concrete R Paper: Title TBD

by David Chen, Thomas Gerds, Helene Rytgaard, Maya L. Petersen, Mark van der Laan, ...

Abstract

This article introduces the R package **concrete** (CONtinuous-time Competing Risks Estimation using TmlE) implementing a recently developed targeted maximum likelihood-based estimator (TMLE) targeting the cause-specific absolute risks of time-to-event outcomes measured in continuous time. This package can be used to estimate the effects of static and dynamic interventions on a binary treatment given at baseline, quantified as causally-interpretable absolute risks, risk differences, and risk ratios. Cause-specific hazards are estimated by cross-validated Super Learner ensembles of Cox regressions, which are then used to compute g-formula plug-in and TMLE point estimates of absolute risks. Influence curve-based asymptotic inference is provided for TMLE estimates and simultaneous confidence bands can be computed for target estimands that span multiple multiple times or events. In this paper we review one-step continuous-time TMLE methodology as it is situated in the larger causal inference targeted learning workflow, describe how it is implemented in **concrete**, and demonstrate its use on the PBC dataset.

1 Introduction

In biomedical survival applications, study subjects are often susceptible to competing risks such as all-cause mortality. In recent decades, several competing risk methods have been developed; including the Fine-Gray (Fine and Gray (1999)), pseudovalue (Klein and Andersen (2005)), and direct binomial (Scheike et al. (2008), Gerds et al. (2012)) regressions; and authors have consistently cautioned against the use of standard survival estimands for causal questions involving competing risks. Nevertheless, reviews of clinical literature by Koller et al. (2012) and Austin and Fine (2017) found that most trials still fail to adequately address the effect of potential competing risks in their studies. Meanwhile, formal causal inference frameworks (Holland (1986), Pearl et al. (2016)) have gained recognition for their utility in translating clinical questions into statistical analyses and the targeted maximum likelihood estimation (TMLE) (Laan and Rose (2011), Laan and Rose (2018)) methodology developed from the estimating equation and one-step estimator lineage of semi-parametric efficient methods based on solving efficient influence curves (EICs). The targeted learning roadmap (Petersen and van der Laan (2014)) combines these developments into a cohesive causal inference workflow and provides a structured way to think about statistical decisions. In this paper we apply the targeted learning roadmap to an analysis of time-to-event outcomes and demonstrate the R package **concrete**, which implements a recently developed continuous-time TMLE targeting cause-specific absolute risks (Rytgaard et al. (2021b), Rytgaard and van der Laan (2021)).

The one-step continuous-time TMLE implemented in **concrete** can, given identification and regularity assumptions, efficiently estimate the treatment effect of interventions given at baseline. Rytgaard and van der Laan (2021) provides a rigorous explanation of this method and the necessary assumptions for causal identification, but in short this one-step TMLE procedure consists of two stages: 1) an initial estimation of nuisance parameters and 2) a targeted update of the initial estimators to solve the EIC of the target statistical estimand (Laan and Robins (2003), Kennedy (2016)). In **concrete** this initial nuisance parameter estimation is performed using Super Learning, a cross-validated machine learning ensemble algorithm with asymptotic oracle guarantees (Laan and Dudoit (2003), Laan et al. (2007), Polley et al. (2021)). When data-generating mechanisms are unknown, Super Learners with robust candidate libraries and appropriate loss functions often give users the best chance of achieving the convergence rates needed for TMLE's asymptotic properties. The subsequent targeted update is based in semi-parametric efficiency theory; specifically that efficient regular and asymptotically linear (RAL) estimators must have influence curves equal to the efficient influence curve (EIC) of the target statistical estimand (Laan and Rose (2011), Kennedy (2016)). This TMLE update shfits the initial nuisance parameter estimates to solve target EICs, thus recovering normal asymtotic inference even while leveraging the power of flexible machine-learning algorithms for initial estimation. In Section 2.4.1 we outline how Super Learner is used to estimate nuisance parameters in **concrete** while more detailed guidance on how to best specify Superlearner estimators is provided in Phillips et al. (2022). Section 2.4.4 details the subsequent targeted update, with a full description provided in Rytgaard and van der Laan (2021).

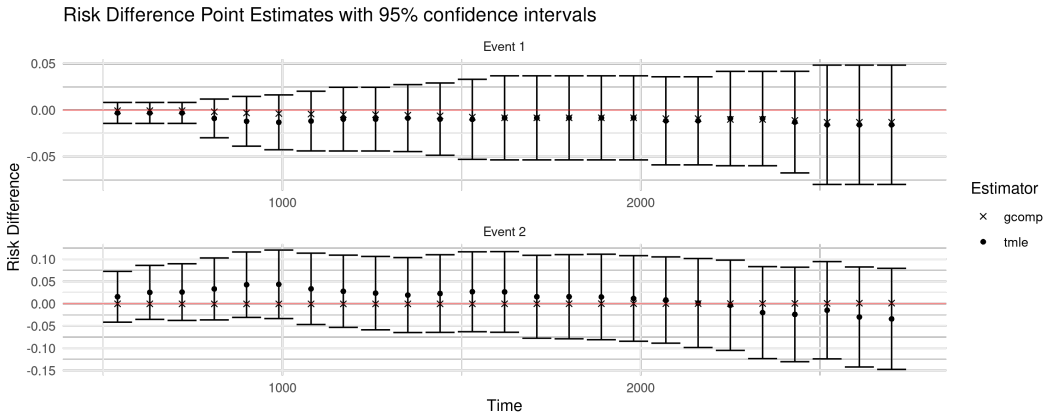
Currently **concrete** can be used for estimands derived from cause-specific absolute risks, such as risk ratios and risk differences, under static and dynamic interventions on binary treatments given at baseline. Estimands can be jointly targeted at multiple times, up to full risk curves over an interval, and for multiple events in cases with competing risks. Methods are available to handle right-censoring, competing risks, and confounding by baseline covariates. Point estimates can be computed using g-formula plug-in or one-step TMLE, and asymptotic inference for the latter is derived from the variance of the efficient influence curve (EIC).

concrete is not meant to be used for left trunctation (i.e. delayed entry), interval censored data, or clustered data. Currently the Super Learners for estimating conditional hazards must be comprised of Cox regressions

though the incorporation of penalized Cox (coxnet) and hazard estimators based on highly adaptive lasso (HAL) are planned in future package versions. Support for stochastic interventions or interventions on multinomial and continuous treatments and support for longitudinal methods to handle time-dependent treatments (e.g. drop-in) and time-dependent confounding are in longer term development.

1.1 A concrete Example: Competing Risks Analysis of the PBC Dataset

```
# Prepare Data
library(concrete)
data <- survival::pbc[, c("time", "status", "trt", "age", "sex", "albumin")]
data <- subset(data, subset = !is.na(data$trt))
data$trt <- data$trt - 1

# Specify Analysis
ConcreteArgs <- formatArguments(DataTable = data,
                EventTime = "time",
                EventType = "status",
                Treatment = "trt",
                Intervention = 0:1,
                TargetTime = 90 * (6:30))

# Compute
ConcreteEst <- doConcrete(ConcreteArgs)

# Return Output
ConcreteOut <- getOutput(ConcreteEst, Estimand = "RD", Simultaneous = FALSE)
plot(ConcreteOut, ask = FALSE)
```



1.2 Other packages

concrete is the first R package for general use implementing a continous-time TMLE, but several existing packages implement discrete-time TMLEs for survival estimands; namely **ltmle** (Schwab et al. (2020)), **stremr** (Sofrygin et al. (2017)), and **survtmle** (Benkeser and Hejazi (2019)). These packages either natively or can be adapted to perform TMLE for absolute risks of right-censored survival or competing risks estimands; **ltmle** and **stremr** use the method of iterated expectations while **survtmle** can target the hazard-based survival formulation. Notably these packages all operate in discrete-time and would require discretization of continuous-time data which can negatively impact estimator performance.

As for existing continuous-time survival estimation packages, **concrete** adds a TMLE method to a small number of packages implementing semi-parametric efficient survival estimators. The *Causal Inference* CRAN Task View lists **riskregression** (Gerds et al. (2022)) as estimating treatment effect estimands in survival settings. **riskregression** implements the IPTW and double-robust AIPTW estimators as well as a g-formula plug-in. None of the packages listed on the *Survival* CRAN Task View are described as implementing efficient semi-parametric estimators for survival estimands, though available via Github are the R packages adjustedCurves (Denz et al. (2022)) and CFsurvival (Westling et al. (2021)), which implement the AIPTW and a cross-fitted doubly-robust estimator respectively.

1.3 Structure of this manuscript

This article is written for readers wishing to use the **concrete** package for their own analyses and for readers interested in an applied introduction to the one-step continuous-time TMLE method described in Rytgaard and van der Laan (2021). Section 2 outlines the targeted learning approach to time-to-event causal effect estimation, with subsection 2.4 providing details on our one-step TMLE implementation. Usage of the **concrete** package and its features is then provided in Section 3, using the example of a simple competing risks analysis of the PBC dataset.

2 Theoretical Framework

2.1 The Targeted Learning Roadmap

At a high level, the targeted learning roadmap for analyzing continuous-time survival or competing risks consists of:

1. Specifying the causal model and defining a causal estimand (e.g. causal risk difference). Considerations include defining a time zero and time horizon, specifying the intervention (i.e. treatment) variable and the desired intervention(s), and specifying the target time(s) and event(s) of interest.
2. Defining a statistical model and statistical estimand, and evaluating the assumptions necessary for the statistical estimand to identify the causal estimand. Considerations include identifying confounding and right-censoring variables, establishing positivity for desired interventions, and formalizing knowledge about the statistical model (e.g. dependency structures or functional structures such as proportional hazards).
3. Performing estimation and providing inference. Considerations include prespecification to avoid misleading inference, selecting an estimator with desirable theoretical properties (e.g. consistency and efficiency within a desired class), and assessing via outcome-blind simulations the estimator's robustness and suitability for the data at hand.

In the following sections we discuss these three stages in greater detail.

2.2 The Causal Model: Counterfactuals, Interventions, and Causal Estimands

With time-to-event data, typical counterfactual outcomes are how long it would take for some events to occur if subjects were hypothetically to receive some intervention. Let A be this intervention variable and let d be the intervention rule, i.e. the function that assigns values to A. The simplest interventions are static rules setting A to some value a in the space of treatment values $\mathcal{A}$, while more flexible dynamic treatment rules might assign treatments based on subjects' baseline covariates, and stochastic treatment rules can incorporate randomness and may even depend on the natural treatment assignment mechanism in so-called modified treatment policies. No matter the type of intervention, if we let d represent the intervention rule then the associated counterfactual survival data $X \sim P^d$ might take the form

$$X = \left(T^d, \Delta^d, A^d, W\right) \tag{1}$$

where $T^d \in (0, t_{max}]$ is earliest occurence of any of the J events under intervention d, $\Delta^d \in \{1, \ldots, J\}$ shows which of the J events occured first under intervention d, and A^d is the treatment value that would be assigned under intervention d. Notably, we do not include censoring in this counterfactual data and instead isolate just those events that experimenters would like to observe in their ideal hypothetical experiment. For ideal experiments tracking just one event, the causal setting is one of classic survival; if instead mutually exclusive events would be allowed to compete, then the causal setting is one with competing risks.

With the counterfactual data defined, causal estimands can then be specified as functions of the counterfactual data. For instance, if we were interested in effects of interventions d_0 versus d_1 on time-to-event outcomes, the counterfactual data $\tilde{X} \sim P^{0,1}$ might be represented as

$$\tilde{X} = \left(T^{d_0}, \Delta^{d_0}, A^{d_0}, T^{d_1}, \Delta^{d_1}, A^{d_1}, W\right)$$

We could then define estimands such as the causal event j relative risks at time t

$$\Psi_{F_{j,t}}(P^{0,1}) = \frac{P(T^{d_1} \leq t, \Delta^{d_1} = j)}{P(T^{d_0} \leq t, \Delta^{d_0} = j)} \tag{2}$$

These estimands may be of interest at a single timepoint or at many, including full curves over a time interval $(\Psi_{F_{j,t}}(P^{0,1}) : t \in (0, t_{max}])$, and in the case of competing risks may involve multiple events $(\Psi_{F_{j,t}}(P^{0,1}) : j \in$

$1, \ldots, J$). In any case, once the desired causal quantity of interest has been expressed as a function of the counterfactual data, efforts can then be made to identify the causal estimand with a function of observed data, i.e. a statistical estimand.

2.3 Observed Data, Identification, and Statistical Estimands

Observed time-to-event data $O \sim P_0$ with J competing events can be represented as:

$$O = \left(\widetilde{T},\, \widetilde{\Delta},\, A,\, W\right) \tag{3}$$

where $\widetilde{T} \in (0, t_{max}]$ is the earlier of the first event time T or the right-censoring time C, $\widetilde{\Delta} \in \{0, \ldots, J\}$ indicates which event occurs (with 0 indicating right-censoring), A is the observed treatment and W is the set of baseline covariates.

To link causal estimands such as Eq. (2) to statistical estimands, we need a set of untestable identification assumptions to hold: consistency, positivity, no unmeasured confounding, and conditionally independent censoring. Readers can find a full discussion of these identification assumptions for absolute risk estimands in Section 3 of Rytgaard et al. (2021a). Given these assumptions, we can identify the cause-j absolute risk at time t under intervention d using the g-computation formula as

$$P(T^d \leq t, \Delta^d = j) = \mathbb{E}_{\mathcal{W}}\left[\int_{\mathcal{A}} F_j(t \mid a,\, w)\, \pi^*(a \mid w)\, da\right] \tag{4}$$

where $\pi^*(a \mid w)$ is the treatment propensity implied by the intervention d and $F_j(t \mid a,\, w)$ is the conditional cause-j absolute risk

$$F_j(t \mid a,\, w) = \int_0^t \lambda_j(s \mid a,\, w)\, S(s\text{-} \mid a,\, w)\, ds$$

with the cause-j conditional hazard

$$\lambda_j(t \mid a,\, w) = \lim_{h \to 0} \frac{1}{h} P(\widetilde{T} \leq t + h,\, \widetilde{\Delta} = j \mid \widetilde{T} \geq t,\, a,\, w)$$

and conditional event-free survival

$$S(t \mid a,\, w) = \exp\left(-\int_0^t \sum_{j=1}^{J} \lambda_j(s \mid a,\, w)\, ds\right) \tag{5}$$

From Eq (4), it follows that we can identify the causal cause-j relative risk (2) at time t by

$$\Psi_{F_{j,t}}(P_0) = \frac{\mathbb{E}_{\mathcal{W}}\left[\int_{\mathcal{A}} F_j(t \mid a,\, w)\, \pi^*_{d_1}(a \mid w)\, da\right]}{\mathbb{E}_{\mathcal{W}}\left[\int_{\mathcal{A}} F_j(t \mid a,\, w)\, \pi^*_{d_0}(a \mid w)\, da\right]} \tag{6}$$

where $\pi^*_{d_0}$ and $\pi^*_{d_1}$ represent the treatment propensities implied by treatment rules d_0 and d_1 respectively.

It should be noted that even without the identification assumptions for causal inference, statistical estimands such as Eq. (6) may still have valuable interpretations as standardized measures isolating the importance of the "intervention" variable (Laan (2006)).

2.4 Targeted Estimation

The TMLE procedure for estimands derived from cause-specific absolute risks begins with estimating the treatment propensity π, the conditional hazard of censoring λ_c and the conditional hazards of events $\lambda_j : j = 1, \ldots, J$. In **concrete** these nuisance parameters are estimated using the Super Learner algorithm, which involves specifying a cross-validation scheme, compiling a library of candidate algorithms, and designating a cross-validation loss function and a Super Learner meta-learner.

Specifying Super Learners

For a simple V-fold cross validation setup, let $\mathcal{Q}_n = \{O_i\}_{i=1}^n \sim P_n$ be the observed n i.i.d observations of $O \sim P_0$ and let $B_n \in \{1, ..., V\}^n$ be a random vector that assigns the n observations into V validation folds. Then for each v in $1, ..., V$ we define a training set $\mathcal{Q}_v^{\mathcal{T}} = \{O_i : B_n^i = v\} \sim P_v^{\mathcal{T}}$ and corresponding validation set $\mathcal{Q}_v^{\mathcal{V}} = \{O_i : B_n^i \neq v\} \sim P_v^{\mathcal{V}}$.

Having specified a cross-validation scheme, the next steps are to construct the Super Learner candidate

library, define an appropriate loss function, and select a Super Learner meta-learner. Super Learner libraries should be comprised of candidate algorithms that range in flexibility while respecting existing data-generating knowledge. For instance, candidate estimators should incorporate covariates and interactions known to be predictive of outcomes, and if the number of independent observations n is much greater than the number of covariates, then more highly flexible candidate algorithms such as Highly Adaptive Lasso (HAL) should be included in the Super Learner library. If on the other hand the number of covariates approaches n, then libraries should be comprised of fewer and less flexible candidate algorithms, potentially with native penalization as with coxnet or by pairing candidate regression algorithms with screening algorithms. It should be noted that using HAL for initial nuisance parameter estimation can achieve the neceesary convergence rates (Laan (2017); Benkeser and Van Der Laan (2016); Rytgaard et al. (2021a)) for TMLE to be efficient. Super Learner loss functions should imply a risk that is minimized by the true data-generating process and define a loss-based dissimilarity tailored to the target parameter and for maximal robustness the discrete selector that simply selects the best performing candidate should be used as the Super Learner meta-learner. For more flexibility, Super Learners using more flexible meta-learner algorithms can be nested as candidates within a larger Super Learner, and additional guidance is provided in Laan et al. (2007) and Chapter 3 of Laan and Rose (2011).

Currently the default cross-validation setup in **concrete** generally follows the guidelines laid out in Phillips et al. (2022), with the number of cross-validation folds increasing with smaller sample sizes. Default Super Learner libraries are provided and will be detailed in the following sections, but should be amended to suit the data at hand and to incorporate subject matter knowledge.

Estimating Treatment Propensity

For estimating the treatment propensity let π_0 be the true conditional distribution of A given W, let $\mathcal{M}_\pi = \{\hat{\pi} : P_n \to \hat{\pi}(P_n)\}$ be the library of candidate propensity score estimators, and let L_π be a loss function such that the risk $\mathbb{P}_0 L_\pi(\pi) \equiv \mathbb{E}_0 [L_\pi(\pi, O)]$ is minimized by π_0. The discrete Super Learner estimator is then the candidate propensity estimator with minimal cross validated risk,

$$\hat{\pi}^{SL} = \underset{\hat{\pi} \in \mathcal{M}_\pi}{\operatorname{argmin}} \sum_{v=1}^{V} \mathbb{P}_{Q_v^{\mathcal{V}}} L_\pi(\hat{\pi}(P_v^{\mathcal{T}})) \tag{7}$$

where $\hat{\pi}(P_v^{\mathcal{T}})$ are candidate propensity score estimators trained on data $Q_v^{\mathcal{T}}$. Currently **concrete** uses the default **SuperLearner** loss functions and specifies a default library consisting of glmnet and xgboost.

Estimating Conditional Hazards

For $\delta = 0, \ldots, J$ where $(\delta = 0)$ is censoring and $(\delta \in \{1, \ldots, J\})$ are outcomes of interest, let $\lambda_\delta : \delta = 0, \ldots, J$ be the true conditional hazards, let $\mathcal{M}_\delta = \{\hat{\lambda}_\delta : P_n \to \hat{\lambda}_\delta(P_n)\}$ be the libraries of candidate estimators, and let L_δ be loss functions such that the risks $\mathbb{P}_0 L_\delta(\cdot)$ are minimized by the true conditional hazards λ_δ. The discrete Super Learner selectors for each δ then chooses the candidate which has minimal cross validated risk

$$\hat{\lambda}_\delta^{SL} = \underset{\hat{\lambda}_\delta \in \mathcal{M}_\delta}{\operatorname{argmin}} \sum_{v=1}^{V} \mathbb{P}_{Q_v^{\mathcal{V}}} L_\delta(\hat{\lambda}_\delta(P_v^{\mathcal{T}})) \ : \ \delta = 0, \ldots, J \tag{8}$$

where $\hat{\lambda}_\delta(P_v^{\mathcal{T}})$ are candidate event δ conditional hazard estimators trained on data $Q_v^{\mathcal{T}}$. The current **concrete** default is a library of two Cox models, treatment-only and main-terms, with cross-validated risk computed using negative log Cox partial-likelihood loss

$$\mathbb{P}_{Q_v^{\mathcal{V}}} L_\delta(\hat{\lambda}_\delta(P_v^{\mathcal{T}})) = \mathbb{P}_{Q_v^{\mathcal{V}}} L_\delta(\hat{\beta}_{\delta, Q_v^{\mathcal{T}}}) = - \sum_{i:\, O_i \in Q_v^{\mathcal{V}}} \left[\hat{\beta}'_{\delta, Q_v^{\mathcal{T}}} W_i - \log \left[\sum_{h \in \mathcal{R}(\widetilde{T}_i)} \exp(\hat{\beta}'_{\delta, Q_v^{\mathcal{T}}} W_h) \right] \right]$$

where $\mathcal{R}(t)$ is the risk set at time t, $\{h : \widetilde{T}_h \geq t\}$ and $\hat{\beta}'_{\delta, Q_v^{\mathcal{T}}}$ are the coefficients of an event δ candidate Cox regression trained on data $Q_v^{\mathcal{T}}$.

Solving the Efficient Influence Curve

For parameters such as risk ratios which are derived from cause-specific absolute risks, we solve a vector of absolute risk EICs with one element for each combination of target event, target time, and intervention. That is, the EIC for a target parameter involving J competing events, K target times, and M interventions is a $J \times K \times M$ dimensional vector where the component corresponding to the cause-specific risk of event j, at

time t_k, and under intervention propensity π^*_m is:

$$D^*_{m,j,k}(\lambda,\pi,S_c)(O) = \sum_{l=1}^{J} \int h_{m,j,k,l,s}(\lambda,\pi,S_c)(O) \left(N_l(s) - \mathbf{1}(\widetilde{T} \geq s)\,\lambda_l(s \mid A, W)\right) ds + \int_{\mathcal{A}} F_j(t_k \mid A = a, W)\,\pi^*_m(a \mid W)\,da - \Psi_{\pi^*,j,t}(P_0) \quad (9)$$

where $N_l : l = 0, \ldots, J$ are the cause-specific counting processes

$$N_l(s) = \mathbf{1}\left\{\widetilde{T} \leq s, \widetilde{\Delta} = l\right\}$$

and $h_{m,j,k,l,s}(\lambda,\pi,S_c)(O)$ is the TMLE "clever covariate"

$$h_{m,j,k,l,s}(\lambda,\pi,S_c)(O) = \frac{\pi^*_m(A \mid W)\,\mathbf{1}(s \leq t_k)}{\pi(A \mid W)\,S_c(s\text{-} \mid A, W)} \left(\mathbf{1}(l = j) - \frac{F_j(t_k \mid A, W) - F_j(s \mid A, W)}{S(s \mid A, W)}\right) \quad (10)$$

We highlight here that clever covariate is a function of the intervention-defined treatment propensity, the observed intervention-related densities which are unaffected by TMLE targeting, and the observed outcome-related densities which will be updated by TMLE targeting. Note also that notation for the EIC ($D^*_{m,j,k}(\lambda,\pi,S_c)(O)$) and clever covariate ($h_{m,j,k,l,s}(\lambda,\pi,S_c)(O)$) reflect their dependence on P through the outcome-related conditional hazards $\lambda = (\lambda_l : l = 1, \ldots, J)$ and the intervention-related treatment propensity π and conditional censoring survival $S_c(t \mid a, w) = \exp\left(-\int_0^t \lambda_0(s \mid a, w)\,ds\right)$.

The one-step continuous-time survival TMLE involves updating the cause-specific hazards λ along the universally least favorable submodel, which is implemented as recursive limited updates along a sequence locally least favorable submodels. To describe this procedure, let us first introduce the following vectorized notation:

$$D^* = \left(D^*_{m,j,k} : m = 1, \ldots, M,\ j = 1, \ldots, J,\ k = 1, \ldots, K\right)$$
$$h_{l,s} = \left(h_{m,j,k,l,s} : m = 1, \ldots, M,\ j = 1, \ldots, J,\ k = 1, \ldots, K\right)$$

The one-step continuous-time survival TMLE recursively updates the cause-specific hazards in the following manner: starting from $b = 0$, with $\lambda^0_j = \hat{\lambda}^{SL}_j$, and $\lambda^b = \left(\lambda^b_l : l = 1, \ldots, J\right)$

$$\lambda^{b+1}_l = \lambda^b_l \exp\left(\frac{\left\langle \mathbb{P}_n D^*(\lambda^b,\pi,S_c)(O),\ h_{j,s}(\lambda^b,\pi,S_c)(O)\right\rangle}{||\mathbb{P}_n D^*(\lambda^b,\pi,S_c)(O)||}\,\epsilon_b\right), \quad l = 1, \ldots, J \quad (11)$$

where

$$\langle x, y\rangle = x^\top y \quad , \quad ||x|| = \sqrt{x^\top x}$$

and the step sizes ϵ_b are chosen such that

$$||\mathbb{P}_n D^*(\lambda^{b+1},\pi,S_c)(O)|| < ||\mathbb{P}_n D^*(\lambda^b,\pi,S_c)(O)||$$

The recursive update following Eq (11) is completed at the iteration B where

$$\mathbb{P}_n D^*(\lambda^B,\pi,S_c)(O) \leq \frac{\sqrt{\mathbb{P}_n D^*(\lambda^B,\pi,S_c)(O)^2}}{\sqrt{n}\,\log(n)} \quad (12)$$

This updated vector of conditional hazards λ^B is then used to compute a plug-in estimate the target statistical estimand.

Estimating TMLE Variance

In **concrete**, the variance of TMLE estimates of targeted risks is estimated from the EIC's variance divided by the sample size, $\frac{\mathbb{P}_n\, D^*(\lambda^B,\pi,S_c)(O)^2}{n}$, which is a consistent estimator for the variance of asymptotically linear estimators. In the presence of significant positivity violations (which may be seen as propensity scores close to 0), this EIC-derived variance estimator will be anti-conservative and variance estimation by bootstrap may be more reliable. However, bias resulting from positivity violations cannot be remedied in this way, and so other methods of addressing positivity violations (Petersen et al. (2012)) are recommended instead. For multidimensional estimands, simultaneous confidence intervals can be computed by simulating the $1 - \alpha$ quantiles of a multivariate normal distribution with the covariance structure of the estimand EICs.

3 Using concrete

The basic **concrete** workflow consists of calling three functions in sequence: `formatArguments()`, `doConcrete()`, and `getOutput()`. Users specify their estimation problem and desired analysis through `formatArguments()`, which checks the specified analysis for red flags and then produces a `"ConcreteArgs"` environment object. The `"ConcreteArgs"` object is then passed into `doConcrete()` which performs the specified continuous-time one-step survival TMLE and produces a `"ConcreteEst"` object which can be interrogated for diagnostics and estimation details. The `"ConcreteEst"` object can then be passed into `getOutput()` to produce tables and plots of cause-specific absolute risk derived estimands such as risk differences and relative risks.

3.1 formatArguments()

3.2 ConcreteArgs

The arguments of `formatArguments()` are primarily involved in specifying 1) the observed data structure, 2) the target estimand, or 3) the TMLE estimator. The output `"ConcreteArgs"` object is an environment containing these necessary elements of a continuous-time TMLE analysis.

```
ConcreteArgs <- formatArguments(
  DataTable = data,
  EventTime = "time",
  EventType = "status",
  Treatment = "trt",
  Intervention = 0:1,
  TargetTime = 90 * (6:30),
  TargetEvent = 1:2,
  MaxUpdateIter = 500
)
```

Data

Observed data is passed into the `DataTable` argument as either a `data.frame` or `data.table` object, which must contain columns corresponding to the observed time-to-event $\widetilde{T}$, the indicator of which event occured Δ, and the treatment variable A. Any number of columns containing baseline covariates W can also be included. Note that the treatment values in A must be numeric, with binary treatments encoded as 0 and 1. The input data must also be without missingness; imputation of missing covariates should be done prior to passing data into **concrete** while data with missing treatment or outcome values is not supported by **concrete**. If the input data includes a column with uniquely identifying subject IDs, its name should be passed into the `ID` argument; this is for compatibility with planned future functionality for clustered and longitudinal data.

In the PBC example, the observed data is the `data` object, $\widetilde{T}$ is the column `"time"`, Δ is the column `"status"`, A is the column `"trt"`, and covariates L are the remaining columns: (`"age"`, `"sex"`, and `"albumin"`).

Target Estimand: Intervention, Target Events, and Target Times

Static interventions on a binary treatment A setting all observations to $A = 0$ or $A = 1$ can specified with 0, 1, or c(0, 1) if both interventions are of interest, i.e. for contrastive parameters such as risk ratios and risk differences. More complex interventions can be specified with a list containing a pair of functions: an "intervention" function which outputs desired treatment assignments and a "g.star" function which outputs desired treatment probabilities. Dynamic interventions can be passed in as "intervention" functions without an accompanying "g.star" function. These functions can take treatment and covariates as arguments and must produce treatment assignments and probabilities respectively, each with the same dimensions as the observed treatment. The function `makeITT()` creates list of functions corresponding to the binary treat-all and treat-none static interventions, which can be used as a template for specifying more complex interventions.

The `TargetEvent` argument specifies the event types of interest. Event types must be be coded as integers, with non-negative integers reserved for censoring. If `TargetEvent` is left `NULL`, then all positive integer event types in the observed data will be jointly targeted. In the pbc dataset, there are 3 event values encoded by the`status` column: 0 for censored, 1 for transplant, and 2 for death. To analyze pbc with transplants treated as right-censoring, `TargetEvent` should be set to 2, whereas for a competing risks analysis one could either leave `TargetEvent = NULL` or set `TargetEvent = 1:2` as in the above example.

The `TargetTime` argument specifies the times at which the cause-specific absolute risks or event-free survival are estimated. Target times should be restricted to the time range in which target events are observed and `formatArguments()` will return an error if target time is after the last observed failure event time. If no `TargetTime` is provided, then **concrete** will target the last observed event time, though this is likely to result in a highly variable estimate if prior censoring is substantial. The `TargetTime` argument can either be a single number or a vector, as one-step TMLE can target cause-specific risks at multiple times simultaneously. For estimands involving full curves, `TargetTime=` should be set to a fine grid covering the desired interval (Rytgaard et al. (2021b)).

Estimator Specification

The `formatArguments()` arguments involved in estimation are the cross-validation setup `CVArg`, the Super-learner candidate libraries `Model`, the software backends `PropScoreBackend` and `HazEstBackend`, and the practical TMLE implementation choices `MaxUpdateIter`, `OneStepEps`, and `MinNuisance`. Note that `Model` is used in this section in line with common usage in statistical software, rather than to refer to formal statistical or causal models as in preceding sections.

Cross-validation is implemented using `origami::make_folds()`. The default scheme if the `CVArg` argument is left `NULL`, is a stratified V-fold cross-validation following the recommendations in Phillips et al. (2022). Chapter 5 of the online Targeted Learning Handbook lists and demonstrates the specification of several other cross-validation schemes.

Super Learner libraries for estimating nuisance parameters are specified through the `Model` argument. The input should be a named list with an element for the treatment variable and one for each event type including censoring. The list element corresponding to treatment must be named with the column name and the list elements corresponding to each event type must be named for the numeric value of the event type (e.g. "0" for censoring). Any missing specifications will be filled in with defaults, and the resulting list of libraries can be accessed in the output `.[["Model"]]` which can be then edited by the user, as shown below

```
ConcreteArgs$Model <- list(
  "trt" = c("SL.glmnet", "SL.ranger", "SL.xgboost", "SL.glm"),
  "0" = NULL, # will use the default library
  "1" = list(Surv(time, status == 1) ~ trt, ~ .),
  "2" = list("~ trt", "Surv(time, status == 2) ~ .")
)
ConcreteArgs <- formatArguments(ConcreteArgs)
```

In **concrete**, propensity scores are by default estimated using the **SuperLearner** package `PropScoreBackend = "Superlearner"` with candidate algorithms `c("xgboost","glmnet")` implemented by packages **xgboost** and **glmnet**. Alternatively the **sl3** package can be used by specifying `PropScoreBackend = "sl3"`. For further details about these packages, see their respective package documentations.

For estimating the necessary conditional hazards, **concrete** currently relies on a discrete Superlearner consisting of a library of Cox models implemented by `survival::coxph()` evaluated on cross-validated pseudo-likelihood loss. Support for estimation of hazards using coxnet, Poisson-HAL and other methods may be added in the future, but currently the `HazEstBackend` argument must be "coxph". The default Cox specifications are a treatment-only model and a main-terms model with treatment and all covariates. These models can be specified as strings or formulas as can be seen in the above example.

As detailed by Eq. (11) and (12), the one-step TMLE update step involves recursively updating cause-specific hazards, summing along small steps ϵ_b. At default the maximum step size is 0.1, and is halved persistently whenever a step would increase . The `formatArguments(MaxUpdateIter)` argument is provided to provide a definite stop to the recursive TMLE update. The default is 500 and should be sufficient for most applications, but may need to be increased such as when support for targeted estimands in the data is low or when targeting estimands with many components. The argument `formatArguments(MinNuisance)` can be used to specify a lower bound for the product of the propensity score and lagged survival probablity for remaining uncensored; this term is present in the denominator of the efficient influence function and enforcing a lower bound decreases estimator variance at the cost of introducing bias but improving stability.

ConcreteArgs object The `"ConcreteArgs"` output of `formatArguments()` is an environment containing the estimation specification as objects that can be modified by the user. The modified `"ConcreteArgs"` object should then be passed back through `formatArguments()` to check the modified estimation specification.

```
ConcreteArgs$MaxUpdateIter <- 600
ConcreteArgs[["Model"]][["2"]][[3]] <- "~ trt*."
ConcreteArgs <- formatArguments(ConcreteArgs)
```

"ConcreteArgs" objects can be printed to display summary information about the specified estimation problem

```
print(ConcreteArgs, Verbose = TRUE)
```

```
Observed Data (312 rows x 7 cols)
Unique IDs: "ID" (n=312),  Time-to-Event: "time",  Event Type: "status",  Treatment: "trt"

Cens. 0 : n=168 (0.54),  [min,max] = [788, 4556]
Event 1 : n=19 (0.06),  [min,max] = [533, 3092]
Event 2 : n=125 (0.4),  [min,max] = [41, 4191]

Treatment: 0: n=158 (0.51)   1: n=154 (0.49)

3 Baseline Covariates
   ColName CovName CovVal
1:      L1     age      .
2:      L2 albumin      .
3:      L3     sex      f


Target Events: 1, 2
Target Times (n at risk): 540 (117/312), 630 (113/312), 720 (110/312),  ... , 2520 (25/312), 2610 (22/312), 2700 (21/312)
Intervention "A=1":  Trt Assignments = (1,1,1,1,1,1,1,1,1,1 ... ),  Observed Prevalence = 0.49
Intervention "A=0":  Trt Assignments = (0,0,0,0,0,0,0,0,0,0 ... ),  Observed Prevalence = 0.51

Stratified 19-Fold Cross Validation
Trt Pr Estimation (SuperLearner): Default SL Selector, Default Loss Fn, 4 candidates - SL.glmnet, SL.ranger, SL.xgboost, SL.glm
Cens. 0 Estimation (coxph): Discrete SL Selector, Log Partial-LL Loss, 2 candidates - TrtOnly, MainTerms
Event 1 Estimation (coxph): Discrete SL Selector, Log Partial-LL Loss, 2 candidates - model1, model2
Event 2 Estimation (coxph): Discrete SL Selector, Log Partial-LL Loss, 3 candidates - model1, model2, model3

One-step TMLE (finite sum approx.) simultaneously targeting all cause-specific Absolute Risks
g nuisance bounds = [0.04929, 1],  max update steps = 600,  starting one-step epsilon = 0.1
```

In particular, we can see that the target analysis is for competing risks (target events = 1, 2) under interventions "A=1" and "A=0" that assign all subjects to treated and control arms respectively. Objects in the "ConcreteArgs" environment can be interrogated directly for details about any particular aspect of the estimation specification.

3.3 doConcrete()

Adequately specified "ConcreteArgs" objects can then be passed into the doConcrete() function which will then perform the specified TMLE analysis. The output is an object of class "ConcreteEst" which contains TMLE point estimates and influence curves for the cause-specific absolute risks for each targeted event at each targeted time. If the GComp argument is set to TRUE, then a Super Learner-based g-formula plugin estimate of the targeted risks will be included in the output.

```
ConcreteEst <- doConcrete(ConcreteArgs)
```

We have previously reviewed the one-step continuous-time TMLE implementation in Section 2.4, so here we will name the non-exported functions in doConcrete() which perform each of the steps of the one-step continuous-time survival TMLE procedure, in case users wish to explore the implementation in depth.

The cross-validation (Section 2.4.1) is checked and evaluated in formatArguments(), returning fold assignments as .[["CVFolds"]] of the "ConcreteArgs" object.

The initial estimation of nuisance parameters (Sections 2.4.2 and 2.4.3) is performed by the function getInitialEstimate(): getPropScore() estimates propensity scores and getHazEstimate() estimates the conditional hazards.

The one-step TMLE update procedure (Section 2.4.4) is performed by doTmleUpdate() with getEIC() computing the efficient influence curves (9).

ConcreteEst objects

The print method for "ConcreteEst" objects summarizes the estimation target and displays diagnostic information about TMLE update convergence, intervention-related nuisance parameter truncation, and the nuisance parameter Super Learners.

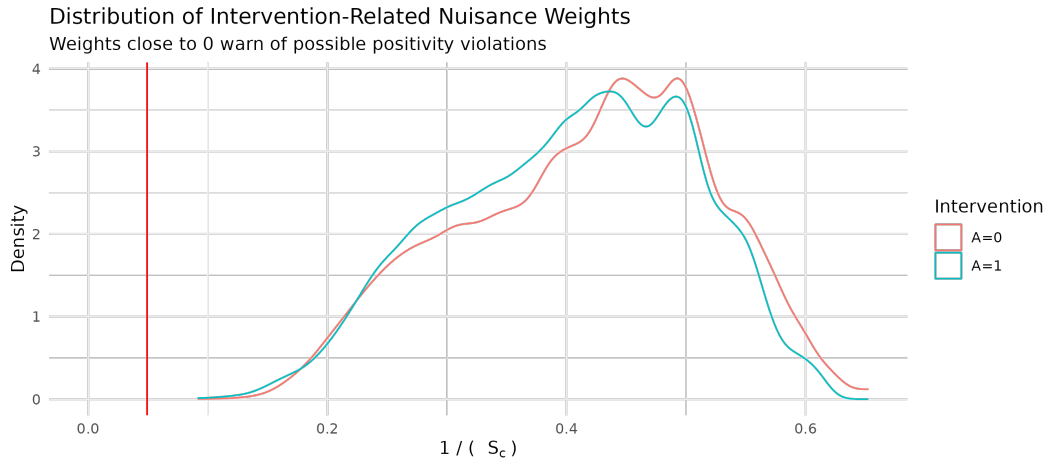
```
print(ConcreteEst, Verbose  = TRUE)
```

```
Continuous-Time One-Step TMLE targeting the Cause-Specific Absolute Risks for:
Interventions: "A=1", "A=0"  |  Target Events: 1, 2  |  Target Times: 540, 630, 720, ... ,2520, 2610, 2700

**TMLE did not converge!!**

   Intervention Time Event   Pt Est        se      PnEIC abs(PnEIC / Stop Criteria)
1:          A=1  810     1 0.005005 0.0004654 -0.0018032                    22.252
2:          A=1  540     1 0.001542 0.0001486 -0.0003761                    14.533
3:          A=1  630     1 0.001542 0.0001486 -0.0003761                    14.533
4:          A=1  720     1 0.001542 0.0001486 -0.0003761                    14.533
5:          A=1  990     1 0.011998 0.0056983 -0.0014028                     1.414

For Intervention "A=1", no subjects had G-related nuisance weights falling below 0.0493
For Intervention "A=0", no subjects had G-related nuisance weights falling below 0.0493

Initial Estimators:
Treatment:
                    Risk       Coef
SL.glmnet_All  0.2494320 0.76338619
SL.ranger_All  0.2527777 0.18669719
SL.xgboost_All 0.2948025 0.04991662
SL.glm_All     0.2506454 0.00000000

Cens. 0:
             Risk Coef
TrtOnly   286.0636    0
MainTerms 285.3721    1

Event 1:
           Risk Coef
model1 44.61230    0
model2 38.61817    1

Event 2:
           Risk Coef
model1 277.4118    0
model2 242.9892    1
model3 246.7695    0
```

If TMLE has not converged, the mean EICs that have not attained the desired cutoff will be displayed in a table. Convergence can be attained by increasing the maximum number of iterations, though as seen above, even very small PnEIC values may not meet the convergence criteria at target times when few events have yet occurred.

The extent of g-related nuisance parameter truncation for each intervention is also reported, both in terms of the percentage of nuisance weights that are truncated and the percentage of subjects that have truncated nuisance weights. and if users suspect possible positivity issues, the plot method for "`ConcreteEst`" objects can be used to visualize the distribution of estimated propensity scores for each intervention, with the red vertical line marking the cutoff for truncation.

```
plot(ConcreteEst, ask  = FALSE)
```



Propensity scores close to 0 indicate the possibility of positivity violations and may warrant re-examining the target time(s), interventions, and covariate adjustment sets. In typical survival applications, positivity issues may arise when targeting times at which some subjects are highly likely to have been censored, or if certain subjects are unlikely to have received a desired treatment intervention. For guidance on handling positivity issues, see Petersen et al. (2012).

Lastly, the candidate estimators of nuisance parameters are summarized with the cross-validated risk of each estimator followed by their weighting in the Super Learner ensemble.

3.4 getOutput()

`getOutput()` takes as an argument the `"ConcreteEst"` object returned by `doConcrete()` and can be used to produce tables and plots of the cause-specific risks, risk differences, and relative risks. By default `getOutput()` returns a `data.table` with point estimates and pointwise standard errors for cause-specific absolute risks, risk differences, and risk ratios. By default, the first listed intervention is used as the "treated" group while the second is considered "control"; other contrasts can be specified via the `Intervention` argument. Below we show a subset of the relative risk estimates produced by the "nutshell" estimation specification for the pbc dataset.

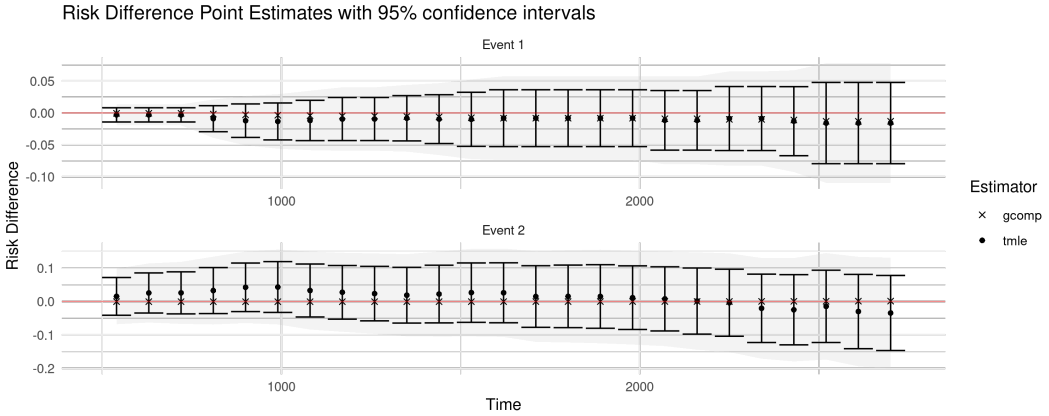
```
ConcreteOut <- getOutput(ConcreteEst = ConcreteEst, Estimand = "RD",
             Intervention = 1:2, GComp = TRUE, Simultaneous = TRUE, Signif = 0.05)
head(ConcreteOut, 12)
```

```
    Time Event  Estimand  Intervention Estimator   Pt Est     se CI Low  CI Hi SimCI Low SimCI Hi
 1:  540     1 Risk Diff [A=1] - [A=0]      tmle -3.1e-03 0.0058 -0.014 0.0083    -0.020    0.014
 2:  540     1 Risk Diff [A=1] - [A=0]     gcomp -6.3e-04     NA     NA     NA       NA       NA
 3:  540     2 Risk Diff [A=1] - [A=0]      tmle  1.5e-02 0.0290 -0.042 0.0730    -0.071    0.100
 4:  540     2 Risk Diff [A=1] - [A=0]     gcomp -4.6e-04     NA     NA     NA       NA       NA
 5:  630     1 Risk Diff [A=1] - [A=0]      tmle -3.1e-03 0.0058 -0.014 0.0083    -0.020    0.014
 6:  630     1 Risk Diff [A=1] - [A=0]     gcomp -6.3e-04     NA     NA     NA       NA       NA
 7:  630     2 Risk Diff [A=1] - [A=0]      tmle  2.5e-02 0.0310 -0.036 0.0860    -0.067    0.120
 8:  630     2 Risk Diff [A=1] - [A=0]     gcomp -5.1e-04     NA     NA     NA       NA       NA
 9:  720     1 Risk Diff [A=1] - [A=0]      tmle -3.1e-03 0.0058 -0.014 0.0083    -0.020    0.014
10:  720     1 Risk Diff [A=1] - [A=0]     gcomp -6.3e-04     NA     NA     NA       NA       NA
11:  720     2 Risk Diff [A=1] - [A=0]      tmle  2.6e-02 0.0330 -0.038 0.0900    -0.071    0.120
12:  720     2 Risk Diff [A=1] - [A=0]     gcomp -5.5e-04     NA     NA     NA       NA       NA
[ reached getOption("max.print") -- omitted 89 rows ]
```

From left to right, the first five columns show the target times, target events, estimands, interventions, and estimators. The following columns show the point estimates, estimated standard error, confidence intervals and simultaneous confidence bands. Desired level of CI coverage is controlled by the `Signif` argument which is set to a default alpha = 0.05, and whether or not to compute a simultaneous confidence band is controlled by the `Simultaneous` argument.

However, as can often be the case when estimands involve many time points or multiple events, it can be difficult to quickly read treatment effects from a table. Instead plotting can make treatment effects and trends visible at a glance.

```
plot(ConcreteOut, NullLine = TRUE, ask = FALSE)
```



Here 95% confidence bands for the cause-specific risk differences across the target times is shown in grey. The `plot` method for `"ConcreteOut"` object invisibly returns a list of `"ggplot"` objects, which can be useful for personalizing these graphs. Currently these plots will not signal whether or not TMLE has converged and whether positivity may be an issue, so users should take care not to ignore the diagnostic output of the `"ConcreteEst"` object prior to obtaining effect estimates using `getOutput()`.

3.5 Summary

This paper introduces the **concrete** R package implementation of continuous-time estimation for absolute risks of right-censored time-to-event outcomes. The package fits into the principled causal-inference workflow laid out by the targeted learning roadmap and allows fully compatible estimation of cause-specific absolute risk estimands for multiple events and at multiple times. The `formatArguments()` function is used to specify desired analyses, `doConcrete()` performs the specified analysis, and `getOutput()` is used to produce formatted output of the target estimands. Cause-specific hazards can be estimated using ensembles of proportional hazards regressions and flexible options are available for estimating treatment propensities. Confidence intervals and confidence bands can be computed for TMLEs, relying on the asymptotic linearity of the TMLEs. We are currently looking into adding support for estimating cause-specific risks using coxnet and HAL-based regressions, as well as supporting stochastic interventions with multinomial or continuous treatment variables.

Bibliography

P. K. Andersen, R. B. Geskus, T. de Witte, and H. Putter. Competing risks in epidemiology: possibilities and pitfalls. *International Journal of Epidemiology*, 41(3):861–870, June 2012. ISSN 0300-5771. doi: 10.1093/ije/dyr213. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3396320/. [p]

P. C. Austin and J. P. Fine. Accounting for competing risks in randomized controlled trials: a review and recommendations for improvement. *Statistics in Medicine*, 36(8):1203–1209, 2017. ISSN 1097-0258. doi: 10.1002/sim.7215. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.7215. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.7215. [p1]

D. Benkeser and N. Hejazi. survtmle: Compute Targeted Minimum Loss-Based Estimates in Right-Censored Survival Settings, Apr. 2019. URL https://CRAN.R-project.org/package=survtmle. [p2]

D. Benkeser and M. Van Der Laan. The Highly Adaptive Lasso Estimator. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 689–696, Oct. 2016. doi: 10.1109/DSAA.2016.93. ISSN: null. [p5]

R. Denz, R. Klaaßen-Mielke, and N. Timmesfeld. A Comparison of Different Methods to Adjust Survival Curves for Confounders, Mar. 2022. URL http://arxiv.org/abs/2203.10002. Number: arXiv:2203.10002 arXiv:2203.10002 [stat]. [p2]

J. P. Fine and R. J. Gray. A Proportional Hazards Model for the Subdistribution of a Competing Risk. *Journal of the American Statistical Association*, 94(446):496–509, June 1999. ISSN 0162-1459. doi: 10.1080/01621459.1999.10474144. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1999.10474144. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1999.10474144. [p1]

T. A. Gerds, T. H. Scheike, and P. K. Andersen. Absolute risk regression for competing risks: interpretation, link functions, and prediction. *Statistics in Medicine*, 31(29):3921–3930, 2012. ISSN 1097-0258. doi: 10.1002/sim.5459. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.5459. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.5459. [p1]

T. A. Gerds, J. S. Ohlendorff, P. Blanche, R. Mortensen, M. Wright, N. Tollenaar, J. Muschelli, U. B. Mogensen, and B. Ozenne. riskRegression: Risk Regression Models and Prediction Scores for Survival Analysis with Competing Risks, Sept. 2022. URL https://CRAN.R-project.org/package=riskRegression. [p2]

P. W. Holland. Statistics and Causal Inference. *Journal of the American Statistical Association*, 81(396):945–960, 1986. ISSN 0162-1459. doi: 10.2307/2289064. URL http://www.jstor.org/stable/2289064. Publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [p1]

E. H. Kennedy. Semiparametric Theory and Empirical Processes in Causal Inference. In H. He, P. Wu, and D.-G. D. Chen, editors, *Statistical Causal Inferences and Their Applications in Public Health Research*, ICSA Book Series in Statistics, pages 141–167. Springer International Publishing, Cham, 2016. ISBN 978-3-319-41259-7. doi: 10.1007/978-3-319-41259-7_8. URL https://doi.org/10.1007/978-3-319-41259-7_8. [p1]

J. P. Klein and P. K. Andersen. Regression Modeling of Competing Risks Data Based on Pseudovalues of the Cumulative Incidence Function. *Biometrics*, 61(1):223–229, 2005. ISSN 1541-0420. doi: 10.1111/j.0006-341X.2005.031209.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.0006-341X.2005.031209.x. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.0006-341X.2005.031209.x. [p1]

M. T. Koller, H. Raatz, E. W. Steyerberg, and M. Wolbers. Competing risks and the clinical community: irrelevance or ignorance? *Statistics in Medicine*, 31(11-12):1089–1097, May 2012. ISSN 1097-0258. doi: 10.1002/sim.4384. [p1]

M. J. v. d. Laan. Statistical Inference for Variable Importance. *The International Journal of Biostatistics*, 2 (1), Feb. 2006. ISSN 1557-4679. doi: 10.2202/1557-4679.1008. URL https://www.degruyter.com/document/doi/10.2202/1557-4679.1008/html?lang=en. Publisher: De Gruyter. [p4]

M. J. v. d. Laan. A Generally Efficient Targeted Minimum Loss Based Estimator based on the Highly Adaptive Lasso. *The International Journal of Biostatistics*, 13(2), Oct. 2017. doi: 10.1515/ijb-2015-0097. [p5]

M. J. v. d. 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. *U.C. Berkeley Division of Biostatistics Working Paper Series*, Nov. 2003. [p1]

M. J. v. d. Laan and J. M. Robins. *Unified Methods for Censored Longitudinal Data and Causality*. Springer Series in Statistics. Springer-Verlag, New York, 2003. ISBN 978-0-387-95556-8. [p1]

M. J. v. d. Laan and S. Rose. *Targeted Learning: Causal Inference for Observational and Experimental Data*. Springer Series in Statistics. Springer-Verlag, New York, 2011. ISBN 978-1-4419-9781-4. doi: 10.1007/978-1-4419-9782-1. [p1, 5]

M. J. v. d. Laan and S. Rose. *Targeted Learning in Data Science: Causal Inference for Complex Longitudinal Studies*. Springer Series in Statistics. Springer International Publishing, 2018. ISBN 978-3-319-65303-7. doi: 10.1007/978-3-319-65304-4. [p1]

M. J. v. d. Laan, E. C. Polley, and A. E. Hubbard. Super Learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), Sept. 2007. ISSN 1544-6115, 2194-6302. doi: 10.2202/1544-6115.1309. [p1, 5]

J. Pearl, M. Glymour, and N. P. Jewell. *Causal Inference in Statistics - A Primer*. Wiley, Chichester, West Sussex, 1st edition edition, Mar. 2016. ISBN 978-1-119-18684-7. [p1]

M. L. Petersen and M. J. van der Laan. Causal models and learning from data: integrating causal modeling and statistical estimation. *Epidemiology (Cambridge, Mass.)*, 25(3):418–426, May 2014. ISSN 1531-5487. doi: 10.1097/EDE.0000000000000078. [p1]

M. L. Petersen, K. E. Porter, S. Gruber, Y. Wang, and M. J. van der Laan. Diagnosing and responding to violations in the positivity assumption. *Statistical Methods in Medical Research*, 21(1):31–54, Feb. 2012. ISSN 0962-2802, 1477-0334. doi: 10.1177/0962280210386207. URL http://journals.sagepub.com/doi/10.1177/0962280210386207. [p6, 10]

R. V. Phillips, M. J. van der Laan, H. Lee, and S. Gruber. Practical considerations for specifying a super learner, Apr. 2022. URL http://arxiv.org/abs/2204.06139. arXiv:2204.06139 [stat]. [p1, 5, 8]

E. Polley, E. LeDell, C. Kennedy, S. Lendle, and M. v. d. Laan. SuperLearner: Super Learner Prediction, May 2021. URL https://CRAN.R-project.org/package=SuperLearner. [p1]

H. C. Rytgaard, T. A. Gerds, and M. J. van der Laan. Continuous-time targeted minimum loss-based estimation of intervention-specific mean outcomes, May 2021a. URL http://arxiv.org/abs/2105.02088. arXiv:2105.02088 [math, stat]. [p4, 5]

H. C. W. Rytgaard and M. J. van der Laan. One-step TMLE for targeting cause-specific absolute risks and survival curves. *arXiv:2107.01537 [stat]*, Sept. 2021. URL http://arxiv.org/abs/2107.01537. arXiv: 2107.01537 version: 2. [p1, 3]

H. C. W. Rytgaard, F. Eriksson, and M. van der Laan. Estimation of Time-Specific Intervention Effects on Continuously Distributed Time-to-Event Outcomes by Targeted Maximum Likelihood Estimation. *arXiv:2106.11009 [stat]*, June 2021b. URL http://arxiv.org/abs/2106.11009. arXiv: 2106.11009. [p1, 8]

T. H. Scheike, M.-J. Zhang, and T. A. Gerds. Predicting cumulative incidence probability by direct binomial regression. *Biometrika*, 95(1):205–220, Mar. 2008. ISSN 0006-3444. doi: 10.1093/biomet/asm096. URL https://doi.org/10.1093/biomet/asm096. [p1]

J. Schwab, S. Lendle, M. Petersen, M. v. d. Laan, and S. Gruber. ltmle: Longitudinal Targeted Maximum Likelihood Estimation, Mar. 2020. URL https://CRAN.R-project.org/package=ltmle. [p2]

O. Sofrygin, M. J. v. d. Laan, and R. Neugebauer. stremr: Streamlined Estimation of Survival for Static, Dynamic and Stochastic Treatment and Monitoring Regimes, Jan. 2017. URL https://CRAN.R-project.org/package=stremr. [p2]

T. Westling, A. Luedtke, P. Gilbert, and M. Carone. Inference for treatment-specific survival curves using machine learning, June 2021. URL http://arxiv.org/abs/2106.06602. Number: arXiv:2106.06602 arXiv:2106.06602 [stat]. [p2]