional data and/or covariate selection, three penalties can be used for the previous semiparametric PH model: Lasso, Ridge, or Elastic-Net (**glmnet** package). The quantitative covariates are transformed with B-splines to relax the log-linear assumption. The hyperparameters λ of the Lasso regression allows for selection of the covariates, while it allows for the shrinkage of the regression coefficients in the Ridge regression. The Elastic-Net allows the two penalties. By default, we implemented the grid search proposed by Simon et al. (2011) for λ and $\{0.1, 0.2, ..., 0.9\}$ for α. The Lasso penalization corresponds to $\alpha = 0$, while the Ridge penalization corresponds to $\alpha = 1$.
- `Random survival forests.` This ensemble tree method extends Breiman's random forest framework to right-censored data (Ishwaran et al., 2008). It allows estimation of the cumulative hazard function in each node by using the nonparametric estimator of survival proposed by Aalen (1978). It involves three hyperparameters (**randomSRC** package): number of covariates to test for splitting the node, minimal number of events per terminal node, and number of trees. The default grid search is `mtry` = $\{1, 2, ..., 2 + 0.5 \times$ number of covariates$\}$, `nodesize` = $\{2, 4, 6, 10, 20, 30, 50, 100\}$ and `ntree` = 500, respectively.
- `Survival neural networks.` We implemented the method proposed by Biganzoli et al. (1998). It allows using the **nnet** package for estimating a feed forward neural network model for partial logistic regression. It involves four hyperparameters: the length of the time intervals, the number of units in the hidden layer, the weight decay, the maximum number of iterations, and the maximum allowable number of weights. The optimal combination is estimated by cross-validation and the previous possible loss functions. The default grid search is composed by `inter` = 1, `size` = $\{2, 4, 6, 8, 10\}$, `decay` = $\{0.001, 0.01, 0.02, 0.05\}$, `maxit` = 100, and `MaxNWts` = 10000, respectively.

3 Simulations

3.1 Design

We simulated 1000 data sets for each scenario. The times-to-event were generated from Weibull distributions with PH assumption. The censoring times were generated from uniform distributions to obtain a 40% censoring rate. We studied two sample sizes for learning (200 and 500), while the validation samples were composed of 500 subjects. We proposed two contrasting scenarios (Figure **??** in Appendix):

- A simple scenario with 6 independent covariates, 2 were continuous and 4 qualitative. Among them, 1 continuous and 2 qualitative covariates were associated with the times-to-event distribution without any log-linearity issue.

- A complex scenario with 23 correlated covariates: 12 continuous covariates (several nonlinear relationships, two effects were step functions, two were quadratic functions, five were log-linear), 11 qualitative covariates and 1 interaction.

We compared our proposed SL with each included learner, the perfectly specified PH model (i.e., the model used to generate the times-to-event, the only estimation consisting of the regression coefficients), and the SL obtained by the **survSuperLearner** package (Westling, 2021). We decided to use the package proposed by Westling (2021) as a comparator because it includes additional learners compared to the functions proposed by Golmakani and Polley (2020). The SL weights were estimated by minimizing the integrated BS (IBS) for all the methods.

In the SL proposed by Westling (2021), we included the 5 possible learners: the PH model with the Breslow estimator, the Exponential PH model, the random survival forests, the Lasso PH model and the PH model with the univariate selection of covariates ($p < 0.05$). We used the default grid of the hyperparameters.

In our proposed SL, we included the 7 possible learners: the PH model with the Breslow estimator, the same model with forward selection based on AIC minimization, the Elastic-Net PH model with B-splines on quantitative covariates, the random survival forest, the Exponential PH model, the Gamma distribution-based AFT model, and the survival neural network. Note that we did not consider the Weibull PH model since it was used for data generation. We used the default grid of the hyperparameters.

3.2 Results of the simple scenario

Figure 1 presents the distributions of the IBS over the learning (on the left) and validation samples (on the right). The related `R` code is available in Appendix. Consider the situation where $N = 200$ for learning and $N = 500$ for validation (the first two plots on the top). The perfectly specified model had the lower IBS. This result was expected since it constitutes the gold standard to reach by the other methods. The Elastic-Net PH regression outperformed all the other approaches in the learning samples. This result explained the important weight of this method, approximately 25% in our SL, as illustrated in Figure 5 in Appendix. Nevertheless, this apparent result was not observed in the validation samples, illustrating overfitting.

In both the learning and the validation samples, our proposed SL, the PH model with the Breslow estimator, and the same model with forward selection based on the AIC had the best performances, higher than that of the SL proposed by Westling (2021).

The survival neural network and the PH model with an Exponential distribution were associated with larger values of the IBS in both the learning and the validation samples. Comparatively, the results were closed across the other methods. In the validation samples, the highest variability of the IBS was observed for the survival neural network.

When $N = 500$ in both the learning and the validation samples (the two bottom plots of Figure 1), the results were close. However, one can note that the means and variances of the IBS were lower.

Results of the complex scenario

The related `R` code is available in Appendix. As illustrated in Figure 2, regarding the results in the validation samples, our proposed SL and the Elastic-Net PH regression outperformed the other models/algorithms. More precisely, the mean of the IBS was lower for the Elastic-Net PH model for a learning sample size of 200. The value of our proposed SL was close. When $N = 500$ for learning, the mean of the IBS was lower for our SL, but that of the Elastic-Net PH model was close. Nevertheless, regardless of the sample size for learning, our SL was associated with lower variance in the IBS.

Among the other differences compared to the simple context, the random survival forests and the survival neural networks performed better than the other model-based predictors, except the Elastic-Net PH model.

In both the learning and the validation samples, and regardless the sample size for learning, our proposed SL had slightly higher performances compared to the results of the SL proposed by Westling (2021).

3.3 Running times

We further explored the time required to estimate a SL depending on the sample size and the number of predictors. For this purpose, we simulated data according to the complex scenario (Figure **??** in Appendix). We used a 20-fold cross-validation, tested sample sizes from 500 to 3500, and reduced

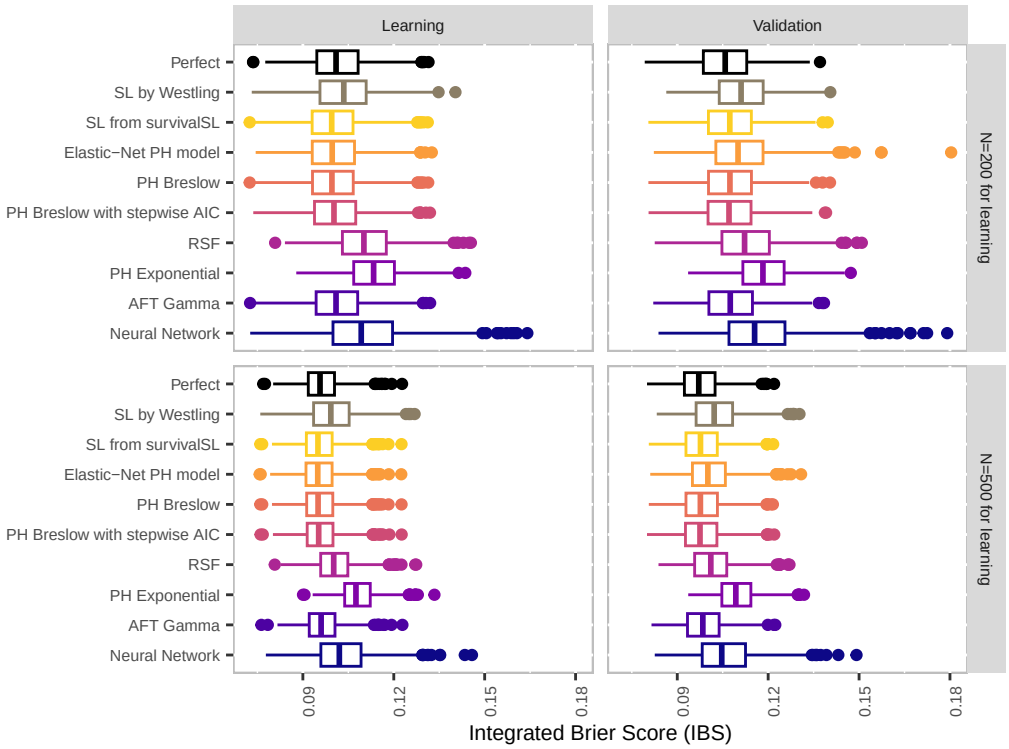

Figure 1: Simulation results in the simple context.

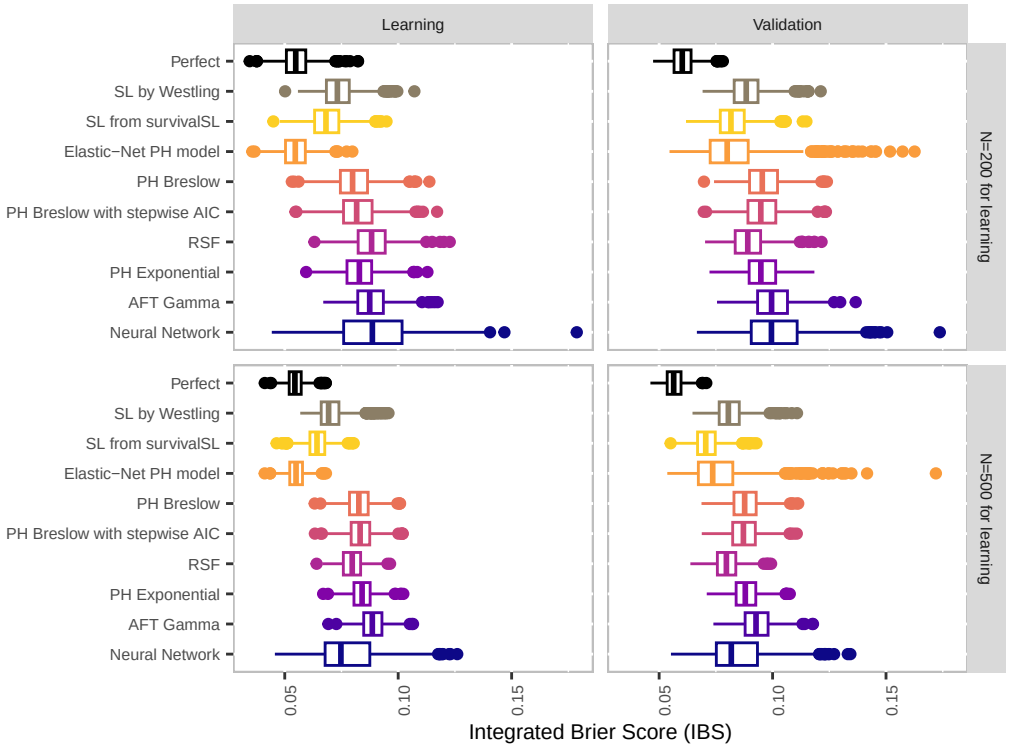

Figure 2: Simulation results in the complex context.

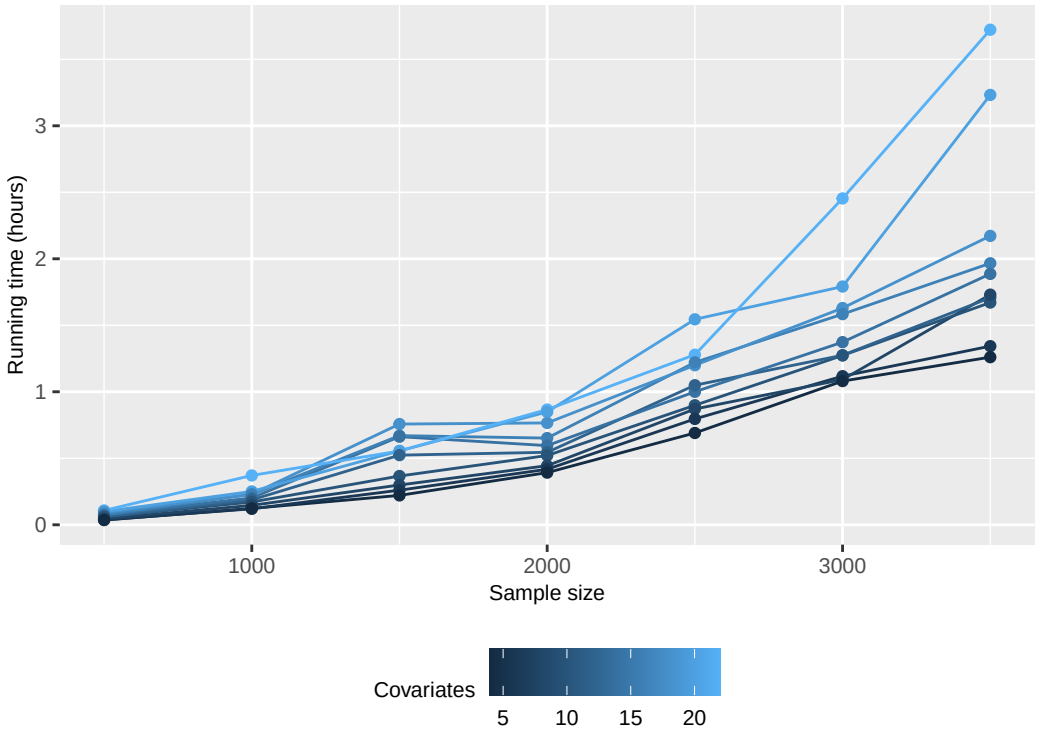

Figure 3: Running times according to the sample size and the number of covariates. Results obtained for a MacBook Pro 2.6 GHz Intel Core i7 6 cores.

the number of predictors from 22 to 4. The results are illustrated by Figure 3. If we considered 20 predictors, the running times increased exponentially from 0.10 ($N = 500$) to 3.23 hours ($N = 3500$). For 10 predictors, the times increased from 0.06 to 1.67 hours.

4 Usage

4.1 The main functions of the package

The `survivalSL` function fits the SL. It is mainly based on the following arguments:

- `formula`: A formula object, with the response on the left, and the predictors on the right. The response must be a survival object as returned by the `Surv` function..
- `methods`: The names of the learners included in the SL. At least two models/algorithms must be included. The complete list of candidates is described in Table 1.
- `data`: The data frame in which to look for the variables related to the follow-up time, the event indicator, and the covariates.
- `metric`: The loss function used to estimate the weights of the learners.
- `pro.time`: The prognostic time used for the Brier score, the negative binomial log-likelihood or the corresponding restricted versions.
- `cv`: The number of cross-validated samples to estimate the weights of the learners.
- `param.tune`: Optional list to define the grid search of the hyperparameter(s) or to set the value(s) of the hyperparameter(s). If `NULL`, the default grid is used.
- `seed`: A numeric value for random seed for reproducibility. The default is `NULL`.

Note that a method based on hyperparameters can be used several times with different values of hyperparameters. When the hyperparameters are specified by the user, the argument `param.tune` must be a list with their value(s) and name(s). For a user-friendly manipulation of the resulting `sltime` object, we proposed several S3 methods listed in Table 2. They allow for evaluating the calibration and discrimination capacities, or predicting survival curves. These functions can be applied to new data sets. Similar S3 methods are available for other learners (`LIB_AFTgamma`, `LIB_COXlasso`, etc.).

Table 1: List of possible learners (#: Breslow's estimation of the baseline hazard function). The last column lists the learners included in the survSuperLearner proposed by Westling (2021). This package additionally considers generalized additive Cox regression and piecewise constant hazard regression.

Name	Description	Westling
LIB_COXall	Proportional hazards (PH) model with all covariates (#)	Yes
LIB_COXaic	PH model with covariate selection by AIC minimization (#)	No
LIB_COXen	PH model with B-spline for the quantitative covariates and Elastic-Net penalization (#, hyperparameters: alpha and lambda)	Yes
LIB_COXlasso	PH model with B-spline for the quantitative covariates and Lasso penalization (#, hyperparameter: lambda)	Yes
LIB_COXridge	PH model with B-spline for the quantitative covariates and Ridge penalization (#, hyperparameter: lambda)	Yes
LIB_AFTgamma	Accelerated failure times (AFT) model with Gamma distribution	No
LIB_AFTggamma	AFT model with generalized Gamma distribution	No
LIB_AFTllogis	AFT model with log-logistic distribution	No
LIB_AFTweibull	AFT model with Weibull distribution	Yes
LIB_PHexponential	Parametric PH model with Exponential distribution	Yes
LIB_PHgompertz	Parametric PH model with Gompertz distribution	Yes
LIB_PHspline	PH model with natural cubic spline as baseline distribution (hyperparameters: k)	No
LIB_RSF	Random survival forest (hyperparameters: nodesize, mtry and ntree)	Yes
LIB_PLANN	One-layer survival neural network (hyperparameters: n.nodes, decay, batch.size and epochs)	No

Table 2: List of S3 functions applicable to an sltime object

Function	Description
plot.sltime	To obtain a calibration plot.
predict.sltime	To predict the survival probabilities from the SL and the included learners.
summary.sltime	To obtain metrics describing the prognostic capacities of the SL and the included learners.
print.sltime	To print the learners and their weights

4.2 Application to multiple sclerosis

This section is not intended to propose guidelines for data analysis. It only consists of illustrating the use of the **survivalSL** package. Beside the present application, a recent work proposed recommendations for implementing such a SL (Phillips et al., 2023).

The following application is related to the prediction of the time to disease progression from prognostic factors collected one year after the initiation of a first-line treatment, which is the baseline of the cohort ($T^\star = 0$). We also compared the prognostic capacities of the obtained SL with those of the modified Rio score (Sormani et al., 2013).

`Data description.` We detailed the implementation of the previous function with an application based on the OFSEP cohort, which was composed of 1300 simulated patients with multiple sclerosis. The simulations were performed from the observed cohort to obtain the present shared data set which complies with the General Data Protection Regulation of the European Union. More precisely, the prognostic factors were first simulated one by one, i.e., using the factors already simulated to obtain the next one. These simulations were based on linear or logistic models for continuous or categorical factors, respectively. Finally, the time to disease progression was simulated by a Weibull PH model, which explains why it was not considered among the learners.

```
library("survivalSL")
data(dataOFSEP); head(dataOFSEP)
```

```
#>        time event age duration period gender relapse edss t1 t2 rio
#> 1 2.1195051     1  34     1113      0      1       1  low  0  0   1
#> 2 0.8160995     1  37     1296      0      1       1 high 1+  0   1
#> 3 1.9709546     1  33      995      1      1       0  low  0  0   0
#> 4 2.5881311     1  35      858      1      1       0  low  0  0   0
#> 5 1.4726224     1  31      759      1      0      2+ miss 1+  0   2
#> 6 1.6970962     1  40     1642      1      1       0 high  0  0   0
```

The first two columns of the `dataOFSEP` table correspond to the time to disease progression (`time`) and the `event` indicator equals 1 for patients with events and 0 for right-censored patients. The time to disease progression was defined by the minimum among the time to first relapse, the time to increase of Expanded Disability Status Scale (EDSS), and the time to switch for inefficacy. The EDSS ranges from 0 to 10 (the higher levels of disability) and is based on an examination by a neurologist.

The other columns were the available predictors one year after the initiation of first-line therapy. The third and fourth columns were two quantitative predictors: patient `age` (in years) and disease `duration` (in days). The other columns were qualitative predictors: calendar `period` (1 if between 2014 and 2018, and 0 otherwise), `gender` (1 if women), diagnosis of `relapse` since treatment initiation (1 if at least one event, and 0 otherwise), and EDSS level (`miss` if missing, `low` if level between 0 and 2, and `high` otherwise). The variables `t1` and `t2` were related to the lesions observed from magnetic resonance imaging, namely, the new gadolinium-enhancing T1 lesion (`miss` if missing, `1+` if at least one lesion, and `0` otherwise) and the new T2 lesion (1 if at least one lesion, and 0 otherwise), respectively.

The last column corresponded to the modified Rio score, i.e., the sum of the relapses (0 for none, 1 for one event, and 2 for at least 2 events) and the new T2 lesion (0 for none, and 1 for at least one lesion) observed since the treatment initiation (Sormani et al., 2013). We transformed the categorical predictors into binary variables.

```
dataOFSEP$relapse.1 <- 1*(dataOFSEP$relapse=="1")
dataOFSEP$relapse.2 <- 1*(dataOFSEP$relapse=="2+")
dataOFSEP$edss.1 <- 1*(dataOFSEP$edss=="low")
dataOFSEP$edss.2 <- 1*(dataOFSEP$edss=="high")
dataOFSEP$t1.1 <- 1*(dataOFSEP$t1=="0")
dataOFSEP$t1.2 <- 1*(dataOFSEP$t1=="1+")
```

`Super learner training.` We randomly selected two-thirds of the sample to train the SL, and the other third was further used for validation in the next subsection. We set the seed to ensure the results replicability.

```
set.seed(117)
dataOFSEP$train <- 1*rbinom(n=dim(dataOFSEP)[1], size=1, prob=2/3)
dataTRAIN <- dataOFSEP[dataOFSEP$train==1,]
dataVALID <- dataOFSEP[dataOFSEP$train==0,]
```

We fitted the SL with the following learners: PH model with Gompertz baseline hazard function, AFT model with generalized Gamma distribution, Elastic-Net PH model with nonparametric baseline hazard function (Breslow estimator) and B-spline transformations of the quantitative variables, and random survival forest. We used the default grids to estimate the hyperparameters. The weights were estimated to minimize the concordance index at 2 years with a 30-fold cross-validation. Note that 851 patients were included in the training sample, and 651 events were observed. In such a situation, Phillips et al. (2023) recommended at least 20 folds.

```
.f <- Surv(time, event) ~ age + duration + period + gender +
   relapse.1 + relapse.2 + edss.1 + edss.2 + t1.1 + t1.2
sl1 <- survivalSL(formula=.f, metric="uno_ci",  data=dataTRAIN,
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, show_progress=FALSE, seed=117)
print(sl1, digits=4)

#> The contribution of the learners:
#>
#>         leaners weights
#> 1     LIB_COXen  0.1775
#> 2  LIB_PHspline  0.2519
#> 3 LIB_AFTggamma  0.3039
#> 4       LIB_RSF  0.2667
#>
#> Minimum of the 30-fold CV of the metric uno_ci:0.6851.
```

Moreover, the package allows the user to propose a more precise grid for the hyperparameter-based methods. For instance for the spline-based PH model, the user may precise his/her own grid search:

```
.tune <- vector("list",4)
.tune[[2]] <- list(k=1:6)
sl2 <- survivalSL(formula=.f, metric="uno_ci",  data=dataTRAIN,
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, param.tune=.tune, show_progress=FALSE, seed=117)
print(sl2, digits=4)

#> The contribution of the learners:
#>
#>         leaners weights
#> 1     LIB_COXen  0.2486
#> 2  LIB_PHspline  0.3340
#> 3 LIB_AFTggamma  0.1773
#> 4       LIB_RSF  0.2401
#>
#> Minimum of the 30-fold CV of the metric uno_ci:0.684.
```

On a MacBook Pro 2.6 GHz Intel Core i7 6 cores, the running times to estimate `sl1` and `sl2` were 11.0 and 13.2 minutes, respectively. Note that the optional argument related to the definition of the grid may be used to estimate the hyperparameters from the entire sample and use the related values in each fold of the steps #4 and #5 of the super learner estimation. This strategy may be associated with overfitting, but an important decrease in the computation time and the fluctuation of the results. Indeed, regarding the random allocation of the individuals in the folds at the steps #1 (for estimating the learners from the entire training sample) and #3 (for estimating the prediction matrix includes in the loss function), the results depend on the generating pseudo random numbers. To illustrate this fluctuation, the `sl2` estimation was performed for several seeds in Appendix. The weights of the learners significantly varied but the predictive performances of the super learners were stable.

`Super learner validation.` We further studied the prognostic capacities of this algorithm, always up to 2 years. First, we observed that the metrics between the training and validation samples were close, illustrating the absence of overfitting.

```
rbind(train = summary(sl2, method="sl", pro.time=2, digits=4)$metrics,
      test =  summary(sl2,  newdata=dataVALID, method="sl",
                      pro.time=2, digits=4)$metrics)
```

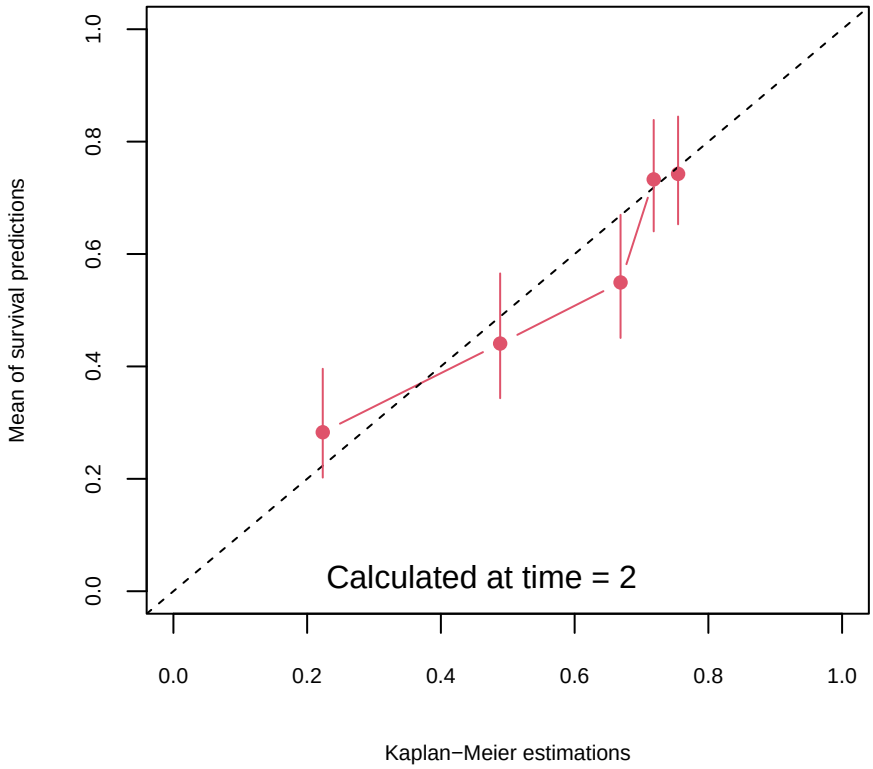

Figure 4: Calibration plot at 2 years for the validation sample.

```
#>         p_ci uno_ci    auc     bs    ibs   ribs    bll ibll ribll        ll
#> train 0.7463 0.7444 0.8056 0.1838 0.0813 0.0840 0.5487  NaN   NaN -465.7561
#> test  0.6737 0.6713 0.7036 0.2170 0.0924 0.0969 0.6232  NaN   NaN -595.7444
```

The user may also want to evaluate the prognostic capacities of the Elastic-Net PH model since it mainly contributes to the SL. It illustrates a marginal increase of the prognostic capacities related to the SL:

```
rbind(train = summary(sl2, method="LIB_COXen", pro.time=2, digits=4)$metrics,
      test =  summary(sl2,newdata=dataVALID, method="LIB_COXen",
                      pro.time=2, digits=4)$metrics)
```

```
#>         p_ci uno_ci    auc     bs    ibs   ribs    bll   ibll  ribll        ll
#> train 0.6782 0.6762 0.7204 0.1990 0.0875 0.0903 0.5843 0.2780 0.2921 -566.8255
#> test  0.6620 0.6588 0.6874 0.2201 0.0929 0.0974 0.6296 0.2952 0.3146 -664.7764
```

As illustrated by Figure 4, which is simply obtained by applying the `plot.sltime` function to the `sl2` object, the calibration of the SL was acceptable for patients of the validation sample. It represents the observed survival and the related 95% confidence intervals, which are respectively obtained by the Kaplan and Meier estimator and the Greenwood formula, against the mean of the predicted values, for individuals stratified into groups of the same size according to the percentiles of the predicted values. The identity line is usually included for reference.

```
plot(sl2, newdata=dataVALID, cex.lab=0.70,
     cex.axis=0.70, n.groups=5, pro.time=2, col=2)
```

The `predict.sltime` function returns predicted survival probabilities. With this function, one can also investigate the calibration by comparing the mean of the predictions of the patients included in the validation sample and the observed relapse-free survival, as illustrated in Figure 6 (Appendix). The SL did not improve the discrimination capacities offered by the modified Rio score for a prognostic up to 2 years after the treatment initiation (Figure 7 in Appendix).

5 Conclusion

The present paper introduced the **survivalSL** package to estimate a SL from right-censored data, to evaluate its prognostic capacities, and to predict survival curves for new individuals. The solution allows a larger set of learners (models or algorithms) and several loss functions to estimate their contributions. The user can define a grid for estimating the hyperparameters by cross-validation, or even fix their values.

The simulation study illustrated the SL performances. More precisely, in a simple context (several independent covariates, respect of the log-linearity assumption, no interaction), its performances are equivalent to more simple PH models. In contrast, in a complex context, the performances of the simple PH models decreased, and the SL resulted in the best performances: both the means and the variances of the loss functions were comparatively small relative to other methods considered.

We illustrated the usage of the package for predicting the non-response of patients with multiple sclerosis treated with a first-line therapy. The S3 methods allow a simple evaluation of the prognostic capacities, which can be compared with those of existing scoring systems. In this simple application, comparable to the simple scenario of the simulation study, the SL did not result in higher prognostic capacities compared to the Elastic-net PH model or even the existing modified Rio score.

As machine learning is increasingly being used in predictive studies, we believe that our proposal will be useful for a large community of data analysts. Nevertheless, we can point to three limitations. Firstly, even if we considered many learners, the current version of the package does not allow the users to include their own algorithms or models. An important perspective of development is to integrate this functionality. Secondly, one can also note that the estimation of learners and their weights assume non-informative censoring. Introducing the inverse probability of censoring weighting (IPCW) method, to correct for dependent censoring, would be of great interest, but challenging for some learners such as neural networks or random survival forests.

6 Acknowledgments

We would like to thank David Laplaud, neurologist and specialist in multiple sclerosis, for his clinical advice. The OFSEP cohort is supported by a grant provided by the French National Agency for Research, within the framework of the Investments for Future (ANR-10-COHO-002), and by the Eugène Devic EDMUS Foundation against multiple sclerosis and the ARSEP Foundation.

7 Bug reports

To improve future versions of the package, you can report errors at the following address:

`https://github.com/chupverse/survivalSL/issues`

8 Supplementary materials

8.1 R code to perform the simulations of the simple scenario

```
library(survivalSL)

# definition of the parameters related to the simulated data
n.valid <- 500 # sample size for validation
n.learn <- 500 # sample size for training

n <- n.valid + n.learn # overall sample size

max.time <- 50 # maximum follow-up time

mean.x <- 0; sd.x <- 1 # normal distribution of the quantitative predictors
proba.x <- .5 # proportion of the binary predictors

a <- 2; b <- .05 # Weibull baseline distribution of the PH model
beta <- c(log(1.8), log(1.8), log(1.3), 0, 0, 0) # regression coefficients

# simulation of  the training and validation samples
x1 <- rnorm(n, mean.x, sd.x)
x2 <- rbinom(n, 1, proba.x)
x3 <- rbinom(n, 1, proba.x)
x4 <- rnorm(n, mean.x, sd.x)
x5 <- rbinom(n, 1, proba.x)
x6 <- rbinom(n, 1, proba.x)
x <- cbind(x1, x2, x3, x4, x5, x6) # matrix of the potential predictors

u <- runif(n, 0, 1)
times <- 1/b*((-exp(-1*(x %*% beta))*(log(1-u)))**(1/a)) # time to event

censoring <- runif(n, min=0, max=max.time)

status <- ifelse(times <= censoring, 1, 0) # event status
obs.times <- ifelse(times <= censoring, times, censoring) # follow-up times

data <- cbind(obs.times, status, as.data.frame(x))

data.simul <- list(data[1:n.valid,], data[(n.valid+1):n,])

# definition of the grid search related to the hyperparameters

slres <- survivalSL(
  formula=Surv(obs.times, status) ~ x1 + x2 + x3 + x4 + x5 + x6,
  methods=c("LIB_COXen", "LIB_COXall", "LIB_RSF", "LIB_PLANN",
            "LIB_AFTgamma", "LIB_PHexponential"),
  metric="ibs",  data=data.simul[[1]], show_progress=FALSE, cv=10)

# loss functions estimated from training sample (Figure 1)
summary(slres)

# from validation sample (Figure 1)
summary(slres, newdata=data.simul[[2]])
```

8.2 R code to perform the simulations of the complex scenario

```
library(survivalSL)

# definition of the parameters related to the simulated data
n.valid <- 500 # sample size for validation
n.learn <- 200 # sample size for training
n <- n.valid + n.learn # overall sample size

max.time <- 50 # maximum follow-up time

b0.t <- (-0.4) # intercept related to the predictors' distribution
b1.t <- log(2) # slope related to the predictors' distribution

a <- 2; b <- .05 # Weibull baseline distribution of the PH model
b1.o <- 0.69 # regression coefficients of the PH model

# simulation of  the training and validation samples
.x1 <- rnorm(n, 0, 1)
.x2 <- rnorm(n, b0.t + b1.t * .x1, 1)
.x3 <- rnorm(n, b0.t - b1.t * .x1 - b1.t * .x2, 1)
.x4 <- rnorm(n, b0.t + b1.t * .x3, 1)
.x5 <- rnorm(n, 0, 1)
.x6 <- 1 * (rnorm(n, 0, 1) > 0.66)
.x7 <- 1 * (rnorm(n, b0.t - b1.t * .x5, 1) > (-0.40))
.x8 <- rnorm(n, b0.t - b1.t * .x6, 1)
.x9 <- 1 * (rnorm(n, b0.t + b1.t * .x7, 1) > (-0.80))
.x10 <- rnorm(n, b0.t + b1.t * .x8, 1)
.x11 <- rnorm(n, 0, 1)
.x12 <- 1 * (rnorm(n, b0.t + b1.t * .x9, 1) > (0.84))
.x13 <- 1 * (rnorm(n, b0.t + b1.t * .x10, 1) > (-0.09))
.x14 <- rnorm(n, b0.t - b1.t * .x12 - b1.t * .x11, 1)
.x15 <- rnorm(n, b0.t - b1.t * .x12, 1)
.x16 <- 1 * (rnorm(n, 0, 1) > (-0.66))
.x17 <- 1 * (rnorm(n, b0.t - b1.t * .x16, 1) > (-0.92))
.x18 <- rnorm(n, 0, 1)
.x19 <- 1 * (rnorm(n, 0, 1) > (0.66))
.x20 <- 1 * (rnorm(n, 0, 1) > (0.66))
.x21 <- rnorm(n, 0, 1)
.x22 <- 1 * (rnorm(n, 0, 1) > (0.66))

data.obs <- data.frame(x1=.x1, x2=.x2, x3=.x3, x4=.x4, x5=.x5, x6=.x6,
    x7=.x7, x8=.x8, x9=.x9, x10=.x10, x11=.x11, x12=.x12, x13=.x13,
    x14=.x14, x15=.x15, x16=.x16, x17=.x17, x18=.x18, x19=.x19,
    x20=.x20, x21=.x21, x22=.x22)

bx <- b0.t + b1.t*data.obs$x1 - b1.t*data.obs$x3 + b1.t*data.obs$x5 -
  b1.t*data.obs$x7 + b1.t*data.obs$x9 - b1.t*data.obs$x11 +
  b1.t*data.obs$x13 - b1.t*data.obs$x15 - b1.t*data.obs$x17 +
  b1.t*data.obs$x19 - b1.t*data.obs$x21

pr.t <- (exp(bx) / (1 + exp(bx)))
data.obs$t.obs <- rbinom(n, 1, prob = pr.t)

bx <- b1.o*(data.obs$x2>-0.40) - b1.o*data.obs$x3 +
  b1.o*0.5*(data.obs$x3^2) + b1.o*data.obs$x6 + b1.o*data.obs$x7 +
  b1.o*data.obs$x10 + b1.o*0.5*(data.obs$x11^2) - b1.o*data.obs$x14 -
  b1.o*(data.obs$x15>-0.57) + b1.o*data.obs$x18 + b1.o*data.obs$x19 +
  b1.o*data.obs$t.obs + b1.o*0.5*data.obs$t.obs*data.obs$x18

u <- runif(n,0,1)
times <- 1/b*((-exp(-bx)*(log(1-u)))**(1/a))

censoring <- runif(n, min=0, max=max.time)
```

```
status <- ifelse(times <= censoring, 1, 0)
obs.time <- ifelse(times <= censoring, times, censoring)

data <- cbind(obs.time, status, data.obs)

data.simul <- list(data[1:n.valid,], data[(n.valid+1):n,])

slres <- survivalSL(
  formula=Surv(obs.time, status) ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
  x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 + x22,
  data=data.simul[[1]],  metric="ibs",  cv=10, show_progress=FALSE,
  methods=c("LIB_COXen", "LIB_COXall", "LIB_RSF", "LIB_PLANN",
            "LIB_AFTgamma", "LIB_PHexponential"))

# loss functions estimated from training sample (Figure 2)
summary(slres)

# from validation sample (Figure 2)
summary(slres, newdata=data.simul[[2]])
```

8.3 Fluctuation of the results according to the seed

We estimate three additional SL with 3 alternative initializations of the pseudo-random number generator.

```
sl2.1 <- survivalSL(formula=.f, data=dataTRAIN, metric="uno_ci",
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, param.tune=.tune, show_progress=FALSE, seed=118)

sl2.2 <- survivalSL(formula=.f, data=dataTRAIN, metric="uno_ci",
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, param.tune=.tune, show_progress=FALSE, seed=119)

sl2.3 <- survivalSL(formula=.f, data=dataTRAIN, metric="uno_ci",
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, param.tune=.tune, show_progress=FALSE, seed=120)

rbind(
  seed.121 = summary(sl2,   newdata=dataVALID, method="sl", pro.time=2, digits=4),
  seed.122 = summary(sl2.1, newdata=dataVALID, method="sl", pro.time=2, digits=4),
  seed.123 = summary(sl2.2, newdata=dataVALID, method="sl", pro.time=2, digits=4),
  seed.124 = summary(sl2.3, newdata=dataVALID, method="sl", pro.time=2, digits=4))

#>          metrics       method pro.time ROC.precision
#> seed.121 data.frame,10 "sl"   2        numeric,99
#> seed.122 data.frame,10 "sl"   2        numeric,99
#> seed.123 data.frame,10 "sl"   2        numeric,99
#> seed.124 data.frame,10 "sl"   2        numeric,99

rbind(
  seed.121 = sl2$weights$values,
  seed.122 = sl2.1$weights$values,
  seed.123 = sl2.2$weights$values,
  seed.124 = sl2.3$weights$values)

#>               [,1]      [,2]       [,3]      [,4]
#> seed.121 0.2486286 0.3339517 0.17734499 0.2400747
#> seed.122 0.2713981 0.3002995 0.17599652 0.2523059
#> seed.123 0.1721614 0.5182388 0.08864614 0.2209536
#> seed.124 0.2427707 0.2397718 0.27035706 0.2471004
```

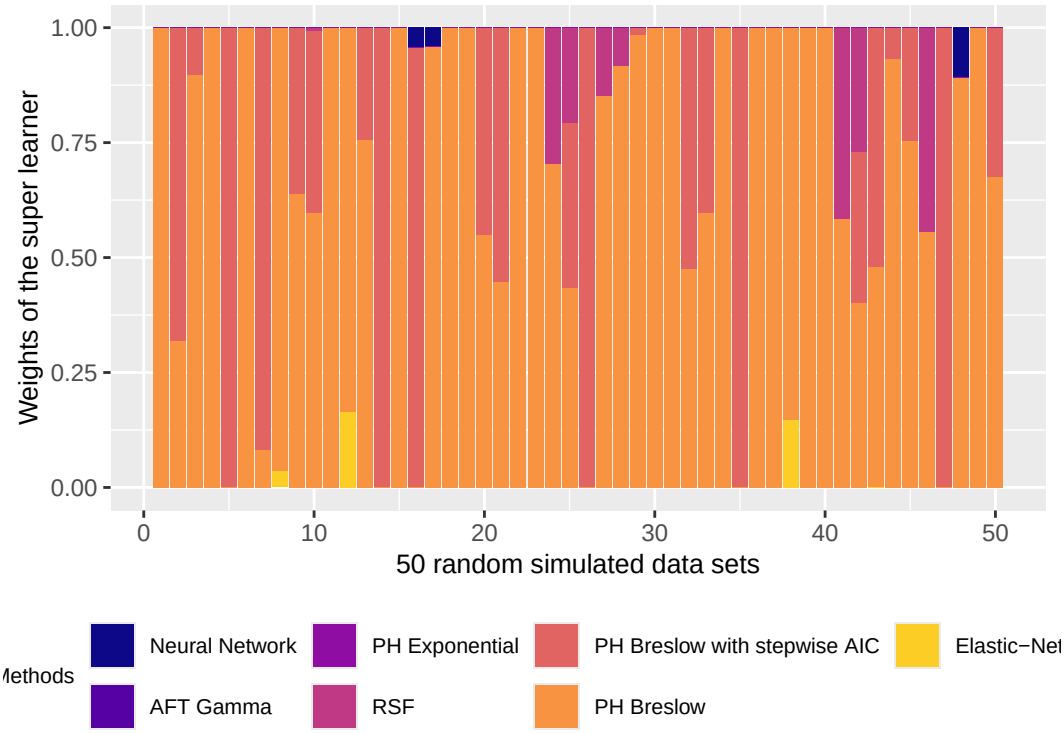

Figure 5: Weight distribution among 50 randomly estimated super learners.

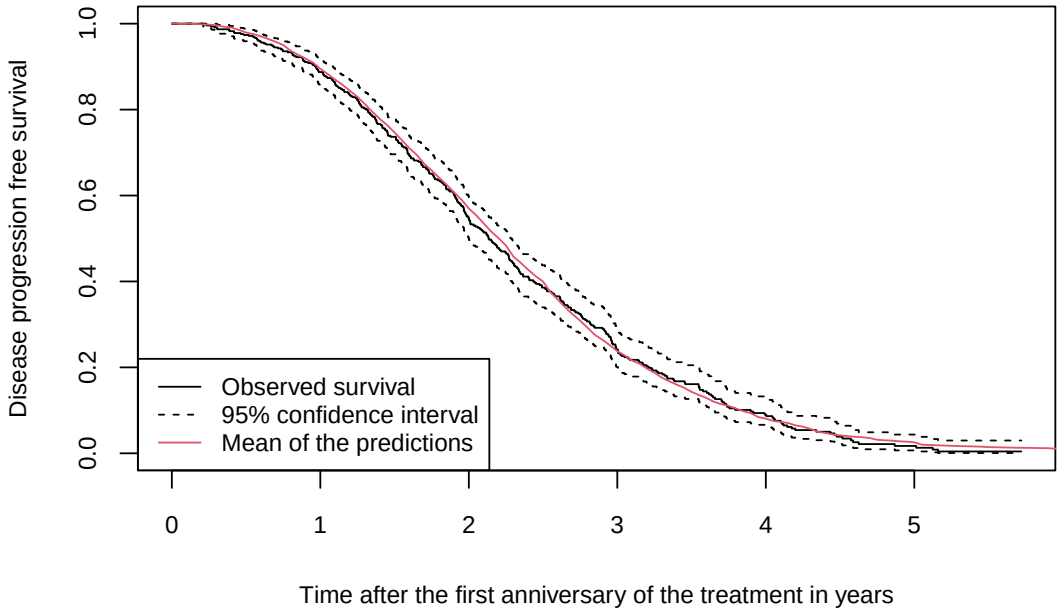

Figure 6: Observed relapse-free survival of the OFSEP validation sample (Kaplan-Meier estimator) compared to the mean of the individual predictions.

```
plot(survfit(Surv(time, event) ~ 1, data = dataVALID),
     ylab="Disease progression free survival",
     xlab="Time after the first anniversary of the treatment in years",
     cex.lab = 0.8, cex.axis=0.8)

.pred.matrix <- predict(sl2, newdata=dataVALID,
                        newtimes = seq(0, 6, by=0.05))$predictions$sl

.pred <- apply(.pred.matrix, MARGIN=2, FUN="mean")

lines(x=seq(0, 6, by=0.05), y=.pred, col=2)

legend("bottomleft", c("Observed survival", "95% confidence interval",
        "Mean of the predictions"), col=c(1, 1, 2), lty=c(1,2,1), cex=0.8)
```

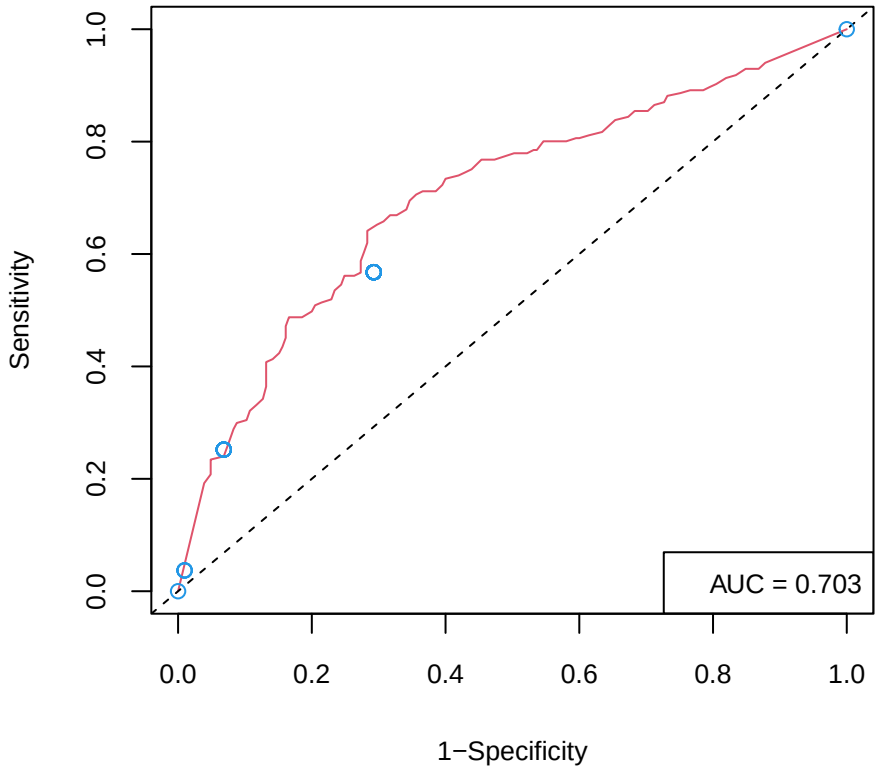

Figure 7: ROC curve of the SL (line in red) compared to the sensitivity and specificity of the modified Rio score (points in blue) for a prognostic up to 2 years in the OFSEP validation sample.

```
library("RISCA")

.pred <- predict(sl2, newdata=dataVALID)

dataVALID$sl <- 1 - .pred$predictions$sl[,sum(.pred$times<2)]

roc.sl <- roc.time(times="time", failures="event", variable="sl",
                   confounders=~1, data=dataVALID, pro.time=2,
                   precision=seq(0.1, 0.9, by=0.01))
# Note: “confounders=~1” for usual time-dependent ROC curves,
# i.e., without considering confounder (doi:10.1177/0962280217702416)

roc.rio <- roc.time(times="time", failures="event", variable="rio",
                    confounders=~1, data=dataVALID, pro.time=2)

plot(roc.sl, col=2, type="l", xlab="1-Specificity", ylab="Sensitivity",
     cex.lab=0.8, cex.axis=0.8)

points(x=1-roc.rio$table$sp, y=roc.rio$table$se, col=4)

legend("bottomright", legend=
  paste("AUC =", round(roc.sl$auc, 3)), cex=0.8 )
```

References

O. Aalen. Nonparametric Inference for a Family of Counting Processes. *The Annals of Statistics*, 6(4): 701–726, July 1978. ISSN 0090-5364. doi: 10.1214/aos/1176344247. [p6]

P. K. Andersen, Ø. Borgan, N. L. Hjort, E. Arjas, J. Stene, and O. O. Aalen. Counting Process Models for Life History Data: A Review [with Discussion and Reply]. *Scandinavian Journal of Statistics*, 12(2): 97–158, 1985. [p4]

E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini. Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. *Statistics in Medicine*, 17(10):1169–1186, May 1998. ISSN 0277-6715. doi: 10.1002/(sici)1097-0258(19980530)17:10<1169::aid-sim796>3.0.co;2-d. [p6]

L. Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996. doi: 10.1007/BF00117832. [p4]

D. R. Cox. Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, Jan. 1972. ISSN 00359246. doi: 10.1111/j.2517-6161.1972.tb00899.x. [p4]

D. Faraggi and R. Simon. A neural network model for survival data. *Statistics in Medicine*, 14(1):73–82, Jan. 1995. ISSN 02776715, 10970258. doi: 10.1002/sim.4780140108. [p4]

J. J. Goeman. L_1 Penalized Estimation in the Cox Proportional Hazards Model. *Biometrical Journal*, pages NA–NA, Nov. 2009. ISSN 03233847, 15214036. doi: 10.1002/bimj.200900028. [p4]

M. K. Golmakani and E. C. Polley. Super Learner for Survival Data Prediction. *The International Journal of Biostatistics*, 16(2), Feb. 2020. ISSN 1557-4679. doi: 10.1515/ijb-2019-0065. [p4, 7]

H. Hung and C.-T. Chiang. Estimation methods for time-dependent AUC models with survival data. *Canadian Journal of Statistics*, 38(1):8–26, 2010. ISSN 1708-945X. doi: 10.1002/cjs.10046. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cjs.10046. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cjs.10046. [p6]

H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3), Sept. 2008. ISSN 1932-6157. doi: 10.1214/08-AOAS169. [p4, 6]

F. Le Borgne, A. Chatton, M. Léger, R. Lenain, and Y. Foucher. G-computation and machine learning for estimating the causal effects of binary exposure statuses on binary outcomes. *Scientific Reports*, 11(1):1435, Dec. 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-81110-0. [p5]

D. Y. Lin. On the Breslow estimator. *Lifetime Data Analysis*, 13(4):471–480, Dec. 2007. ISSN 1380-7870, 1572-9249. doi: 10.1007/s10985-007-9048-y. [p4, 6]

R. V. Phillips, M. J. Van Der Laan, H. Lee, and S. Gruber. Practical considerations for specifying a super learner. *International Journal of Epidemiology*, 52(4):1276–1285, Aug. 2023. ISSN 0300-5771, 1464-3685. doi: 10.1093/ije/dyad023. URL https://academic.oup.com/ije/article/52/4/1276/7076266. [p11, 12]

E. C. Polley and M. J. van der Laan. Super Learner In Prediction. In *U.C. Berkeley Division of Biostatistics Working Paper Series*, May 2010. [p5]

E. C. Polley and M. J. van der Laan. Chapter 16. Super Learning for Right-Censored Data. In M. J. van der Laan and S. Rose, editors, *Targeted Learning*, Springer Series in Statistics. Springer New York, New York, NY, 2011. ISBN 978-1-4419-9781-4 978-1-4419-9782-1. doi: 10.1007/978-1-4419-9782-1. [p4]

E. C. Polley, S. Rose, and M. van der Laan. Chapter 3. Super Learning. In M. J. van der Laan and S. Rose, editors, *Targeted Learning*, Springer Series in Statistics. Springer New York, New York, NY, 2011. ISBN 978-1-4419-9781-4 978-1-4419-9782-1. doi: 10.1007/978-1-4419-9782-1. [p4]

P. S. Rosenberg. Hazard Function Estimation Using B-Splines. *Biometrics*, 51(3):874, Sept. 1995. ISSN 0006341X. doi: 10.2307/2532989. [p4]

P. Royston and M. K. B. Parmar. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in Medicine*, 21(15):2175–2197, Aug. 2002. ISSN 0277-6715. doi: 10.1002/sim.1203. [p6]

P. K. Shivaswamy, W. Chu, and M. Jansche. A Support Vector Approach to Censored Targets. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 655–660, Omaha, NE, USA, Oct. 2007. IEEE. ISBN 978-0-7695-3018-5. doi: 10.1109/ICDM.2007.93. [p4]

N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. *Journal of Statistical Software*, 39(5), 2011. ISSN 1548-7660. doi: 10.18637/jss.v039.i05. [p6]

M. Sormani, J. Rio, M. Tintorè, A. Signori, D. Li, P. Cornelisse, B. Stubinski, M. Stromillo, X. Montalban, and N. De Stefano. Scoring treatment response in patients with relapsing multiple sclerosis. *Multiple Sclerosis Journal*, 19(5):605–612, Apr. 2013. ISSN 1352-4585, 1477-0970. doi: 10.1177/1352458512460605. [p11]

H. Uno, T. Cai, M. J. Pencina, R. B. D'Agostino, and L. J. Wei. On the C-statistics for Evaluating Overall Adequacy of Risk Prediction Procedures with Censored Survival Data. *Statistics in medicine*, 30(10): 1105–1117, May 2011. ISSN 0277-6715. doi: 10.1002/sim.4154. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3079915/. [p5]

M. J. van der Laan and S. Dudoit. Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples. Technical report, Division of Biostatistics, University of California, Bercley, 2003. [p4]

M. J. van der Laan, E. C. Polley, and A. E. Hubbard. Super Learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), Jan. 2007. ISSN 1544-6115, 2194-6302. doi: 10.2202/1544-6115.1309. [p4]

H. van Houwelingen, H. Putter, and J. C. van Houwelingen. *Dynamic Prediction in Clinical Survival Analysis*. Number 123 in Monographs on Statistics and Applied Probability. CRC Press, Taylor & Francis, Boca Raton, Fla., 2012. ISBN 978-1-4398-3543-2 978-1-4398-3533-3. [p5]

L. J. Wei. The accelerated failure time model: A useful alternative to the cox regression model in survival analysis. *Statistics in Medicine*, 11(14-15):1871–1879, 1992. ISSN 1097-0258. doi: 10.1002/sim.4780111409. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4780111409. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.4780111409. [p4]

T. Westling. \pkg{survSuperLearner}: Super Learning for Conditional Survival Functions with Right-Censored Data, 2021. [p4, 7]

Camille Sabathe
Nantes University, INSERM U1246 SPHERE, France
22, Bd Benoni Goullin, F-44200 Nantes
ORCiD: 0000-0001-9353-691X
camille.sabathe@univ-nantes.fr

Yohann Foucher
University and Hospital of Poitiers, CIC INSERM 1402
2, rue de la Miletrie, F-86000 Poitiers, France
ORCiD: 0000-0003-0330-7457
yohann.foucher@univ-poitiers.fr