Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

SuperSurv: A Unified Framework for Machine Learning Ensembles in Survival Analysis

Yue Lyu
UT MD Anderson Cancer Center

Xuelin Huang
UT MD Anderson Cancer Center

Steven Hsesheng Lin
UT MD Anderson Cancer Center

Ziyi Li
UT MD Anderson Cancer Center

Abstract

This paper introduces **SuperSurv**, a user-friendly R package for building, evaluating, and interpreting ensemble models for right-censored survival data. Although many survival modeling methods are available, existing tools are often model-specific and lack a unified platform for systematically integrating, comparing, and ensembling heterogeneous learners. **SuperSurv** addresses this gap by providing a unified interface for diverse survival learners, including models that return full survival curves as well as methods that produce only risk scores. All learner outputs are mapped to calibrated survival probability curves on a common evaluation time grid, enabling direct comparison and ensemble construction across heterogeneous model classes.

SuperSurv implements stacking of survival models using inverse-probability-of-censoring weighted (IPCW) Brier risk to estimate ensemble weights in the presence of right censoring. The framework integrates hyperparameter tuning, time-dependent benchmarking metrics, and visualization tools for survival model evaluation. In addition, the package provides post-hoc interpretability utilities based on SHAP values and supports covariate-adjusted restricted mean survival time (RMST) contrasts through g-computation.

By bridging the gap between theoretical rigor and clinical application, **SuperSurv** offers researchers a comprehensive ecosystem for modern survival analysis. The **SuperSurv** package is open-source and freely available at `https://github.com/yuelyu21/SuperSurv`. An empirical example using the METABRIC breast cancer dataset demonstrates a complete workflow from model training and benchmarking to explainability and clinically interpretable survival contrasts.

Keywords: survival analysis, ensemble learning, Super Learner, explainable AI, R.

1. Introduction

Supervised learning for time-to-event outcomes is a cornerstone of precision medicine and public health, where accurate predictions drive individualized risk assessment. Traditionally, the Cox proportional hazards (PH) model (Cox 1972) has been the dominant analytical tool. However, its reliance on strict log-linear and proportional hazards assumptions often renders it inadequate for capturing the complex data structures present in modern, high-dimensional clinical datasets (Therneau and Grambsch 2000).

To address these limitations, the statistical learning community has developed a vast array of survival-adapted algorithms. These span from non-parametric tree-based methods—such as Random Survival Forests (Ishwaran, Kogalur, Blackstone, and Lauer 2008; Wright and Ziegler 2017) and Accelerated Oblique Random Forests (Jaeger, Welden, Lenoir, Speiser, Segar, Pandey, and Pajewski 2024)—to powerful gradient and component-wise boosting machines (Chen and Guestrin 2016; Binder and Schumacher 2008). Furthermore, the landscape has expanded to include regularized regressions (Simon, Friedman, Hastie, and Tibshirani 2011), flexible generalized additive models (Wood 2017), survival support vector machines (Fouodo, König, Weihs, Ziegler, and Wright 2018), and Bayesian additive regression trees (Sparapani, Logan, McCulloch, and Laud 2016).

Despite this proliferation of methods, it is rarely known *a priori* which specific algorithm or hyperparameter configuration will perform best for a given dataset. Rather than committing to a single parametric or algorithmic specification, predictive accuracy can be systematically optimized by combining candidate models using an ensembling framework.

1.1. The Super Learner and the Survival Challenge

The Super Learner (SL) provides an asymptotically optimal stacking framework for selecting a best-performing learner or an optimally weighted ensemble from a prespecified library by minimizing an out-of-sample loss function via V-fold cross-validation (Van der Laan, Polley, and Hubbard 2007; Polley, Rose, and van der Laan 2011). Originally developed for fully observed continuous or binary outcomes (e.g., via the **SuperLearner** package), extending SL to right-censored time-to-event data presents two major methodological and computational challenges.

First, because outcomes are right-censored, traditional loss functions (such as standard mean squared error) cannot be directly applied. Performance objectives and cross-validation procedures must incorporate Inverse Probability of Censoring Weighting (IPCW) (Graf, Schmoor, Sauerbrei, and Schumacher 1999; Gerds and Schumacher 2006) to remain consistent. Over the past decade, the statistical literature has steadily evolved to address this challenge:

- **Discrete-Time Stacking:** Early theoretical work by Polley and van der Laan (2011) extended SL by reframing continuous survival time as a sequence of discrete-time hazard classification tasks. While innovative, this approach requires artificially expanding the dataset across time intervals, which becomes computationally prohibitive for high-dimensional clinical data.

- **Restricted Continuous-Time Stacking:** Golmakani and Polley (2020) subsequently proposed a continuous-time Super Learner; however, they restricted the candidate library strictly to proportional hazards-based algorithms, such as basic Cox regression,

Elastic Net, and Lasso-Cox. This formulation structurally excluded modern, non-parametric machine learning engines.

- **Generalized Continuous-Time Stacking:** More recent breakthroughs generalized the continuous-time framework to accommodate arbitrary machine learning algorithms. Westling, Luedtke, Gilbert, and Carone (2024) established a robust cross-fitted IPCW objective for drawing inference from treatment-specific survival curves using diverse learners.

- **Joint Ensembling:** Building on the need for robust censoring weights, Munch and Gerds (2024) introduced the "joint survival super learner," demonstrating the theoretical necessity of simultaneously optimizing both the event-time algorithms and the censoring-distribution algorithms.

Second, even with robust continuous-time IPCW loss functions in place, survival algorithms fundamentally differ in their *prediction targets*. Some methods—such as classical parametric models, Kaplan-Meier estimators, and Random Survival Forests (Ishwaran *et al.* 2008)—natively output an estimated survival curve $\widehat{S}(t \mid \boldsymbol{X})$ evaluated across a predefined time grid. Conversely, many of the most highly predictive modern machine learning engines—such as Gradient Boosting Machines (e.g., **xgboost**), Support Vector Machines (e.g., **survivalsvm**), and penalized regressions (e.g., **glmnet**)—are optimized via partial likelihood and output only a relative risk score or linear predictor $\eta(\boldsymbol{X})$ without a fully specified baseline hazard. This architectural mismatch creates a critical implementation gap. Without an additional, model-agnostic mathematical calibration step, risk-score learners cannot be directly stacked alongside curve-based learners. Furthermore, naïve combinations of predictions originating from such disparate algorithmic sources are not guaranteed to be properly scaled as probabilities, directly comparable, or even monotonically decreasing over time t.

Third, even if a diverse survival ensemble is successfully constructed, its clinical application is often hindered by a lack of interpretability and difficulties in evaluating treatment efficacy. High-dimensional ML ensembles are inherently "black boxes," making it difficult for clinicians to interpret individual patient risk factors. Furthermore, extracting a population-level treatment effect from a highly non-linear ensemble is challenging. While classical analysis relies on the Hazard Ratio (HR), the HR is non-collapsible and often misleading when the true underlying data exhibits the crossing hazards naturally captured by ML algorithms (Martinussen and Vansteelandt 2013). Therefore, researchers must shift to absolute, collapsible measures, such as the covariate-adjusted Restricted Mean Survival Time (RMST). By utilizing G-computation (standardization) over the ensemble's predictions (Uno, Claggett, Tian, Inoue, Gallo, Miyata, Schrag, Takeuchi, Uyama, Zhao *et al.* 2014), researchers can obtain a robust marginal treatment effect, which carries a formal causal interpretation only under standard identifiability assumptions.

Despite rapid development of machine learning methods for survival analysis, the practical software ecosystem remains fragmented. Individual packages typically implement a single model class or a narrow family of algorithms, making it difficult to combine heterogeneous learners within a unified ensemble framework. In practice, analysts must manually reconcile incompatible prediction formats, time grids, and risk-score outputs before ensemble learning can be applied.

1.2. The Fragmented Software Landscape

While the theoretical foundations of both discrete-time and continuous-time survival Super Learning have been established, the current R (R Core Team 2024) software ecosystem for survival ensemble learning remains fragmented. Translating these methodological advances into accessible, unified software has proven challenging, leaving applied researchers with several distinct but incomplete paradigms:

1. **Object-Oriented ML Frameworks:** Packages like **mlr3proba** (Sonabend, Király, Bender, Bischl, and Lang 2021) provide an extensive machine learning ecosystem for survival analysis. However, it relies heavily on **R6** object-oriented syntax, presenting a steep learning curve. Furthermore, to ensemble modern risk-score-based learners (like **xgboost**) alongside curve-based learners, users must manually construct complex reduction pipelines (e.g., `distrcompositor`) to estimate the baseline hazard; it does not happen automatically.

2. **Classification-Based Stacking:** Recent frameworks like **survML** (Wolock, Gilbert, Simon, and Carone 2024) avoid IPCW entirely by recasting survival estimation as a pooled binary classification problem ("global survival stacking") evaluated over a discrete time grid. While this allows the use of standard binary classifiers, it artificially expands the dataset size across the time grid (incurring heavy computational costs) and precludes the direct ensembling of native, specialized survival algorithms (e.g., Random Survival Forests).

3. **Parametric-Focused Ensembles:** The **survivalSL** package (Sabathe and Foucher 2024) recently introduced a practical survival Super Learner. However, its utility is constrained by a library focused predominantly on classical parametric models (e.g., AFT models with Weibull or Gamma distributions) and standard Cox penalizations. Because it lacks an automated calibration engine, it cannot seamlessly integrate the raw risk scores of modern "black-box" ML engines.

4. **Methodological Codebases:** The foundational continuous-time SL papers are accompanied by research codebases, such as the **survSuperLearner** GitHub package (Westling *et al.* 2024) and the **jossl** scripts (Munch and Gerds 2024). While mathematically rigorous, they function primarily as proof-of-concept repositories rather than production-ready software. Both are fundamentally constrained by extremely limited built-in algorithm libraries. For instance, **survSuperLearner** supports a library of models that natively output survival curves (e.g., Cox, Weibull, Random Survival Forests) but structurally excludes modern risk-score-based algorithms (e.g., gradient boosting, SVM) because it lacks an automated baseline hazard calibration blueprint. Similarly, **jossl** provides native support for only three algorithms. Furthermore, neither ecosystem provides native tools for Explainable AI (XAI) or the estimation of covariate-adjusted marginal treatment effects.

These limitations motivate the development of **SuperSurv**, an R package designed to unify survival ensemble learning, model calibration, and interpretation within a single extensible framework.

1.3. The SuperSurv Package: Design and Contributions

The **SuperSurv** package is designed to eliminate this software fragmentation. Building conceptually on the continuous-time framework of Westling *et al.* (2024), **SuperSurv** provides the robust software infrastructure necessary to transform methodological theory into a comprehensive, user-facing predictive ecosystem.

Crucially, **SuperSurv** extends beyond pure prediction to directly address the interpretability and treatment evaluation gaps identified above. By integrating Explainable AI (XAI) directly into its architecture, the package bridges complex ensembles to the **fastshap** (Štrumbelj and Kononenko 2014; Greenwell 2024) and **survex** (Spytek, Krzyziński, Langbein, Baniecki, Wright, and Biecek 2023) ecosystems. Additionally, the package provides native G-computation routines to estimate covariate-adjusted Average Treatment Effects (ATE) via the Restricted Mean Survival Time (RMST) (Uno *et al.* 2014), entirely bypassing the non-collapsible Hazard Ratio.

By bridging the gap between theoretical rigor and precision medicine application, **SuperSurv** offers the following core components:

- **Model-Agnostic Calibration:** a plug-in procedure that maps arbitrary risk scores $\eta(\boldsymbol{X})$ to calibrated survival curves via Breslow-type baseline hazard recovery, enabling machine learning models that produce only risk scores to be integrated seamlessly with models that estimate full survival functions. This allows modern ML (e.g., **xgboost**, **survivalsvm**) to be ensembled seamlessly alongside classical models.
- **Dual-Objective Stacking:** Convex weight estimation by minimizing user-specified IPCW Brier risk or IPCW Negative Log-Likelihood (Log Loss), a feature absent in foundational survival SL packages.
- **Unified API and Expanded Library:** A standardized API representing predictions as absolute survival curves on a user-defined time grid, harmonizing 19 distinct base algorithms and 6 automated high-dimensional screening algorithms.
- **Treatment Evaluation and XAI Pipelines:** Native G-computation estimators for the covariate-adjusted ΔRMST, alongside seamless bridging for global and time-dependent interpretability.
- **High-dimensional screening:** Automated variable screening algorithms (e.g., Elastic Net, Marginal Cox, Variance) to handle genomic-scale datasets.
- **Evaluation Utilities:** A built-in visualization and benchmarking suite for time-dependent Brier scores, Uno's C-index, and calibration trajectories.

1.4. Organization of the paper

The remainder of this paper is organized as follows. Section 2.1 introduces the statistical notation for right-censored survival data and inverse probability of censoring weighting (IPCW). Section 2.2 describes the harmonization blueprint used to convert arbitrary risk scores into calibrated survival probabilities. Sections 2.3 and 2.4 detail the theoretical and computational representations of the dual-objective IPCW loss functions used to optimize the ensemble

weights. Section 2.5 highlights how this architecture provides dynamic robustness to proportional hazards misspecification, followed by an outline of the iterative survival–censoring optimization algorithm in Section 2.6.

Shifting from methodology to the software ecosystem, Section 3 outlines the **SuperSurv** R package design, demonstrating its unified API and Explainable AI (XAI) integration. Finally, Section 4 illustrates an end-to-end empirical application of the package using the high-dimensional METABRIC breast cancer dataset (Curtis, Shah, Chin, Turashvili, Rueda, Dunning, Speed, Lynch, Samarajiwa, Yuan *et al.* 2012; Kvamme, Borgan, and Scheel 2019).

2. Statistical Framework and Ensemble Learning Architecture

This section introduces the statistical framework underlying the SuperSurv package, including the representation of right-censored survival data, transformation of heterogeneous model outputs to survival probabilities, IPCW loss functions for ensemble learning, and the iterative procedure used to jointly estimate survival and censoring models.

2.1. Right-Censored Survival Data and IPCW Notation

Let T denote the true event time, C the right-censoring time, and $\Delta = \mathbb{I}(T \leq C)$ the event indicator. Let $\boldsymbol{X} \in \mathbb{R}^p$ represent a p-dimensional vector of baseline covariates. We observe n independent and identically distributed realizations of the right-censored data structure

$$O = (\widetilde{T}, \Delta, \boldsymbol{X}),$$

where $\widetilde{T} = \min(T, C)$ is the observed follow-up time.

We assume conditionally independent censoring,

$$T \perp C \mid \boldsymbol{X},$$

which ensures identifiability of the conditional survival distribution. Our primary predictive target is the conditional survival function

$$S(t \mid \boldsymbol{X}) = \Pr(T > t \mid \boldsymbol{X}), \tag{1}$$

which is related to the cumulative hazard function

$$\Lambda(t \mid \boldsymbol{X}) = \int_0^t \lambda(u \mid \boldsymbol{X})\, du, \qquad S(t \mid \boldsymbol{X}) = \exp\{-\Lambda(t \mid \boldsymbol{X})\}. \tag{2}$$

Because the event time may be censored, empirical risk functions based on fully observed outcomes cannot be applied directly. To address this issue, we employ inverse probability of censoring weighting (IPCW). Let

$$G(t \mid \boldsymbol{X}) = \Pr(C > t \mid \boldsymbol{X})$$

denote the conditional survival function of the censoring distribution. For a prediction evaluated at time t, the IPCW weight is defined as

$$W^\star(t; \widetilde{T}, \Delta, \boldsymbol{X}) = \frac{\Delta\, \mathbb{I}(\widetilde{T} \leq t)}{\widehat{G}(\widetilde{T} \mid \boldsymbol{X})} + \frac{\mathbb{I}(\widetilde{T} > t)}{\widehat{G}(t \mid \boldsymbol{X})}. \tag{3}$$

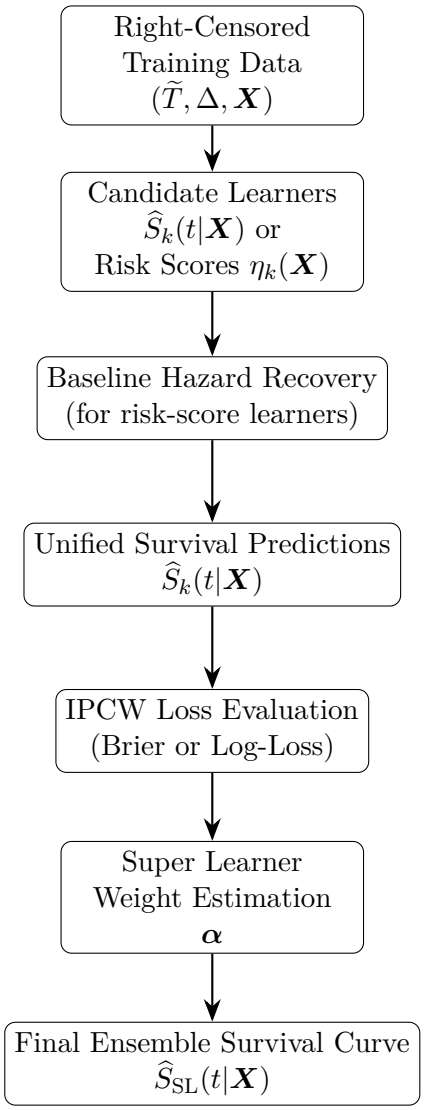

Figure 1: Overview of the **SuperSurv** framework. Heterogeneous survival learners are first converted to survival probability curves (if necessary), then combined using IPCW-based meta-learning to produce the final ensemble survival estimator.

The first term corresponds to individuals who experience the event prior to time t, while the second term accounts for individuals who remain at risk beyond time t. These weights correct for informative loss of follow-up and allow censored observations to contribute appropriately to risk estimation. These weights enable unbiased estimation of prediction risk under right-censoring and form the basis of the ensemble loss functions used in Section 2.3.

In the **SuperSurv** package, this mathematical formulation maps directly to the core function interface for fitting the ensemble model.

2.2. Model Output Harmonization

To construct a valid Super Learner ensemble, all candidate algorithms must estimate the

same target parameter,

$$S(t \mid \boldsymbol{X}) = \Pr(T > t \mid \boldsymbol{X}),$$

on a common evaluation time grid

$$\mathcal{T} = \{t_1, t_2, \ldots, t_m\}.$$

However, modern survival learning algorithms produce heterogeneous outputs. Some algorithms directly estimate the survival function, while others optimize likelihoods that yield only relative risk scores or linear predictors. Consequently, the predicted quantities must be transformed into a common representation before stacking can occur.

Let $\widehat{S}_k(t \mid \boldsymbol{X})$ denote the survival estimate produced by learner k. In **SuperSurv**, every learner is wrapped into a standardized interface that returns a matrix

$$\widehat{\mathbf{S}}_k = \left[\widehat{S}_k(t_j \mid \boldsymbol{X}_i)\right]_{i=1,\ldots,n;\ j=1,\ldots,m},$$

where rows correspond to individuals and columns correspond to time points in $\mathcal{T}$.

To achieve this harmonization, **SuperSurv** distinguishes between three classes of learners:

1. **Direct survival learners** that natively output survival curves.
2. **Hazard-based learners** that produce linear predictors or log-risk scores.
3. **Utility-score learners** that produce ranking scores not directly linked to the hazard scale.

Table 1 summarizes the output format of each learner currently implemented in **SuperSurv**.

Table 1: Output format of base learners implemented in **SuperSurv**.

Learner	Output type	Conversion to survival curve
Kaplan–Meier	survival probability	direct
Parametric AFT (Weibull, Lognormal, etc.)	survival probability	direct
Random Survival Forest (**ranger**, **randomForestSRC**)	survival probability	direct
Oblique Random Survival Forest (**aorsf**)	survival probability	direct
Bayesian Additive Regression Trees (**BART**)	survival probability	direct
Generalized Additive Model (**mgcv**)	survival probability	direct
Cox PH (**survival**)	log-risk	Breslow hazard recovery
Penalized Cox (**glmnet**, ridge)	linear predictor	Breslow hazard recovery
Gradient Boosting (**gbm**)	log-risk	Breslow hazard recovery
XGBoost Cox	log-risk	Breslow hazard recovery
CoxBoost	log-risk	Breslow hazard recovery
Survival Tree (**rpart**)	event-rate proxy	Breslow hazard recovery
Survival SVM	utility score	Cox calibration

Baseline hazard recovery

Many machine learning survival algorithms optimize the Cox partial likelihood and therefore output only a linear predictor

$$\eta(\boldsymbol{X}) = \beta^\top \boldsymbol{X}$$

or an equivalent log-risk score. Because the partial likelihood leaves the baseline hazard $H_0(t)$ unspecified, such models do not directly provide absolute survival probabilities.

To recover survival probabilities, **SuperSurv** estimates the baseline cumulative hazard using a Breslow-type estimator. Let (t_i, Δ_i) denote the observed follow-up time and event indicator for subject i, and let

$$R(u) = \{i : t_i \geq u\}$$

denote the risk set at time u.

For a learner producing linear predictors η_i, the cumulative baseline hazard is estimated as

$$\widehat{H}_0(t) = \sum_{u \leq t} \frac{d(u)}{\sum_{j \in R(u)} \exp(\eta_j)},$$

where $d(u)$ denotes the number of events observed at time u.

Algorithm 1 describes the computational procedure used within **SuperSurv** to recover the baseline hazard from predicted risk scores.

Algorithm 1 Baseline hazard recovery (Breslow-type updates)

Require: Training data (t_i, Δ_i, η_i) for $i = 1, \ldots, n$, where $\eta_i = \eta(\mathbf{X}_i)$ denotes the predicted risk score.

1: Identify ordered unique event times $\{u_1, \ldots, u_m\}$ such that $\Delta_i = 1$.

2: **for** each event time u **do**

3: Compute the number of events

$$d(u) = \sum_{i=1}^{n} I(t_i = u)\Delta_i$$

4: Define the risk set

$$R(u) = \{i : t_i \geq u\}$$

5: Compute the hazard increment

$$d\widehat{H}_0(u) = \frac{d(u)}{\sum_{j \in R(u)} \exp(\eta_j)}$$

6: **end for**

7: Form the cumulative baseline hazard

$$\widehat{H}_0(t) = \sum_{u \leq t} d\widehat{H}_0(u)$$

8: Return the survival estimate

$$\widehat{S}(t \mid \mathbf{X}) = \exp\{-\widehat{H}_0(t) \exp(\eta(\mathbf{X}))\}$$

Cox calibration for utility-score learners

Certain algorithms, such as Survival Support Vector Machines, produce utility scores that

are not guaranteed to lie on the proportional hazards scale. For these learners, **SuperSurv** fits a univariate Cox proportional hazards model

$$\lambda(t \mid s) = \lambda_0(t) \exp(\beta s)$$

where s denotes the raw model score. This step calibrates the score onto the hazard scale before applying the Breslow estimator.

Interpolation onto a common time grid

Different learners evaluate survival curves at different time points. To ensure compatibility across models, predicted survival curves are interpolated onto the shared grid $\mathcal{T}$ using a step-function interpolation

$$\widehat{S}(t_j \mid \boldsymbol{X}) = \widehat{S}(u_k \mid \boldsymbol{X}) \quad \text{for } u_k \leq t_j < u_{k+1}.$$

This constant interpolation preserves the monotonicity of survival curves while ensuring all learners produce predictions on the same grid.

Numerical stability safeguards

Several engineering safeguards are implemented to ensure numerical stability during hazard recovery:

- Linear predictors are truncated to avoid numerical overflow when computing $\exp(\eta)$.
- Survival curves are constrained to remain monotone decreasing via cumulative minima operations.
- Design matrices for training and prediction are aligned by padding missing columns to handle differing factor levels across cross-validation folds.
- Extremely small time values are truncated to prevent undefined logarithms in parametric models.

These safeguards ensure robust prediction across heterogeneous machine learning libraries.

Once predictions from heterogeneous learners are expressed as survival probability curves $S(t \mid \boldsymbol{X})$, they can be compared and combined using a common prediction loss. The SuperSurv framework estimates ensemble weights by minimizing an inverse-probability-of-censoring weighted loss defined over these survival predictions.

2.3. Dual-Objective IPCW Loss Functions

The goal of the meta-learning stage is to estimate the optimal convex combination of K candidate learners. Let the stacked predictor be defined as

$$\widehat{S}_{\mathrm{SL}}(t \mid \boldsymbol{X}) = \sum_{k=1}^{K} \alpha_k \widehat{S}_k(t \mid \boldsymbol{X}), \tag{4}$$

where the weight vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$ lies on the simplex

$$\alpha_k \geq 0, \quad \sum_{k=1}^{K} \alpha_k = 1.$$

Weights are estimated via V-fold cross-validation by minimizing an inverse-probability-of-censoring weighted (IPCW) prediction loss.

IPCW Brier-Loss The default objective minimizes the IPCW brier-loss, which forms the basis of the Super Learner for conditional survival distributions developed by Westling *et al.* (2024). For a prediction $S(t \mid \boldsymbol{X})$ at time t, the loss is defined as

$$\mathcal{L}_{BS}(S, t, G) = \frac{\Delta \mathbf{1}(\widetilde{T} \leq t)}{\widehat{G}(\widetilde{T} \mid \boldsymbol{X})}(0 - S(t \mid \boldsymbol{X}))^2 + \frac{\mathbf{1}(\widetilde{T} > t)}{\widehat{G}(t \mid \boldsymbol{X})}(1 - S(t \mid \boldsymbol{X}))^2.$$

This formulation corresponds to an inverse-probability-of-censoring weighted squared error between the predicted survival probability and the true survival status at time t.

While the integrated Brier score provides the theoretical foundation for survival distribution stacking (Westling *et al.* 2024), modern machine learning systems often rely on log-loss objectives due to their strong probabilistic interpretation and sensitivity to miscalibrated predictions. In the binary outcome setting, log-loss is equivalent to the cross-entropy loss widely used in machine learning. To accommodate this perspective, **SuperSurv** additionally implements an IPCW log-loss objective for estimating ensemble weights.

Compared with squared-error loss, log-loss penalizes overconfident but incorrect survival predictions more strongly, which may improve ensemble calibration in practice.

IPCW Log-Loss In addition to the integrated Brier score objective, **SuperSurv** provides an alternative meta-learning criterion based on IPCW log-loss. This loss emphasizes probabilistic accuracy by penalizing confident but incorrect predictions more strongly, making it particularly suitable when well-calibrated survival probabilities are required.

The IPCW log-loss is defined as

$$\mathcal{L}_{NLL}(S, t, G) = -\left[\frac{\Delta \mathbf{1}(\widetilde{T} \leq t)}{\widehat{G}(\widetilde{T} \mid \boldsymbol{X})} \log(1 - S(t \mid \boldsymbol{X})) + \frac{\mathbf{1}(\widetilde{T} > t)}{\widehat{G}(t \mid \boldsymbol{X})} \log(S(t \mid \boldsymbol{X}))\right].$$

The ensemble weights are obtained by minimizing the empirical IPCW log-loss

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha} \in \Delta^K} \sum_{i=1}^{n} \sum_{t \in \mathcal{T}} \mathcal{L}_{NLL}(\widehat{S}_{\boldsymbol{\alpha}}(t \mid \boldsymbol{X}_i), t, \widehat{G}),$$

where Δ^K denotes the simplex of nonnegative weights that sum to one.

To our knowledge, this represents one of the first practical implementations of an IPCW cross-entropy objective for survival distribution stacking within a Super Learner framework.

While the loss functions above are defined for a fixed time point, the SuperSurv framework evaluates prediction error across an entire time horizon. In practice, the continuous-time loss

is approximated using a discrete evaluation grid, which leads to the computational representation described below.

2.4. Computational Representation of the Loss Function

Following Westling *et al.* (2024), the Super Learner for survival outcomes is defined through the minimization of an expected loss for the conditional survival distribution. For the weighted integrated Brier score, the population loss is defined as

$$L_{S,G} = \int_0^\tau \left(S(t \mid \boldsymbol{X}) - f_G(t)\right)^2 dt, \tag{5}$$

where

$$f_G(t) = 1 - \frac{\Delta\, \mathbb{I}(\widetilde{T} \leq t)}{G(\widetilde{T} \mid \boldsymbol{X})}. \tag{6}$$

Here $\widetilde{T} = \min(T, C)$ denotes the observed follow-up time, $\Delta = \mathbb{I}(T \leq C)$ is the event indicator, and $G(t \mid \boldsymbol{X})$ denotes the conditional censoring survival function. The function $f_G(t)$ serves as an inverse-probability-of-censoring weighted pseudo-outcome that corrects for right censoring.

In practice, the integral over time cannot be evaluated analytically and must be approximated numerically. **SuperSurv** therefore evaluates predictions on a predefined time grid

$$\mathcal{T} = \{t_1, t_2, \ldots, t_m\},$$

and approximates the continuous-time loss using a discrete summation over this grid.

Let $\widehat{S}_j(t \mid \boldsymbol{X}_i)$ denote the predicted survival probability for observation i at time t from candidate learner j. The empirical loss used for model comparison is then computed as

$$\widehat{L}_j = \frac{1}{n|\mathcal{T}|} \sum_{i=1}^{n} \sum_{t \in \mathcal{T}} \left(\widehat{S}_j(t \mid \boldsymbol{X}_i) - f_G^{(i)}(t)\right)^2, \tag{7}$$

where $f_G^{(i)}(t)$ denotes the IPCW pseudo-outcome for observation i evaluated at time t. This formulation corresponds directly to the matrix-based implementation used in the package, where survival predictions are represented as an $n \times |\mathcal{T}|$ matrix and the loss is evaluated element-wise across observations and time points.

The asymptotic properties of the Super Learner estimator under this loss function, including oracle inequalities and consistency results, are established in Westling *et al.* (2024). The implementation in **SuperSurv** follows this theoretical framework while providing a computationally efficient discrete-time approximation suitable for large-scale machine learning ensembles.

2.5. Robustness to Proportional Hazards Misspecification

A key advantage of the Super Learner framework is its ability to combine learners that rely on different structural assumptions about the hazard function.

Many classical survival models, including Cox proportional hazards models and their penalized extensions, assume that the hazard function factorizes as

$$\lambda(t \mid \boldsymbol{X}) = \lambda_0(t) \exp(\beta^\top \boldsymbol{X}),$$

which implies proportional hazards across covariate strata. While this assumption simplifies estimation and interpretation, it can be violated in complex biomedical datasets where treatment effects or covariate influences vary over time.

The **SuperSurv** library intentionally includes both proportional hazards models (e.g., **coxph**, **glmnet**, **xgboost** Cox objectives) and flexible machine learning estimators that do not rely on proportional hazards assumptions (e.g., Random Survival Forests, Bayesian Additive Regression Trees, and survival trees).

Because ensemble weights are estimated by minimizing cross-validated prediction risk, the Super Learner automatically adapts to the underlying hazard structure. When the proportional hazards assumption holds, Cox-type learners typically achieve low prediction error and receive higher ensemble weights. Conversely, when proportional hazards are violated, flexible nonparametric learners tend to achieve better predictive performance, causing the ensemble weights assigned to Cox-type models to shrink toward zero.

This adaptive weighting mechanism provides robustness to proportional hazards misspecification while retaining the efficiency and interpretability of classical models when their assumptions are approximately satisfied.

2.6. Iterative Survival–Censoring Ensemble Optimization

The IPCW loss functions introduced in Section 2.3 depend on accurate estimation of the censoring survival function $G(t \mid \boldsymbol{X})$. Rather than relying on a single parametric censoring model, **SuperSurv** optionally estimates the censoring distribution using a second Super Learner ensemble. This follows the joint survival–censoring stacking framework proposed by Westling *et al.* (2024), in which the survival distribution $S(t \mid \boldsymbol{X})$ and the censoring distribution $G(t \mid \boldsymbol{X})$ are estimated simultaneously through an iterative procedure.

The key idea is that estimation of $S(t \mid \boldsymbol{X})$ depends on the censoring model through IPCW weights, while estimation of $G(t \mid \boldsymbol{X})$ can similarly be expressed using pseudo-outcomes constructed from the current estimate of the survival distribution. Consequently, the two ensembles are updated alternately until convergence.

Specifically, let $\mathcal{K}_S$ and $\mathcal{K}_G$ denote candidate learner libraries for the survival and censoring distributions, respectively. Cross-validated predictions from these learners are first obtained on a common evaluation time grid $\mathcal{T}$. The algorithm then iteratively updates the ensemble weights for $S(t \mid \boldsymbol{X})$ and $G(t \mid \boldsymbol{X})$ using pseudo-outcomes derived from the current estimates of the opposite distribution.

The ensemble weights are estimated by minimizing the cross-validated empirical loss subject to simplex constraints (nonnegative weights that sum to one). For the IPCW Brier score objective, this optimization reduces to a weighted least-squares problem, and the weights are estimated using nonnegative least squares (NNLS). Specifically, the cross-validated predictions are regressed on the IPCW pseudo-outcome under nonnegativity constraints, and the resulting coefficients are normalized to sum to one so that the final weight vector lies on the simplex $\Delta^K = \{\boldsymbol{\alpha} : \alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1\}$.

For the IPCW negative log-likelihood objective, the loss function is no longer quadratic, and the ensemble weights are obtained by minimizing the cross-validated cross-entropy loss using

constrained numerical optimization over the simplex.

Algorithm 2 Iterative survival–censoring stacking

Require: Candidate learner libraries $\mathcal{K}_S$ for $S(t \mid \boldsymbol{X})$ and $\mathcal{K}_G$ for $G(t \mid \boldsymbol{X})$.

1: Fit all candidate learners for $S(t \mid \boldsymbol{X})$ and $G(t \mid \boldsymbol{X})$ using V-fold cross-validation to obtain cross-validated predictions $\widehat{S}_k^{(-v)}(t \mid \boldsymbol{X}_i)$ and $\widehat{G}_m^{(-v)}(t \mid \boldsymbol{X}_i)$.

2: Fit an initial censoring model $\widehat{G}^{(0)}(t \mid \boldsymbol{X})$ on the full data.

3: **repeat**

4: Compute pseudo-outcomes for survival stacking

$$f_G(i,t) = 1 - \frac{\Delta_i \mathbf{1}(\widetilde{T}_i \leq t)}{\widehat{G}(\widetilde{T}_i \mid \boldsymbol{X}_i)}.$$

5: Estimate survival ensemble weights $\boldsymbol{\alpha}_S$ by minimizing the IPCW loss using $\widehat{S}_k^{(-v)}(t \mid \boldsymbol{X}_i)$.

6: Update survival ensemble

$$\widehat{S}_{\text{ens}}(t \mid \boldsymbol{X}) = \sum_k \alpha_{S,k} \widehat{S}_k(t \mid \boldsymbol{X}).$$

7: Compute pseudo-outcomes for censoring stacking

$$f_S(i,t) = 1 - \frac{(1-\Delta_i)\mathbf{1}(\widetilde{T}_i < t)}{\widehat{S}_{\text{ens}}(\widetilde{T}_i \mid \boldsymbol{X}_i)}.$$

8: Estimate censoring ensemble weights $\boldsymbol{\alpha}_G$ using predictions $\widehat{G}_m^{(-v)}(t \mid \boldsymbol{X}_i)$.

9: Update censoring ensemble

$$\widehat{G}_{\text{ens}}(t \mid \boldsymbol{X}) = \sum_m \alpha_{G,m} \widehat{G}_m(t \mid \boldsymbol{X}).$$

10: **until** maximum change in predictions $< \epsilon$

Following Westling *et al.* (2024), the candidate learners themselves are fitted only once during the initial cross-validation stage. Subsequent iterations involve only the estimation of ensemble weights, making the procedure computationally efficient.

In addition to the standard ensemble estimator, **SuperSurv** also provides an optional model-selection strategy that returns the single best performing learner. Specifically, users may specify `selection = "best"`, which selects the candidate learner with the smallest cross-validated risk and assigns it weight 1, with all other weights set to zero. This corresponds to a hard model selection procedure, whereas the default `selection = "ensemble"` option estimates a convex combination of learners that typically achieves lower prediction error by averaging across models.

3. Software Design and Usage

The SuperSurv package provides a unified interface for survival ensemble learning built on

the methodology described in Section 2. In addition to implementing the core Super Learner framework, the package includes utilities for model tuning, variable screening, performance evaluation, visualization, and model interpretation.

This section describes the overall package architecture and highlights several tools designed to facilitate practical survival modeling workflows.

3.1. Package Architecture

A major practical challenge in survival machine learning is the lack of a unified software interface. Popular algorithms such as Cox regression, random survival forests, gradient boosting, and penalized regression are implemented in different R packages, each requiring distinct data formats and producing predictions with incompatible output structures.

SuperSurv addresses this challenge by introducing a modular wrapper architecture that standardizes how survival learners are trained and how predictions are returned. Each base learner is implemented as a wrapper function with the prefix `surv.*`. These wrappers internally call external packages (e.g., **randomForestSRC**, **xgboost**, **glmnet**), but expose a unified interface to the **SuperSurv** framework.

The wrapper system ensures that all models return survival probability predictions on a common time grid. Regardless of the underlying algorithm, prediction outputs are converted into aligned matrices with dimension $N \times |\mathcal{T}|$, where N denotes the number of observations and $\mathcal{T}$ denotes the evaluation time grid introduced in Section 2.3. This harmonized structure allows heterogeneous learners to be combined within the Super Learner ensemble.

Available learners and screening algorithms can be inspected directly within the package:

```
R> # Install from GitHub
R> devtools::install_github("yuelyu21/SuperSurv")
R> library("SuperSurv")
R> library("survival")
R> list_wrappers()

--- Prediction Models (surv.*) ---
 [1] "surv.aorsf"       "surv.bart"        "surv.coxph"       "surv.glmnet"
 [5] "surv.parametric"  "surv.rfsrc"       "surv.weibull"     "surv.xgboost" ...

--- Screening Algorithms (screen.*) ---
[1] "screen.all"        "screen.elasticnet" "screen.marg"       "screen.var"
```

Prediction models are implemented as `surv.*` wrappers, while feature screening procedures follow the `screen.*` convention. This modular design allows new algorithms to be integrated into the framework by simply implementing a compatible wrapper that returns predictions in the required format.

Each wrapper follows a standardized interface that allows heterogeneous survival learners to be integrated seamlessly into the Super Learner framework. The design of this wrapper interface is described in Section 3.2.

3.2. Model Libraries and Wrapper Interface

All base learners in **SuperSurv** are implemented using a standardized wrapper interface. Each learner is defined as a function with the prefix `surv.*`. These functions serve as adapters that translate the input data into the format required by the underlying modeling package and convert model predictions into a standardized representation.

Specifically, each wrapper returns two components:

- `fit`: the fitted model object produced by the underlying algorithm.
- `pred`: a prediction matrix of dimension $N \times |\mathcal{T}|$, where N is the number of observations and $\mathcal{T}$ is the evaluation time grid.

This contract ensures that predictions from heterogeneous algorithms can be directly stacked within the Super Learner meta-learning stage described in Section 2.3.

Prediction methods follow the standard R S3 convention using `predict.surv.*` methods, which accept a `newdata` argument and return survival probability predictions on the common time grid.

```
R> # Fit a single wrapped base learner
R> fit0 <- surv.rfsrc(time  = train$time,
+                     event = train$event,
+                     X     = train[, covariate_cols],
+                     newdata   = test[, covariate_cols],
+                     new.times = t_grid)

R> # Wrapper returns a standardized prediction matrix on the common grid
R> dim(fit0$pred)

[1] n_test  length(t_grid)
```

3.3. Hyperparameter Tuning and Grid Search

Many survival learning algorithms require tuning of hyperparameters that strongly influence predictive performance. To facilitate systematic hyperparameter exploration, **SuperSurv** provides a dynamic grid-generation utility that automatically constructs multiple wrapper functions corresponding to different parameter combinations.

The function `create_grid()` generates new learner wrappers by expanding a user-specified hyperparameter grid and programmatically creating wrapper functions with fixed parameter values. Each generated learner is assigned a descriptive name encoding its hyperparameter configuration.

```
R> rf_tuning_params <- list(
+   nodesize = c(3, 15, 30),
+   ntree    = c(100, 500)
+ )
```

```
R> rf_grid <- create_grid(
+   base_learner = "surv.rfsrc",
+   grid_params  = rf_tuning_params
+ )

R> rf_grid

[1] "surv.rfsrc_nodesize3_ntree100"
[2] "surv.rfsrc_nodesize15_ntree100"
[3] "surv.rfsrc_nodesize30_ntree100"
[4] "surv.rfsrc_nodesize3_ntree500"
[5] "surv.rfsrc_nodesize15_ntree500"
[6] "surv.rfsrc_nodesize30_ntree500"
```

These dynamically generated learners can then be included directly in the Super Learner library. Because each configuration is treated as an independent base learner, the meta-learning stage automatically identifies the optimal hyperparameter setting through cross-validated risk minimization.

3.4. Variable Screening

High-dimensional survival prediction often involves settings where the number of covariates p is large relative to the sample size n (e.g., molecular profiles or large EHR feature sets). To reduce computational burden and mitigate overfitting, **SuperSurv** supports an optional screening stage that selects a subset of covariates prior to fitting each base learner. This design follows the screening-then-estimation paradigm used in survival Super Learner implementations, where different learners may employ different screening rules.

Let $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ denote the training covariate matrix. A screening algorithm produces a binary inclusion vector

$$\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p) \in \{0, 1\}^p,$$

where $\gamma_j = 1$ indicates that covariate j is retained. The screened design matrix is then $\boldsymbol{X}_\gamma$, consisting of the subset of columns with $\gamma_j = 1$. In **SuperSurv**, screening is performed *within each cross-validation training fold* to avoid information leakage.

Implemented screeners. In addition to the marginal Cox screening and penalized Cox (LASSO/Elastic Net) screening commonly used in survival Super Learner libraries, **SuperSurv** extends the screening library with both supervised and unsupervised alternatives: (i) `screen.all`, which retains all covariates; (ii) `screen.marg`, which ranks features by univariate Cox association; (iii) `screen.glmnet` and `screen.elasticnet`, which fit penalized Cox models and retain covariates with nonzero coefficients; (iv) `screen.rfsrc`, which fits a fast random survival forest and retains covariates with positive variable importance (VIMP); and (v) `screen.var`, which retains the top fraction of covariates by empirical variance, providing an unsupervised filter suitable for noisy high-dimensional inputs.

To ensure stability, each screener enforces a minimum number of retained variables (argument `minscreen`), so that the downstream learner is always estimable even when all features appear weak in finite samples.

Screened learner libraries. A screened library is specified as a list of learner–screener pairs, allowing different learners to use different screening rules:

```
R> my_screen_library <- list(
+   c("surv.coxph",    "screen.all"),
+   c("surv.coxph",    "screen.marg"),
+   c("surv.weibull",  "screen.elasticnet"),
+   c("surv.rfsrc",    "screen.var")
+ )
```

The selected covariates within each fold are stored (optionally) to enable reproducibility and post hoc inspection of the screening behavior.

3.5. Model Evaluation Metrics

SuperSurv provides a unified set of evaluation utilities for comparing the fitted ensemble against each candidate base learner on a common time grid $\mathcal{T} = \{t_1, \ldots, t_m\}$. Given test data $(\widetilde{T}_i, \Delta_i, \boldsymbol{X}_i)_{i=1}^n$ and predicted survival probabilities $\widehat{S}(t \mid \boldsymbol{X}_i)$ evaluated at $t \in \mathcal{T}$, the package computes time-dependent performance curves based on three widely used metrics: the IPCW Brier score, cumulative/dynamic time-dependent AUC, and Uno's C-index.

Time-dependent IPCW Brier score. For each evaluation time $t \in \mathcal{T}$, the IPCW Brier score is computed as an empirical risk over the expanded (i, t) set:

$$\widehat{\mathrm{BS}}(t) = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\Delta_i \mathbf{1}(\widetilde{T}_i \le t)}{\widehat{G}(\widetilde{T}_i \mid \boldsymbol{X}_i)} \left(0 - \widehat{S}(t \mid \boldsymbol{X}_i)\right)^2 + \frac{\mathbf{1}(\widetilde{T}_i > t)}{\widehat{G}(t \mid \boldsymbol{X}_i)} \left(1 - \widehat{S}(t \mid \boldsymbol{X}_i)\right)^2 \right], \tag{8}$$

where $\widehat{G}(\cdot \mid \boldsymbol{X})$ denotes an estimator of the censoring survival function. Lower values indicate better accuracy of the predicted survival probabilities.

Time-dependent AUC. To assess time-varying discrimination, **SuperSurv** computes a cumulative/dynamic AUC curve across $\mathcal{T}$ based on time-dependent ROC methodology for right-censored outcomes. Conceptually, the AUC at time t measures the ability of the model's risk ranking (equivalently, low $\widehat{S}(t \mid \boldsymbol{X})$) to separate individuals who experience the event by time t from those who remain event-free beyond t, while accounting for censoring.

Uno's C-index. As a complementary measure of concordance that remains well-defined under heavy censoring, **SuperSurv** provides Uno's C-index evaluated at each time point $t \in \mathcal{T}$, producing a longitudinal concordance curve. Higher values indicate better discrimination.

Benchmarking the ensemble against base learners. The primary benchmarking utility, `plot_benchmark()`, constructs longitudinal performance curves for the ensemble and each base learner using predictions returned by `predict()` evaluated on $\mathcal{T}$. This comparison is implemented in a fully model-agnostic manner: any fitted object that supports `predict(object, newdata, new.times)` and returns an $n \times m$ survival probability matrix can be benchmarked under the same interface.

```
R> plot_benchmark(
+   object = fit_supersurv,
+   newdata = X_te,
+   time = test$duration,
+   event = test$event,
+   eval_times = new.times
+ )
```

Visualization of individual survival curves. To facilitate clinical interpretation at the individual level, `plot_predict()` visualizes predicted survival trajectories $\{\widehat{S}(t_j \mid \boldsymbol{X}_i)\}_{j=1}^{m}$ for one or more patients using step-function plots consistent with the discrete evaluation grid.

```
R> preds <- predict(fit_supersurv, newdata = X_te, new.times = new.times)
R> plot_predict(preds = preds, eval_times = new.times, patient_idx = 1:6)
```

Calibration assessment at a fixed time horizon. In addition to discrimination and overall accuracy, **SuperSurv** includes a pragmatic calibration diagnostic via `plot_calibration()` at a user-specified horizon $t^\star$. The function partitions individuals into quantile bins of predicted survival $\widehat{S}(t^\star \mid \boldsymbol{X})$, then compares the mean predicted survival in each bin to the observed Kaplan–Meier estimate at $t^\star$ within that bin. The resulting reliability curve is plotted against the 45° line representing perfect calibration.

```
R> plot_calibration(
+   object = fit_supersurv,
+   newdata = X_te,
+   time = test$duration,
+   event = test$event,
+   eval_time = 150,
+   bins = 5
+ )
```

3.6. Model Interpretation and Explainability

To support transparent reporting and clinical interpretability, **SuperSurv** provides a unified explainability layer for both ensemble models and individual base learners. The primary goal is to explain a *scalar risk functional* derived from a fitted survival model, denoted $r(X) = \phi\Big(\widehat{S}(\cdot \mid X)\Big)$, where $\phi(\cdot)$ may represent, for example, risk at a prespecified horizon $t^\star$ or another user-defined summary of the predicted survival curve. This design allows model-agnostic explanations even when base learners differ in their internal parameterization.

Kernel SHAP for global and local explanations. The function `explain_kernel()` interfaces with the **fastshap** package to compute Kernel SHAP values for $r(X)$ using Monte Carlo Shapley estimation with a user-specified background dataset. For a fitted model $r(\cdot)$ and

feature set $\{1, \ldots, p\}$, SHAP decomposes the prediction into additive feature contributions

$$r(x) = \mathbb{E}\{r(X)\} + \sum_{j=1}^{p} \varphi_j(x),$$

where $\varphi_j(x)$ is the Shapley value for feature j at observation x.

Weighted SHAP for ensembles. For a convex Super Learner ensemble, **SuperSurv** optionally computes SHAP values for each *active* base learner and aggregates them using the estimated meta-learner weights. Specifically, if the explained risk functional is linear in the library predictions,

$$r_{\text{ens}}(X) = \sum_{k=1}^{K} \alpha_k \, r_k(X), \qquad \alpha_k \geq 0, \; \sum_{k=1}^{K} \alpha_k = 1,$$

then the linearity property of Shapley values implies $\varphi_{j,\text{ens}}(x) = \sum_{k=1}^{K} \alpha_k \, \varphi_{j,k}(x)$. In practice, **SuperSurv** computes Kernel SHAP for each base learner k and returns the weight-averaged SHAP matrix, enabling global importance ranking (mean $|\varphi_j|$), cohort-level summary plots (beeswarm), and patient-level explanations (waterfall plots).

```
R> # Global Kernel SHAP for an ensemble (weighted across active learners)
R> shap_vals <- explain_kernel(model = fit_sl,
+                              X_explain = X_test[1:50, ],
+                              X_background = X_train[1:100, ],
+                              nsim = 20)
R> plot_global_importance(shap_vals, top_n = 10)
R> plot_patient_waterfall(shap_vals, patient_index = 1, top_n = 10)
```

Time-dependent explainability via survex. Because survival prediction is intrinsically time-indexed, **SuperSurv** also provides `explain_survex()`, which constructs a standard **survex** explainer object by supplying a unified survival-prediction wrapper. This bridge unlocks the full **survex** ecosystem for time-dependent diagnostics, including dynamic permutation importance, time-indexed partial dependence profiles, and local SurvSHAP(t) explanations of how covariates shift the predicted survival trajectory across a prespecified grid of times.

```
R> surv_explainer <- explain_survex(fit_sl, data = X_test, y = Surv(time, event), times =
R> library(survex)
R> plot(model_parts(surv_explainer))
```

3.7. Adjusted Marginal Contrasts via Restricted Mean Survival Time (RMST)

Although hazard ratios from proportional hazards models remain a common reporting standard, they can be difficult to interpret on an absolute time scale and may be unstable under non-proportional or crossing hazards. To provide an estimand on the survival-time scale,

SuperSurv implements restricted mean survival time (RMST) contrasts using g-computation (standardization) on top of the fitted Super Learner survival model.

For a prespecified horizon τ, the individual-level RMST is defined as the area under the survival curve,

$$\mathrm{RMST}(\tau \mid X, A) \;=\; \int_0^\tau S(t \mid X, A)\, dt, \tag{9}$$

which **SuperSurv** evaluates numerically on the prediction grid using a left Riemann sum. Given a binary exposure or treatment indicator $A \in \{0, 1\}$ included among the covariates, **SuperSurv** estimates the difference in RMST using standardization (g-computation). Specifically, the causal estimand average treatment effect (ATE) is

$$\mathrm{ATE}_{\mathrm{RMST}}(\tau) = \mathbb{E}_X \left[\int_0^\tau S(t \mid X, A = 1)\, dt - \int_0^\tau S(t \mid X, A = 0)\, dt \right]. \tag{10}$$

Standardized RMST contrast. Let $A \in \{0, 1\}$ denote a binary exposure or treatment indicator included among the model covariates. **SuperSurv** estimates the standardized mean RMST under two counterfactual interventions by (i) setting $A = 1$ for all individuals, predicting $\widehat{S}_{\mathrm{SL}}(t \mid X, A = 1)$, and integrating to obtain $\widehat{\mathrm{RMST}}_i(\tau, 1)$; (ii) repeating with $A = 0$ to obtain $\widehat{\mathrm{RMST}}_i(\tau, 0)$. The resulting marginal contrast is

$$\widehat{\Delta\mathrm{RMST}}(\tau) = \frac{1}{n} \sum_{i=1}^{n} \widehat{\mathrm{RMST}}_i(\tau, 1) - \frac{1}{n} \sum_{i=1}^{n} \widehat{\mathrm{RMST}}_i(\tau, 0). \tag{11}$$

When A corresponds to a manipulable intervention, $\widehat{\Delta\mathrm{RMST}}(\tau)$ may be interpreted as an average treatment effect (ATE) under standard identification assumptions (consistency, positivity, and no unmeasured confounding). For non-manipulable attributes (e.g., biomarkers), the same procedure yields an adjusted marginal contrast.

```
R> # Adjusted RMST contrast via standardization (G-computation)
R> rmst_res <- estimate_marginal_rmst(
+   fit = fit_sl,
+   data = clinical_data,
+   trt_col = "treatment_status",
+   times = prediction_grid,
+   tau = 150
+ )
R> rmst_res$ATE_RMST
```

Effect curves and diagnostics. To assess how the absolute contrast evolves with the restriction time, **SuperSurv** provides `plot_marginal_rmst_curve()`, which evaluates $\widehat{\Delta\mathrm{RMST}}(\tau)$ over a user-specified grid of τ values. For a simple model-checking diagnostic, `plot_rmst_vs_obs()` compares each individual's predicted RMST against their observed follow-up time, with visual stratification by event status.

4. Empirical Application: METABRIC Breast Cancer Data

To demonstrate an end-to-end survival prediction workflow, we apply **SuperSurv** to the METABRIC breast cancer dataset included with the package. This example highlights three core strengths of the framework: (i) a unified API for training heterogeneous survival learners under right censoring, (ii) automated ensemble construction with time-dependent benchmarking, and (iii) downstream interpretability and clinically meaningful marginal contrasts using RMST.

4.1. Data Preparation and Reproducible Initialization

We begin with a standard 70%/30% train–test split. Predictors are selected as the covariate block indexed by the prefix `x`, and evaluation times are defined on a clinically interpretable grid.

```
R> library(SuperSurv)
R> library(survival)
R> data("metabric", package = "SuperSurv")
R> set.seed(42)
R> train_idx <- sample(seq_len(nrow(metabric)), size = 0.7 * nrow(metabric))
R> train <- metabric[train_idx, ]
R> test  <- metabric[-train_idx, ]
R> x_cols <- grep("^x", names(metabric), value = TRUE)
R> X_tr <- train[, x_cols]
R> X_te <- test[, x_cols]
R> new.times <- seq(20, 150, by = 10)
```

4.2. Library Construction and Hyperparameter Tuning

We construct an event-model library that spans interpretable parametric and semiparametric baselines as well as a flexible nonparametric learner. In particular, we include (i) a Cox model (`surv.coxph`), (ii) a Weibull model (`surv.weibull`), and (iii) a tuned family of Random Survival Forests generated automatically via `create_grid()`. For the censoring model used for IPCW, we use a Cox model for stability.

```
R> rf_grid <- create_grid(
+   base_learner = "surv.rfsrc", grid_params  = list(nodesize = c(5, 20), ntree = 100)
+ )
R> event_lib <- c("surv.coxph", "surv.weibull", rf_grid)
R> cens_lib  <- c("surv.coxph")
```

4.3. Ensemble Training and Weight Inspection

We fit the meta-learner using K-fold cross-validation. To enable downstream interpretability and RMST-based contrasts that require refitting/prediction under counterfactual covariate settings, we save the fitted base-learner library via `control = list(saveFitLibrary = TRUE)`. If computational resources permit, cross-validation can be parallelized using **future**; here we illustrate a lightweight configuration.

```
R> fit_sl <- SuperSurv(
+   time = train$duration, event = train$event,
+   X = X_tr, newdata = X_te,
+   new.times = new.times,
+   event.library = event_lib, cens.library  = cens_lib,
+   selection = "ensemble",
+   control = list(saveFitLibrary = TRUE),
+   nFolds = 5,
+   verbose = FALSE
+ )
R> round(fit_sl$event.coef, 3)
```

```
surv.coxph_screen.all                          surv.weibull_screen.all
                                    0.632                                    0.000
surv.rfsrc_nodesize5_ntree100_screen.all     surv.rfsrc_nodesize20_ntree100_screen.all
                                    0.198                                    0.169
```

The resulting weight vector provides a transparent summary of which model classes contribute most to out-of-sample predictive performance in this dataset.

4.4. Time-Dependent Predictive Benchmarking

We evaluate the fitted ensemble on the held-out test set using the package's built-in benchmarking utilities, which compute time-dependent prediction metrics (e.g., IPCW Brier score, time-dependent AUC, and Uno's C-index) on a common time grid.

```
R> plot_benchmark(
+   object = fit_sl, newdata = X_te,
+   time  = test$duration, event = test$event,  eval_times = new.times
+ )
```

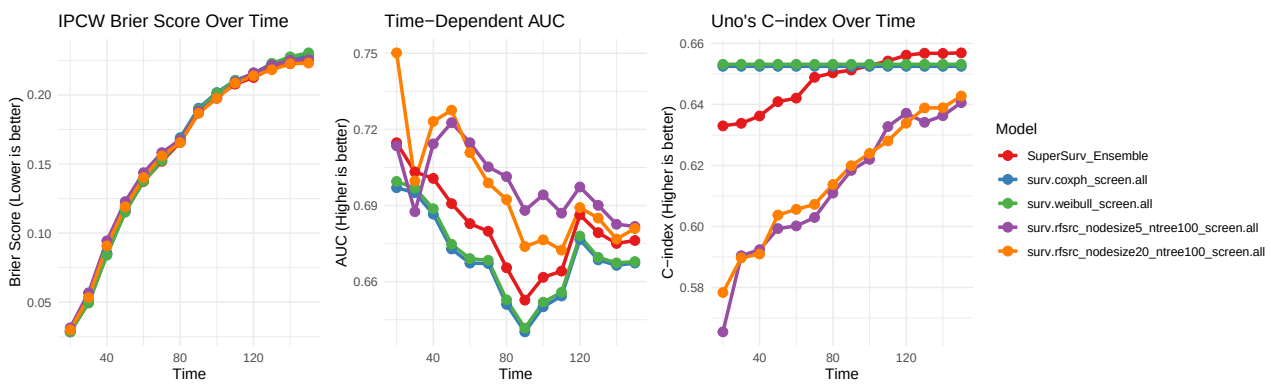

Figure 2: Time-dependent benchmarking on the METABRIC test set comparing the **SuperSurv** ensemble with each base learner.

4.5. Explainable AI via Kernel SHAP

To provide model transparency, we compute Kernel SHAP values for a subset of test patients using a subset of training patients as a reference distribution. Because **SuperSurv** forms an ensemble of base learners, SHAP values can be computed at the learner level and aggregated according to the fitted ensemble weights.

```
R> shap_vals <- explain_kernel(
+   model = fit_sl,
+   X_explain     = X_te[1:50, ],
+   X_background = X_tr[1:100, ],
+   nsim = 10
+ )
R> plot_beeswarm(shap_vals, data = X_te[1:50, ], top_n = 8)
```

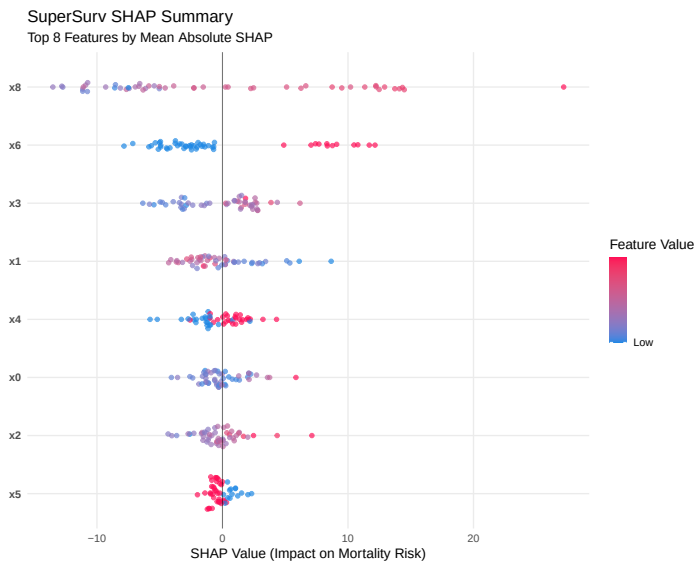

Figure 3: SHAP beeswarm plot for the fitted **SuperSurv** ensemble showing the direction and magnitude of the top contributing features to mortality risk.

4.6. Covariate-Adjusted Marginal Contrast via RMST

Finally, we illustrate how the fitted ensemble can be used to report an absolute, clinically interpretable contrast on the survival-time scale using restricted mean survival time (RMST). Treating `x4` as a binary exposure (e.g., a biomarker), we estimate the covariate-adjusted marginal contrast in RMST up to $\tau = 100$.

```
R> rmst_results <- estimate_causal_rmst(
+   fit  = fit_sl,
+   data = metabric,
+   trt_col = "x4",
+   times = new.times,
```

```
+   tau = 100
+ )
R> rmst_results$ATE_RMST

[1] -1.032961

R> plot_marginal_rmst_curve(
+   fit = fit_sl,
+   data = metabric,
+   trt_col = "x4",
+   times = new.times,
+   tau_seq = seq(20, 140, by = 20)
)
```

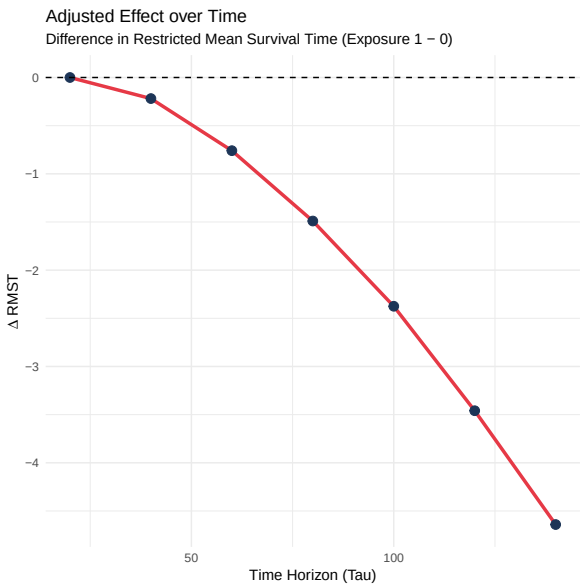

Figure 4: SHAP beeswarm plot for the fitted **SuperSurv** ensemble showing the direction and magnitude of the top contributing features to mortality risk.

This RMST-based contrast summarizes the effect on an absolute time scale and can remain meaningful under non-proportional hazards patterns where hazard ratios may be difficult to interpret.

Together, these steps demonstrate how **SuperSurv** enables a complete survival modeling workflow—from ensemble construction and performance benchmarking to explainable AI and clinically interpretable survival contrasts—within a unified software framework.

5. Discussion

This article introduced **SuperSurv**, an R package for ensemble-based survival modeling built on the Super Learner framework. The package provides a unified infrastructure for combining heterogeneous survival algorithms, performing cross-validated risk minimization, and producing time-dependent survival predictions within a standardized API.

A central contribution of **SuperSurv** is the integration of diverse survival model classes—including classical statistical models, tree-based learners, and modern machine learning algorithms—within a unified ensemble architecture. By estimating convex combinations of candidate learners, the Super Learner ensemble adapts to the underlying structure of the data and can achieve strong predictive performance across a wide range of survival modeling scenarios. The framework is flexible and allows users to incorporate automated hyperparameter tuning, feature screening algorithms, and alternative loss functions such as IPCW log-loss.

Beyond prediction accuracy, **SuperSurv** emphasizes interpretability and practical usability. The package provides built-in tools for explainable machine learning, including Kernel SHAP and integration with the **survex** ecosystem for time-dependent explanation methods such as SurvSHAP(t). These tools enable users to investigate both global feature importance and patient-level explanations of survival predictions.

In addition, **SuperSurv** supports downstream survival contrasts based on restricted mean survival time (RMST). By combining Super Learner survival predictions with standardization (g-computation), the package allows researchers to estimate covariate-adjusted differences in expected survival time between groups. This approach provides an interpretable alternative to hazard ratios and remains valid even when the proportional hazards assumption does not hold.

The empirical application using the METABRIC breast cancer dataset illustrates how **SuperSurv** can be used to construct a complete survival analysis workflow, including ensemble construction, predictive benchmarking, model interpretation, and estimation of adjusted survival contrasts. This end-to-end example highlights the package's ability to bridge predictive modeling and clinically interpretable inference.

Several directions remain for future development. Potential extensions include support for competing risks models, multi-state survival processes, and causal inference frameworks based on targeted learning. In addition, further integration with distributed computing frameworks could enable scalable training for extremely large biomedical datasets.

Overall, **SuperSurv** provides a flexible and extensible toolkit for modern survival analysis, combining the statistical guarantees of Super Learner with practical tools for prediction, interpretation, and clinically meaningful survival contrasts.

References

Binder H, Schumacher M (2008). "Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models." *BMC bioinformatics*, **9**(1), 14.

Chen T, Guestrin C (2016). "Xgboost: A scalable tree boosting system." In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Cox DR (1972). "Regression models and life-tables." *Journal of the royal statistical society: Series B (methodological)*, **34**(2), 187–202.

Curtis C, Shah SP, Chin SF, Turashvili G, Rueda OM, Dunning MJ, Speed D, Lynch AG, Samarajiwa S, Yuan Y, *et al.* (2012). "The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups." *Nature*, **486**(7403), 346–352.

Fouodo CJ, König IR, Weihs C, Ziegler A, Wright MN (2018). "Support vector machines for survival analysis with R."

Gerds TA, Schumacher M (2006). "Consistent estimation of the expected Brier score in general survival models with right-censored event times." *Biometrical Journal*, **48**(6), 1029–1040.

Golmakani MK, Polley EC (2020). "Super learner for survival data prediction." *The International Journal of Biostatistics*, **16**(2).

Graf E, Schmoor C, Sauerbrei W, Schumacher M (1999). "Assessment and comparison of prognostic classification schemes for survival data." *Statistics in medicine*, **18**(17-18), 2529–2545.

Greenwell B (2024). *fastshap: Fast Approximate Shapley Values*. R package version 0.1.1, https://bgreenwell.github.io/fastshap/, URL https://github.com/bgreenwell/fastshap.

Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS (2008). "Random survival forests."

Jaeger BC, Welden S, Lenoir K, Speiser JL, Segar MW, Pandey A, Pajewski NM (2024). "Accelerated and interpretable oblique random survival forests." *Journal of Computational and Graphical Statistics*, **33**(1), 192–207.

Kvamme H, Borgan Ø, Scheel I (2019). "Time-to-event prediction with neural networks and Cox regression." *Journal of machine learning research*, **20**(129), 1–30.

Martinussen T, Vansteelandt S (2013). "On collapsibility and confounding bias in Cox and Aalen regression models." *Lifetime data analysis*, **19**(3), 279–296.

Munch A, Gerds TA (2024). "The state learner–a super learner for right-censored data." *arXiv preprint arXiv:2405.17259*.

Polley EC, Rose S, van der Laan MJ (2011). "Super Learning." In MJ van der Laan, S Rose (eds.), *Targeted Learning: Causal Inference for Observational and Experimental Data*, 1 edition, chapter 3, pp. 43–66. Springer, New York.

Polley EC, van der Laan MJ (2011). "Super Learning for Right-Censored Data." In MJ van der Laan, S Rose (eds.), *Targeted Learning: Causal Inference for Observational and Experimental Data*, 1 edition, chapter 16, pp. 249–258. Springer, New York.

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Sabathe C, Foucher Y (2024). "survivalSL: an R Package for Predicting Survival by a Super Learner." *R JOURNAL*, **16**(4), 4–22.

Simon N, Friedman JH, Hastie T, Tibshirani R (2011). "Regularization paths for Cox's proportional hazards model via coordinate descent." *Journal of statistical software*, **39**, 1–13.

Sonabend R, Király FJ, Bender A, Bischl B, Lang M (2021). "mlr3proba: an R package for machine learning in survival analysis." *Bioinformatics*, **37**(17), 2789–2791.

Sparapani RA, Logan BR, McCulloch RE, Laud PW (2016). "Nonparametric survival analysis using Bayesian additive regression trees (BART)." *Statistics in medicine*, **35**(16), 2741–2753.

Spytek M, Krzyziński M, Langbein SH, Baniecki H, Wright MN, Biecek P (2023). "survex: an R package for explaining machine learning survival models." *Bioinformatics*, **39**(12), btad723.

Štrumbelj E, Kononenko I (2014). "Explaining prediction models and individual predictions with feature contributions." *Knowledge and information systems*, **41**(3), 647–665.

Therneau TM, Grambsch PM (2000). "The cox model." In *Modeling survival data: extending the Cox model*, pp. 39–77. Springer.

Uno H, Claggett B, Tian L, Inoue E, Gallo P, Miyata T, Schrag D, Takeuchi M, Uyama Y, Zhao L, *et al.* (2014). "Moving beyond the hazard ratio in quantifying the between-group difference in survival analysis." *Journal of clinical Oncology*, **32**(22), 2380–2385.

Van der Laan MJ, Polley EC, Hubbard AE (2007). "Super learner."

Westling T, Luedtke A, Gilbert PB, Carone M (2024). "Inference for treatment-specific survival curves using machine learning." *Journal of the American Statistical Association*, **119**(546), 1541–1553.

Wolock CJ, Gilbert PB, Simon N, Carone M (2024). "A framework for leveraging machine learning tools to estimate personalized survival curves." *Journal of Computational and Graphical Statistics*, **33**(3), 1098–1108.

Wood SN (2017). *Generalized additive models: an introduction with R*. chapman and hall/CRC.

Wright MN, Ziegler A (2017). "ranger: A fast implementation of random forests for high dimensional data in C++ and R." *Journal of statistical software*, **77**, 1–17.

Affiliation:

Yue Lyu
Department of Biostatistics
The University of Texas MD Anderson Cancer Center
E-mail: ylyu1@mdanderson.org

Xuelin Huang
Department of Biostatistics
The University of Texas MD Anderson Cancer Center
E-mail: xlhuang@mdanderson.org

Steven Hsesheng Lin
Department of Biostatistics
The University of Texas MD Anderson Cancer Center
E-mail: shlin@mdanderson.org

Ziyi Li
Department of Biostatistics
The University of Texas MD Anderson Cancer Center
E-mail: zli16@mdanderson.org

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd