concrete R Paper

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

Abstract Recently targeted maximum likelihood-based estimation (TMLE) has been used to develop estimators of absolute risk for time-to-event outcomes in continuous-time. The single timepoint continuous-time survival TMLE method is implemented in the **concrete** package for 'R'. **concrete** provides methods to estimate intervention and cause-specific absolute risks as well as contrastive parameters such as risk differences and risk ratios. The package allows the risks of multiple causes to be jointly targeted in the case of competing risks, at multiple time points and in the presence of right-censoring. In this paper we describe and illustrate the usage of the **concrete** package.

1 Introduction

1.1 what concrete is

The R package **concrete** answers causal questions with time-to-event outcomes in continous time. It implements the TMLE method developed in Rytgaard and van der Laan (2021) to estimate time-point specific average treatment effects and returns absolute t-year risks as well as differences and ratios of absolute t-year risks with asymptotic inference based on the efficient influence curve (**?**).

1.2 what is in this manuscript

We write for readers looking for a hands-on introduction to the one-step TMLE method for continuous time survival analysis described in Rytgaard and van der Laan (2021) as well as to readers who want to use our package for their own analyses.

1.3 what concrete does and does not do

The package can be used for targeted estimation of estimands derived from cause-specific absolute risks (e.g. relative risks and risk differences) under static and dynamic binary treatments given at baseline. It deals with baseline covariate confounding, right censoring and competing risks.

The package cannot (yet) analyse multiple treatments, continuous or multinomial treatments, or stochastic interventions. Methods to account for paired or clustered data is not yet supported, nor is support for time-dependent treatments (e.g. drop-in) and time-dependent confounding.

This package is not meant to be used for left trunctation (i.e. delayed entry) or interval censored data,

1.4 how it relates to other peoples work

The **ltmle** (Schwab et al. (2020)), **stremr** (Sofrygin et al. (2017)), and **survtmle** (Benkeser and Hejazi (2019)) TMLE implementations either natively or can be adapted to estimate absolute risks of right-censored survival outcomes; **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 thus requires discretization of continuous-time data. Poorly specified discretization can introduce bias and inflate the variance of estimators and no definitive best practices have yet been established, leading to ad-hoc choices that make unknown trade offs between bias and loss of efficiency. Analyzing continuous-time survival data using a continuous-time method avoids this hurdle entirely; **concrete** will be the first R package implementing TMLE for continuous-time survival.

The *Causal Inference* CRAN Task View shows only **riskregression** (Gerds et al. (2022)) and **DTRreg** (Wallace et al. (2020)) as expressly implementing estimation of survival estimands; both packages do so using g-formula plug-in estimators. The *Survival* CRAN Task View does not show any packages as expressly implementing efficient semi-parametric estimators for survival estimands. Not on CRAN are the packages adjustedCurves (Denz et al. (2022)) (iptw, g-formula, aiptw & others) and CFsurvival (Westling et al. (2021)) (one-step estimator).

2 Concepts

2.1 The Targeted Learning Roadmap

When we analyze survival data and make predictions to guide future actions, we are in essence asking causal questions. The formal causal inference frameworks developed in recent decades allow us to make this process rigorous; a systematic roadmap Petersen and van der Laan (2014) integrates causal modeling with statistical estimation into a cohesive causal inference workflow.

Specifically for our continuous survival case, the targeted learning roadmap consists of three steps: defining a time zero, the event or events of interest, time horizon, specifying the desired intervention

identifying confounders, and right-censoring

At a high level this targeted learning roadmap consists of three steps: defining the causal model and estimand, defining the observed statistical model and establishing the requirements for identifying the causal estimand with a statistical estimand, and estimating the statistical estimand using an estimator with desirable properites such as consistency and efficiency.

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

With time-to-event data, the essential counterfactual outcome is the time until some event or events occur to some subjects if they were intervened upon in some way. Let A represent this intervention variable, which could take on values that are binary (such as with a 2-armed trial), multinomial, or even continuous. Interventions on this variable can take different forms; the simplest is setting the variable to a constant value a in the range of A, a so-called a static treatment. Alternatively the intervention could be a function d, maybe specifying a dynamic treatment with a deterministic function incorporating baseline covariates W, $d(W)$, or maybe specifying a stochastic treatment with a probabilistic function that could even depend on the treatment variable, $d(A,W)$. Whatever the desired intervention may be, if we let d represent this intervention of interest, then for J events of interest we can define the counterfactual time-to-event variables $T_j^d,\ j = 1,...,J$, representing the time until event j happens if subjects were intervened upon following rule d. With this we can write the generic form of time-to-event counterfactual data with J target events, baseline covariates L and intervention rule d determining the value of intervention variable A.

$$X = \left(T_j^d, W\ :\ j \in 1,\ldots,J\right)$$

For a concrete example, take the simple case of the 2-armed PBC trial where researchers wanted to know the effect of d-Penicillamine compared to placebo on the time until subjects either die or receive a liver transplant. The ideal, albeit physically impossible study would have been to:

- Assign a group of subjects to treatment with d-Penicillamine (A = 1); then follow them for some length of time without exception (e.g. no drop-outs and no loss-to-follow-up), and record when subjects die or and when they receive liver transplants.
- Rewind time and assign that same group to placebo (A = 0), follow them for the same length of time without exception, and record when subjects die and when they receive liver transplants.

Counterfactuals allow us to express this data in the following mathematical notation:

$$X = (T_1^1, T_1^0, T_2^1, T_2^0, W : T_j^a \leq t_{max})$$

where W is some collection of baseline covariates, t_{max} is the desired follow-up time, T_1^1 is the time until death given d-Penicillamine, T_1^1 is the time until death given placebo, T_2^1 is the time until liver transplant given placebo, and T_2^0 is the time until liver transplant given d-Penicillamine.

Counterfactual notation also allows us to mathematically define causal estimands such as causal risk differences and causal risk ratios. For instance in the PBC example, if we wish to know the risk of death by some time t in a hypothetical world where everyone is treated and nobody received life-saving liver transplants, then our target causal quantity is

$$\Psi_t^F = P(T_1^1 \leq t) \tag{1}$$

However, if we instead wish to know the risk of death by some time t in a hypothetical world where everyone is treated but liver transplants occur normally, then the question is really about what is the risk of death before a life-saving liver transplant. To formalize this quantity, for some intervention d and some events $j = 1,\ldots,J$ let

$$\widetilde{T}^d = \min_j T_j^d \quad \text{and} \quad \Delta^a = \operatorname*{argmin}_j T_j^d$$

where $\widetilde{T}^d$ is the counterfactual time where subjects treated with intervention d would experience the earlier of either death or liver transplant, and where Δ^d indicates whether death happened first or liver-transplant

happened first. For the pbc dataset, where the desired interventions were to treat everyone with d-Penicillamine or to give everyone placebo, the counterfactual data can be written

$$X = (\widetilde{T}^1, \Delta^1, \widetilde{T}^0, \Delta^0, W : T_j^a \leq t_{max})$$

where $(\widetilde{T}^1, \Delta^1)$ marks in the d-Penicillamine hypothetical the time when subjects would experience the earlier of either death or liver transplant $\widetilde{T}^1$ and whether it was death or liver transplant that happened first Δ^1, while $(\widetilde{T}^0, \Delta^0)$ describe the respective events in the placebo hypothetical. The average risk of death by time t if everyone were treated, while allowing liver transplants to occur is then

$$\Psi_t^F = P(\widetilde{T}^1 \leq t, \Delta = 1) \tag{2}$$

If the target quantities in (1) and (2) look familiar, that may be because they correspond to common survival and competing risk estimands repectively. Here lies a great benefit of following a structured causal roadmap when devising a statistical analyses: the confusion around when to use competing risks versus right-censored analyses can be cleared up by careful definition of the desired hypothetical: events that would be prevented are appropriately analyzed as censoring and events that would be allowed to compete are correctly addressed as competing risks.

2.3 Observed Data

In time-to-event data, subjects are followed over time until some event occurs, a process that may be subject to censoring. Let O denote the observations on one such subject where O is drawn from a distribution P_0 and let C represent subjects' censoring times. The observed data O then might include a vector of baseline covariates which we denote as W as well as a treatment variable A. The observed time to first event (censoring or otherwise) we can write as $\widetilde{T} = \min(C,\ T_j : j = 1, \ldots, J)$, where C is the censoring time and T_j are the event times to each of the events j. To identify which event is observed we define $\Delta = (\underset{j}{\operatorname{argmin}}\, T_j) \times \mathbf{1}(\min_j T_j \leq C)$, with $\Delta = 0$ being that censoring occurred. The observed survival data, potentially with right censoring and competing events, can then be represented as

$$O = (\widetilde{T},\ \Delta,\ A,\ W)$$

In the pbc example, we might be interested in comparing the risk of dying in the absence of liver transplants by some time t if everyone were given the intervention while $P(T_1^1 \leq t)$ versus the analogous risk if everyone were given the placebo $P(T_1^0 \leq t)$. Typically this comparison might be a risk difference $P(T_1^1 \leq t) - P(T_1^0 \leq t)$, or a risk ratio $P(T_1^1 \leq t)/P(T_1^0 \leq t)$.

However when subjects are susceptible to multiple mutually exclusive events of interest, solely focusing on the effect of a treatment on one event can be misleading. For instance in the pbc trial, an intervention that decreases the risk of death with a large increase in the "risk" of liver transplants likely implies a different mechanism of action compared to an intervention that decreases the risk of death while not substantially increasing the "risk" of liver transplant. The ability to distinguish between these mechanisms of effect can be important, and so in competing risks settings we should track the effect of treatment on the set of possible events, e.g. $\left(P(\widetilde{T}^1 \leq t, \Delta = 1) - P(\widetilde{T}^0 \leq t, \Delta = 1),\ P(\widetilde{T}^1 \leq t, \Delta = 2) - P(\widetilde{T}^0 \leq t, \Delta = 2)\right)$.

Given a set of assumptions, namely 1) consistency, 2) positivity for treatments and remaining uncensored, 3) no unmeasured confounding, and 4) coarsening at random on the censoring process, causal survival and risk estimands are identified by statistical estimand counterparts which are purely functions of the observed data. These identification assumptions, formally stated in detail in Appendix 6.1, make explicit the necessary untestable assumptions about the causal model which has produced the observed data for statistical quantities to have causal interpretations. If on the other hand, one does not believe the identification assumptions hold, then these statistical estimands still have an interpretation as standardized risks isolating the importance of the "treatment" variable (Laan (2006)).

2.4 Estimation

concrete implements the one-step TMLE for right-censored survival and competing risks in continuous time described by Rytgaard et al. (2021) and Rytgaard and van der Laan (2021). TMLE (Laan (2006), Laan and Rose (2011)) is a general methodology for constructing semi-parametric efficient substitution estimators consisting of two broad steps: first an initial estimation of nuisance parameters utilizing flexible machine learning and second a targeted update of the initial estimators to solve the efficient influence function of the target statistical estimand. TMLE estimators given certain conditions are efficient regular asymptotically linear estimators and thus are consistent with asymptotically minimal variance and, being substitution estimators, respect global

constraints on target parameters such as monotonicity and being bounded between $[0, 1]$ for survival curves.

For TMLE to have its desirable properties, the initial estimation stage must converge adequately quickly; to this end, we employ a flexible machine learning ensemble with oracle guarantees (Laan et al. (2007), Polley et al. (2021), Laan and Dudoit (2003), Vaart et al. (2006)), particularly with a candidate library incorporating the highly adaptive lasso (HAL) as it converges at the required rate (Laan (2017), Benkeser and Van Der Laan (2016), Rytgaard et al. (2021)).

The subsequent update or targeting step leans on a result from semi-parametric efficiency theory (Bickel et al. (1998)), which states that a regular, asymptotically linear estimator for a statistical target parameter in a semiparametric model is asymptotically efficient if its influence function is equal to the efficient influence curve (EIC). The efficient influence function for treatment and cause-specific absolute risks, Eq. (3), have been derived (Rytgaard et al. (2021), Rytgaard and van der Laan (2021)) and by updating the estimated nuisance parameters to solve the efficient influence curve, we construct a consistent estimator with a normal limiting distribution and the smallest attainable variance.

For every desired treatment regime π^*, every target time τ, and every target event j, the efficient influence functions for the treatment and cause-specific absolute risk is:

$$D^*_{\pi^*,j,\tau}(\hat{\lambda},\hat{\pi},\hat{S}_c)(O) = \sum_{l=1}^{J}\sum_{k=1}^{K} h_{\pi^*,j,l,\tau,s}(\hat{\lambda},\hat{\pi},\hat{S}_c)(O)\,\left(\mathbf{1}(\Delta = j, \tilde{T} = s_k) - \mathbf{1}(\tilde{T} \geq s_K)\,\hat{\lambda}_l(s_k \mid A, W)\right) + \sum_{a\,\in\,\mathcal{A}} F_j(\tau \mid A = a, W)\,\pi^*(a \mid W) - \Psi_{\pi^*,j,\tau}(P_0) \tag{3}$$

where

$$h_{\pi^*,j,l,\tau,s}(\hat{\lambda},\hat{\pi},\hat{S}_c)(O) = \frac{\pi^*(A \mid W)\,\mathbf{1}(s \leq \tau)}{\hat{\pi}^{SL}(A \mid W)\;\hat{S}_c(s\text{-} \mid A, W)}\left(\mathbf{1}(\Delta = j) - \frac{\hat{F}_j(\tau \mid A, W) - \hat{F}_j(s \mid A, W)}{\hat{S}(s \mid A, W)}\right)$$

The clever covariate is a function of the desired intervention density which is user specified, the observed intervention densities which are not changed by tmle targeting, and the outcome-related densities which are updated by targeting.

TMLE one-step update Let D^* be the vector of efficient influence functions

$$D^*(\lambda,\pi,S_c)(O) = \left(D^*_{\pi^*,j,\tau}(\lambda,\pi,S_c)(O) : \pi^* \in \mathcal{A}, j \in \mathcal{J}, \tau \in \mathcal{T}\right)$$

and let $h_{j,s}$ be the vector of clever covariates

$$h_{j,s}(\lambda,\pi,S_c)(O) = \left(h_{\pi^*,j,l,\tau,s}(\lambda,\pi,S_c)(O) : \pi^* \in \mathcal{A}, j \in \mathcal{J}, \tau \in \mathcal{T}\right)$$

The one-step TMLE involves updating the cause-specific hazards along the universal least favorable submodel. This is implemented by updating the hazards in small steps along the sequence of locally-least favorable submodels in the following manner:

$$\hat{\lambda}_{j,\epsilon_m}(t) = \hat{\lambda}^{SL}_j(t)\,\exp\left(\sum_{i=1}^{m}\frac{\left\langle \mathbb{P}_n D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O),\; h_{j,s}(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)\right\rangle_\Sigma}{||D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)||_\Sigma}\,\epsilon_i\right)$$

where

$$\langle x, y\rangle_\Sigma = x^\top \Sigma^{-1} y \quad , \quad ||x||_\Sigma = \sqrt{x^\top \Sigma^{-1} x}$$

The default value of ϵ in the software is 0.1, and the algorithm stops at ϵ_i when

$$\mathbb{P}_n D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O) \leq \frac{\sqrt{\mathbb{P}_n\; D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)^2}}{\sqrt{n}\,\log(n)}$$

3 Using concrete

concrete was written for causal analyses of time-to-event data which is reflected in its structure and variable naming, though of course **concrete** can also be used for non-causal estimation problems. There are 3 main user-facing functions in **concrete**: `formatArguments()`, `doConcrete`, and `getOutput`. Reflecting our vision of good statistical practice, the majority of user effort is directed into defining the desired analysis by

specifying arguments into `formatArguments()`. The output of `formatArguments()` is a `{"ConcreteArgs"}` object which is passed into `doConcrete()` to perform the specified continuous-time one-step survival TMLE. The output of `doConcrete()` is a `"ConcreteEst"` object which will be described in further detail in Section 3.2. This `"ConcreteEst"` object can be passed into getOutput to print, summarize, and/or plot the desired cause-specific absolute risk derived estimand.

3.1 formatArguments()

Broadly speaking, arguments into `formatArguments` fall into 3 broad categories: specifying the observed data structure, specifying the target estimand, and specifying the estimation algorithm. `formatArguments()` checks these inputs for compatibility and returns errors, warnings, and messages as necessary. The output of `formatArguments` is an object of class `"ConcreteArgs"`, which can then be modified by the user and returned through `formatArguments` to be re-checked. This process can be repeated as many times as necessary until the full estimation problem is adequately specified. formatArguments() is how the user specifies the estimation problem which consists of the major features of the observed data structure, the target quantities, and estimation choices.

Data

As a refresher, the general form of some observed right-censored survival data, potentially with competing events is

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

where T is the observed time to first event (censoring or otherwise), Δ marks which event occured (with $\Delta = 0$ indicating right-censoring), A is the intervention variable, and W is a collection of baseline covariates.

In the PBC dataset example, $\widetilde{T}$ is the column `"time"`, Δ is the column `"status"`, A is the column `"trt"`, and W consists of all the other columns except the `"id"` column, which is meant to allow for future functionality analyzing data with clustering or longitudinal confounding.

```
library(concrete)
library(data.table)
set.seed(0)
obs <- as.data.table(survival::pbc)
obs <- obs[,  c("time", "status", "trt", "id", "age", "albumin", "sex", "stage")]
obs <- obs[!is.na(trt), ]
obs[, stage := as.factor(stage)]
head(obs, 5)
```

Error: object 'obs' not found Error in head(obs, 5) : object 'obs' not found

This data is passed into **concrete** through the arguments `DataTable=`, `EventTime=`, `EventType=`, and `Treatment=`. The `formatArguments(DataTable= )` argument takes as in put the full observed dataset as an object of class `"data.table"` or `"data.frame"`. It must contain columns specifying 1) the observed event or censoring times, 2) the event type (where a value of 0 indicates censoring), and 3) the treatment. The input dataset cannot have missing (e.g. NA, NaN) or infinite values, and any necessary covariate imputation should be done by the user before using **concrete** (we further advise augmenting imputed data with additional indicator columns to mark where covariate imputation was done) while missingess in treatment or in event times (other than right-censoring) is outside the scope of this package.

The event/censoring times must be positive numbers and the name of that column is specified by the `formatArguments(EventTime= )` argument. The event/censoring type must be non-negative integers (with 0 indicating censoring) and that column name is specified by the `formatArguments(EventType= )` argument. The treatment must currently be binary numeric (0 or 1) and that column name is specified by the `formatArguments(Treatment= )` argument. Additionally the dataset may include a column containing uniquely identifying subject ids, the name of which should be passed into `formatArguments(ID = )`, and any number of additional columns containing baseline covariates.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, RenameCovs = TRUE)
```

By default columns containing baseline covariates will be renamed in a standardized way and any categorical covariates will be 1-hot encoded (Cox model formulas for hazard estimation will automatically be renamed as necessary, Section 3.1.3.4). The renamed and formatted data table can be accessed through the

`.[["Data"]]` element of the `"ConcreteArgs"` object returned by `formatArguments()`. This behaviour can be turned off by setting `RenameCovs=FALSE`.

```
head(ConcreteArgs$Data)
```

Error in head(ConcreteArgs$Data) : object 'ConcreteArgs' not found

The original columns (and categorical values when applicable) can be linked to the new columns through the returned `.[["Data"]]` element's `"CovNames"` attribute: `"ColName"` lists the columns in the renamed data table, `"CovName"` lists the names of the original columns, and `"CovVal"` lists the values of the original columns for the case when categorical values are spread over several new columns.

```
attr(ConcreteArgs$Data, "CovNames")
```

Error in publish(x, ..., org = TRUE) : object 'ConcreteArgs' not found

The `.[["Data"]]` element is pinned with the `EventTime=` , `EventType=` , `Treatment=` , `ID=` , and `RenameCovs=` arguments, with respectively named attributes (e.g. `attr(*,"EventTime")`).

Target Estimand

concrete implements a continuous time one-step TMLE jointly targeting the cause-specific absolute risks at certain target times under some hypothetical treatments. This idea of what might have happened if something contrary to fact had been done can be formalized using the language of counterfactuals. Given certain identification assumptions, these counterfactual estimands can be identified from functions of the observed data.

Treatment Regime With time-to-event data, the essential counterfactual outcome is the time until some event or events occur to some subjects if they were intervened upon in some way. Let A represent this intervention variable, which could be binary as with 2-armed trials, multinomial, or even continuous. Interventions on this variable can take on different forms; the simplest is setting the variable to a constant value a in the range of A, a so-called static regime. Alternatively the intervention could be a function d, perhaps specifying a dynamic treatment using a deterministic function of baseline covariates W, $d(W)$, or even stochastic treatments using a probabilistic function that could depend on the treatment variable as well as covariates, $d(A,W)$. Let d be some desired intervention. Then for J events of interest we can define the counterfactual time-to-event variables T_j^d , $j = 1, ..., J$, representing the time until event j happens if subjects were intervened upon following rule d. With this we can write the generic form of time-to-event counterfactual data with J target events, baseline covariates L and intervention rule d determining the value of intervention variable A.

$$X = \left(T_j^d, W \,:\, j \in 1, \ldots, J\right)$$

For a concrete example, take the simple case of the 2-armed PBC trial where researchers wanted to know the effect of d-Penicillamine compared to placebo on the time until subjects either die or receive a liver transplant. The ideal, albeit physically impossible study would have been to:

- Assign a group of patients to treatment with d-Penicillamine (A = 1); then observe them for some length of time without exception (e.g. no drop-outs and no loss-to-follow-up), and record when patients die and when they receive liver transplants.
- Rewind time and assign that same group to placebo (A = 0), observe them for the same length of time without exception, and record when patients die and when they receive liver transplants.

Counterfactuals allow us to express this data in the following mathematical notation:

$$X = (T_1^1, T_1^0, T_2^1, T_2^0, W : T_j^a \leq t_{max}, a \in \{0,1\})$$

where W is some collection of baseline covariates, t_{max} is the desired follow-up time, T_1^1 is the time until death given d-Penicillamine, T_1^1 is the time until death given placebo, T_2^1 is the time until liver transplant given placebo, and T_2^0 is the time until liver transplant given d-Penicillamine.

Desired interventions are passed into **concrete** with the `formatArguments(Intervention= )` argument. Static interventions can be specified with the desired global value, e.g. 0, 1, or 0:1. Dynamic and stochastic treatments instead are specified by a pair of functions: an 'intervention' function which outputs desired treatment **assignments** and a 'g.star' function which outputs desired treatment **probabilities**. Functionality for specifying 'g.star' functions based on estimated propensity scores for stochastic interventions is forthcoming.

Though the static regimes for a 2-armed trial can be simply specified as mentioned above, the functions corresponding to assigning everyone the treatment (i.e. $A = 1$) and assigning everyone to a control (i.e.

$A = 0$) can be created using `makeITT()`. The result of `makeITT()` is a list of two desired counterfactual interventions: "A=1" details an the intervention where everyone is assigned treatment, and "A=0" details an intervention where everyone is assigned control. This is meant to be a template for users to explore more complex treatment rules.

```
ITT <- makeITT()
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1)
```

Target Events An equally crucial component of defining a target estimand is specifying the target events. Take the counterfactual formulation of the pbc dataset

$$X = (T_1^1, T_1^0, T_2^1, T_2^0, W : T_j^a \leq t_{max})$$

If we wish to know the risk of death by some time t in a hypothetical world where everyone is treated and nobody received life-saving liver transplants, then our target causal quantity is

$$\Psi_t^F = P(T_1^1 \leq t)$$

However, if we instead wish to know the risk of death by some time t in a hypothetical world where everyone is treated but liver transplants occur normally, then the question is really about what is the risk of death before a life-saving liver transplant. To formalize this quantity, for some intervention d and some events $j = 1, \ldots, J$ let

$$\widetilde{T}^d = \min_{T_j^d} T_j^d \quad \text{and} \quad \Delta^a = \underset{j}{\operatorname{argmin}}\, T_j^d$$

where T^d is the counterfactual time where subjects treated with intervention d would experience the earlier of either death or liver transplant, and where Δ^d indicates whether death happened first or liver-transplant happened first. For the pbc dataset, where the desired interventions were to treat everyone with d-Penicillamine or to give everyone placebo, the counterfactual data can be written

$$X = (\widetilde{T}^1, \Delta^1, \widetilde{T}^0, \Delta^0, W : T_j^a \leq t_{max})$$

where $(\widetilde{T}^1, \Delta^1)$ marks in the d-Penicillamine hypothetical the time when subjects would experience the earlier of either death or liver transplant $\widetilde{T}^1$ and whether it was death or liver transplant that happened first Δ^1, while $(\widetilde{T}^0, \Delta^0)$ describe the respective events in the placebo hypothetical. The average risk of death by time t if everyone were treated, while allowing liver transplants to occur is then

$$\Psi_t^F = P(\widetilde{T}^1 \leq t, \Delta = 1)$$

If the target quantities in (1) and (2) look familiar, that may be because they correspond to the common potentially right-censored survival and competing risk estimands repectively. Here lies a great benefit of following a structured causal roadmap when devising analyses: the confusion around when competing risks versus right-censored analyses are appropriate can be avoided by asking what is the desired hypothetical: events that would be prevented are censoring and events that would be allowed to compete are competing risks.

The `formatArguments(TargetEvent = )` argument is used to specify which events are of interest, events which must be encoded as non-negative integers. In the pbc dataset for example, there are 3 possible event values, encoded by the `status` column : 0 for censored, 1 for transplant, and 2 for death. In **concrete** 0 is reserved to indicate censoring, while events of interest can be encoded as any positive integer. Setting `formatArguments(TargetEvent = 1:2)` for the pbc dataset specifies a joint targeting of the risk of transplant and death. By default **concrete** by targets all observed non-censoring events, so leaving the `formatArguments(TargetEvent = NULL)` would achieve the same result.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2)
```

Target Time The last component to specifying our risk estimand whether in the survival (1) or competing risks case (2) is to choose t, the times at we wish to assess risk. Here one might be a single time, or multiple

times.

The ‘TargetTime‘ argument specifies the time(s) at which estimates of the event-specific absolute risks and/or event-free survival are desired. Target times should be restricted to the time range in which failure events are observed, since estimating event risks after the point in time where all individuals are censored entails unsupported extrapolation. To discourage this behaviour, 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.

```
BadTime <- unique(obs[status > 0, max(time)]) + 1
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = BadTime)
```

Error in data.table::data.table(TimeVal = TimeVal, TypeVal = TypeVal) : object ’obs’ not found

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.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = (3:7)*500)
```

Estimator Specification

The arguments involved in estimation are the cross-validation setup `CVArg`, the estimation models `Model`, the software backends `PropScoreBackend` and `HazEstBackend`, `MaxUpdateIter`, `OneStepEps`, and `MinNuisance`. It should be noted here that `Model` is used here to conform with common usage in statistical analysis R packages, rather than to refer to a statistical or causal model as we have in the previous sections.

Cross-Validation **concrete** uses **origami** to specify cross-validation folds, specifically the function `origami::make_folds()`. If no input is provided to the `formatArguments(CVArg= )` argument, concrete will use origami to implement a simple 10-fold cross-validation scheme. For how to specify more sophisticated cross-validation schemes, see this brief vignette or detailed chapter on using origami from the tlverse handbook

```
library(origami)
# If the CVArg argument is NULL, concrete uses a simple 10-fold CV as the default
    specification, i.e.
CVArgs <- list(n = ncol(obs), fold_fun = folds_vfold, cluster_ids = NULL, strata_
    ids = NULL)

# For different number of folds, simply add the `V = ` argument, e.g.
CVArgs <- list(n = ncol(obs), V = 5L, fold_fun = folds_vfold, cluster_ids = NULL,
    strata_ids = NULL)

ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = (3:7)*500,
                CVArg = CVArgs)
```

Estimating Nuisance Parameters TMLE requires initial estimation of components of the observed data distribution which we term nuisance parameters; for continuous-time TMLE of survival and absolute risks, we require estimates of the treatment propensity score and conditional hazards for each event and censoring type. The ‘formatArguments(Model =)‘ argument is how **concrete** accepts estimator specifications for estimating these nuisance parameters. Inputs into the ‘Model‘ argument must be named lists with one entry for the ’Treatment’ variable, and for each of the event type (and censoring). The list element corresponding to the ’Treatment’ variable must be named as the variable name, and the list elements corresponding to each event type must be named as the numeric value of the event type (with "0" being reserved for censoring). If no input is provided for the ‘Model‘ argument but appropriate arguments specifying the data and target estimands are supplied, then ‘formatArguments‘ will return a correctly formatted list containing default estimator specifications for each nuisance parameter, which can be then augmented.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = (3:7)*500,
                CVArg = NULL, Model = NULL)
str(ConcreteArgs[["Model"]], give.attr = FALSE)
```

Estimating Treatment Propensity Propensity scores for treatment assignment are estimated using the Superlearner stacked ensemble machine learning algorithm, using either the 'SuperLearner' package (PropScoreBackend = "Superlearner") or the 'sl3' package (PropScoreBackend = "sl3"). If using formatArguments(PropScoreBackend = "SuperLearner"), **concrete** passes the 'Model' specification for the treatment variable into SuperLearner(SL.library =). In the next section we illustrate how to specify treatment models using the "SuperLearner" backend, but detailed instructions for how to specify models using **SuperLearner** can be found in the package vignette.

Alternatively, if 'PropScoreBackend' is set to "sl3" then **concrete** uses the 'sl3::Lrnr$_{sl}$' object to estimate the treatment propenity score . Below we show a simple example of using 'sl3' to estimate propensity scores for **concrete**, but Chapter 6 in the tlverse handbook provides an in depth explanation for how to specify a Super learner using 'sl3'.

The default model specification for estimating treatment propensity is with SuperLearner using a library consisting of "xgboost" and "glmnet".

Estimating Event and Censoring Hazards 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 pseuo-likelihood loss. Examples of how to specify models for estimating conditional hazards with **concrete** are shown below. Support for estimation of hazards using Poisson-HAL or 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.

```
ConcreteArgs[["Model"]][["0"]] <- list("model1" = Surv(time, status == 0) ~ trt +
    age:sex,
                       "model2" = Surv(time, status == 0) ~ .)
ConcreteArgs[["Model"]][["1"]] <- list(Surv(time, status == 1) ~ .,
                       ~ trt + age)
ConcreteArgs[["Model"]][["2"]] <- "."

ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = (3:7)*500,
                CVArg = NULL, Model = ConcreteArgs[["Model"]],
                PropScoreBackend = "SuperLearner", HazEstBackend = "coxph")
```

As mentioned in Section 2.3, Cox models are renamed to reflect renamed columns; the revised model names can be checked in the 'Model' element of the 'ConcreteArgs' object returned by `formatArguments()`.

```
str(ConcreteArgs[["Model"]], give.attr = FALSE)
```

One-step TMLE Specification The one-step TMLE implemented in **concrete** can jointly target survival and multiple cause-specific risks at multiple time points up to full curves, producing monotonic curves that sum appropriately to 1 while allowing for simultaneous inference. It does so by updating the cause-specific hazards along the universal least favorable submodel described in Rytgaard and van der Laan (2021), implemented by updating the hazards in small steps along the sequence of locally-least favorable submodels in the following manner:

$$\hat{\lambda}_{j,\epsilon_m}(t) = \hat{\lambda}_j^{SL}(t) \exp\left(\sum_{i=1}^{m} \frac{\left\langle \mathbb{P}_n D^*(\hat{\lambda}_{\epsilon_i}, \hat{\pi}, \hat{S}_c)(O),\ h_{j,s}(\hat{\lambda}_{\epsilon_i}, \hat{\pi}, \hat{S}_c)(O) \right\rangle_\Sigma}{||D^*(\hat{\lambda}_{\epsilon_i}, \hat{\pi}, \hat{S}_c)(O)||_\Sigma} \epsilon_i\right) \tag{4}$$

where $\langle x, y \rangle_\Sigma = x^\top \Sigma^{-1} y \quad , \quad ||x||_\Sigma = \sqrt{x^\top \Sigma^{-1} x}$

D^* is the vector of efficient influence functions

$$D^*(\lambda, \pi, S_c)(O) = \left(D^*_{\pi^*,j,\tau}(\lambda, \pi, S_c)(O) : \pi^* \in \mathcal{A}, j \in \mathcal{J}, \tau \in \mathcal{T})\right)$$

and $h_{j,s}$ is the vector of clever covariates

$$h_{j,s}(\lambda, \pi, S_c)(O) = \left(h_{\pi^*,j,l,\tau,s}(\lambda, \pi, S_c)(O) : \pi^* \in \mathcal{A}, j \in \mathcal{J}, \tau \in \mathcal{T})\right)$$

stopping at the ϵ_i when

$$\mathbb{P}_n D^*(\hat{\lambda}_{\epsilon_i}, \hat{\pi}, \hat{S}_c)(O) \leq \frac{\sqrt{\mathbb{P}_n\ D^*(\hat{\lambda}_{\epsilon_i}, \hat{\pi}, \hat{S}_c)(O)^2}}{\sqrt{n}\ \log(n)}$$

The value of ϵ is provided by the user as input into the argument `OneStepEps`; its default value is 0.1 and user-provided values must be between 0 and 1. The value of `OneStepEps` is meant to be heuristically small as Equation (4) approximates an integral; therefore it is shrunk by a factor of 2 whenever an update step would increase the norm of the efficient influence function.

To ensure that the update step does not continue infinitely, the user specifies the maximum number of small update recursions through the argument `MaxUpdateIter`. This argument takes positive integers and is set to a default of 100.

In keeping with other TMLE packages, an argument `MinNuisance` is available 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. This value should heuristically be small, with default 0.05, but a better solution would be to ask questions about treatment regimes that are better supported in the data. `doConcrete` returns messages about near-positivity truncation and vectors of the untruncated nuisance denominator.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = (3:7)*500,
                CVArg = NULL, Model = ConcreteArgs[["Model"]],
                PropScoreBackend = "SuperLearner", HazEstBackend = "coxph",
                MaxUpdateIter = 100, OneStepEps = 1, MinNuisance = 0.05)
```

Miscellaneous Arguments Additional miscellaneous, but perhaps useful arguments are provided. The argument `Verbose` determines whether or not a TMLE convergence vector will be returned during the one-step TMLE process, `GComp` determines whether or not a simple plug-in g-formula estimator based on the SuperLearner fit will be returned, and `ReturnModels` determines whether or not fitted models will be saved and returned in the final output.

ConcreteArgs object `formatArguments()` returns a list object of class `"ConcreteArgs"`. This object includes a `.[["Data"]]` element as mentioned before (the reformatted input data table tagged with variable names) as well as a `"Regime"` element, which is a list of treatment regimes, each tagged with its accompanying `"g.star"` formula. The other elements are checked versions of the various input arguments. More details are available in the documentation of the `formatArguments()` function.

Importantly, `"ConcreteArgs"` objects can be passed into `formatArguments()` in lieu of supplying each of the arguments directly. This means that the output of `formatArguments()` can be saved, altered, and passed back through `formatArguments()` to be re-checked.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = (3:7)*500,
                CVArg = NULL, Model = ConcreteArgs[["Model"]],
                PropScoreBackend = "SuperLearner", HazEstBackend = "coxph",
                MaxUpdateIter = 100, OneStepEps = 1, MinNuisance = 0.05)

ConcreteArgs <- formatArguments(ConcreteArgs)
```

3.2 doConcrete

Once `formatArguments()` runs without errors, the resulting object of class "`ConcreteArgs`" should be a suitable input into the function `doConcrete()` which should return the desired targeted estimates without any further user interaction. The resulting object contains TMLE point estimates and influence curves for the cause-specific absolute risks for each targeted event at each targeted time. If `GComp=TRUE`, then the object will also contain the result of using the Superlearner predictions as a plug-in g-formula estimate of the targeted risks.

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

For an in depth and rigorous description of the one-step TMLE, see Rytgaard and van der Laan (2021). Here we attempt to give an overview of the major stages of the one-step TMLE process.

Given identification assumptions, the distribution for data generated hypothetically following a desired treatment regime involving $A \sim \pi^*(A \mid W)$ and the prevention of the censoring process can be identified as

$$p^{\pi^*}(O) = p(W)\,\pi^*(A \mid W) \prod_{j=1}^{J} S(\widetilde{T}\text{-} \mid A, W)\lambda_j(\widetilde{T} \mid A, W)^{\mathbf{1}(\Delta=j)}$$

For a target parameter of the cause $j \in J$ absolute risk at time $\tau \in \mathcal{T} \subseteq [0, t_{max}]$ under this treatment regime π^*, the corresponding efficient influence function is

$$D^*_{\pi^*,j,\tau}(P)(O) = \sum_{j=1}^{J} \int_0^\tau \left[h_{\pi^*,j,l,\tau,s}(P)(O) \left(N_j(ds) - \mathbf{1}(\widetilde{T} \geq s)\, \lambda_l(s \mid A, W) \right) \right] ds$$
$$+ \sum_{a=0,1} F_j(t \mid A = a, W)\, \pi^*(a \mid X) - \Psi_{\pi^*,j,\tau}(P_0)$$

with the clever covariate

$$h_{\pi^*,j,l,\tau,s}(P)(O) = \frac{\pi^*(A \mid W)\,\mathbf{1}(s \leq \tau)}{\pi(A \mid W) S_c(s\text{-} \mid A, W)} \left(\mathbf{1}(\delta = j) - \frac{F_j(\tau \mid A, W) - F_j(s \mid A, W)}{S(s \mid A, W)} \right)$$

As the efficient influence function and clever covariates depend on the treatment distribution π, the censoring survival function S_c, and the event cause-specific hazards $\lambda = (\lambda_l : j = 1, ..., J)$, we will in subsequent sections use the following alternative notation for clarity when appropriate:

$$D^*_{\pi^*,j,\tau}(\lambda, \pi, S_c)(O) = D^*_{\pi^*,j,\tau}(P)(O)$$
$$h_{\pi^*,j,l,\tau,s}(\lambda, \pi, S_c)(O) = h_{\pi^*,j,l,\tau,s}(P)(O)$$

Therefore, to efficiently estimate survival-curve derived estimands such as the cause-specific absolute risks, the components of the data distribution that must be estimated are $\pi(A \mid W)$, $S_c(t \mid A, W)$, $\lambda_j(t \mid A, W)$, $F_j(t \mid A, W)$, and $S(t \mid A, W)$

Cross-Validation Specification

Let $Q_n = \{O_i\}_{i=1}^n$ be an observed sample of n i.i.d observations of $O \sim P_0$. For V-fold cross validation, let $B_n = \{1, ..., V\}^n$ be a random vector that assigns the n observations into V validation folds. For each $v \in \{1, ..., V\}$ we then define training set $Q_v^{\mathcal{T}} = \{O_i : B_n(i) = v\}$ with the corresponding validation set $Q_v^{\mathcal{V}} = \{O_i : B_n(i) \neq v\}$.

Propensity Score Estimation

For the true conditional distribution of A given W, $\pi_0(\cdot \mid W)$, and $\hat{\pi} : Q_n \to \hat{\pi}(Q_n)$, let L_π be a loss function such that the risk $\mathbb{E}_0\left[L_\pi(\hat{\pi}, O)\right]$ is minimized when $\hat{\pi} = \pi_0$. For instance, with a binary A, we may specify the negative log loss $L_\pi(\hat{\pi}, O) = \text{-}\log\left(\hat{\pi}(1 \mid W)^A\, \hat{\pi}(0 \mid W)\right)^{1-A}$. We can then define the discrete superlearner selector which chooses from a set of candidate models $\mathcal{M}_\approx$ the candidate propensity score model that has minimal cross validated risk

$$\hat{\pi}^{SL} = \underset{\hat{\pi} \in \mathcal{M}_\pi}{\operatorname{argmin}} \sum_{v=1}^{V} P_{Q_v^{\mathcal{V}}}\, L_\pi(\hat{\pi}(Q_v^{\mathcal{T}}), Q_v^{\mathcal{V}})$$

This discrete superlearner model $\hat{\pi}^{SL}$ is then fitted on the full observed data Q_n and used to estimate $\pi_0(A \mid W)$

Hazard Estimation

Let $\lambda_{0,\delta}$ be the true censoring and cause-specific hazards when $\delta = 0$ and $\delta = 1, \ldots, J$ respectively. Let $\mathcal{M}_\delta$ for $\delta = 0, \ldots, J$ be the sets of candidate models, $\{\hat{\lambda}_\delta : Q_n \to \hat{\lambda}_\delta(Q_n)\}$, for the censoring and cause-specific hazards and let L_δ be loss functions such that the risks $\mathbb{E}_0\left[L_\delta(\hat{\lambda}_\delta, O)\right]$ are minimized when $\hat{\lambda}_\delta = \lambda_{0,\delta}$, for instance log likelihood loss. We can then define the discrete superlearner selectors for each δ which choose from the set of candidate models $\mathcal{M}_\circ$ the candidate propensity score model that has minimal cross validated risk

$$\hat{\lambda}_\delta^{SL} = \underset{\hat{\lambda}_\delta \in \mathcal{M}_\delta}{\operatorname{argmin}} \sum_{v=1}^{V} P_{Q_v^{\mathcal{V}}} \, L_\pi(\hat{\lambda}_\delta(Q_v^{\mathcal{T}}), Q_v^{\mathcal{V}})$$

These discrete superlearner selections $\hat{\lambda}_\delta^{SL}$ are then fitted on the full observed data Q_n and used to estimate $\lambda_\delta(t \mid A, W)$, $F_\delta(t \mid A, W)$, $S(t \mid A, W)$, and $S_c(t\text{-} \mid A, W)$ for $j = 1, \ldots, J$.

Lagged Censoring Survival

Let $\mathcal{S}$ be the set containing all target and observed event times, ordered such that $s_1 < s_2 < \ldots s_{max}$. Then for all $s_K \in \mathcal{S}$ we compute

$$\hat{S}_c(s_K\text{-} \mid A, W) = \exp\left(-\int_0^{s_K\text{-}} \hat{\lambda}_c^{SL}(s \mid A, W) ds\right)$$

Cause-Specific Hazards, Event-Free Survival, and Cause-Specific Absolute Risks

For $l = 1, \ldots, J$ and $s_K \in \mathcal{S}$, the super learner selections $\hat{\lambda}_l^{SL}$ are fit on the full observed data Q_n, and used to compute the event free survival

$$\hat{S}(s_K \mid A, W) = \exp\left(-\int_0^{s_K} \sum_{l=1}^{J} \hat{\lambda}_l^{SL}(s \mid A, W) ds\right)$$

cause-specific absolute risks

$$\hat{F}_l(s_K \mid A, W) = \int_0^{s_K} \hat{S}(s\text{-} \mid A, W) \, \hat{\lambda}_l^{SL}(s \mid A, W) ds$$

Computing the Efficient Influence Function

For each desired treatment regime π^*, each target time τ, and each target event j, the efficient influence functions for each individual are computed in parts.

$$D^*_{\pi^*,j,\tau}(\hat{\lambda}, \hat{\pi}, \hat{S}_c)(O) =$$
$$\sum_{l=1}^{J} \sum_{k=1}^{K} h_{\pi^*,j,l,\tau,s}(\hat{\lambda}, \hat{\pi}, \hat{S}_c)(O) \left(\mathbf{1}(\Delta = j, \widetilde{T} = s_k) - \mathbf{1}(\widetilde{T} \geq s_K) \, \hat{\lambda}_l(s_k \mid A, W)\right)$$
$$+ \sum_{a \in \mathcal{A}} F_j(\tau \mid A = a, W) \, \pi^*(a \mid W) - \Psi_{\pi^*,j,\tau}(P_0)$$

Clever Covariate $h_{\pi^*,j,l,\tau,s}(O)$ For $l = 1, \ldots, J$ and $s \in \mathcal{S}$, the stored cause-specific hazards $\hat{\lambda}_l^{SL}(s \mid A, W)$ and event-free survival $\hat{S}(s \mid A, W)$ are used to calculate the cause-specific absolute risks $\hat{F}_l(s \mid$

$A, W)$, then combined with the nuisance weight to calculate the clever covariates.

$$h_{\pi^*,j,l,\tau,s}(\hat{\lambda},\hat{\pi},\hat{S}_c)(O) = \frac{\pi^*(A \mid W)\,\mathbf{1}(s \le \tau)}{\hat{\pi}^{SL}(A \mid W)\,\hat{S}_c(s\text{-} \mid A, W)}\left(\mathbf{1}(\Delta = j) - \frac{\hat{F}_j(\tau \mid A, W) - \hat{F}_j(s \mid A, W)}{\hat{S}(s \mid A, W)}\right)$$

The clever covariate is a function of the desired intervention density which is user specified, the observed intervention densities which are not changed by tmle targeting, and the outcome-related densities which are updated by targeting.

TMLE one-step update Let D^* be the vector of efficient influence functions

$$D^*(\lambda,\pi,S_c)(O) = \left(D^*_{\pi^*,j,\tau}(\lambda,\pi,S_c)(O) : \pi^* \in \mathcal{A}, j \in \mathcal{J}, \tau \in \mathcal{T}\right)$$

and let $h_{j,s}$ be the vector of clever covariates

$$h_{j,s}(\lambda,\pi,S_c)(O) = \left(h_{\pi^*,j,l,\tau,s}(\lambda,\pi,S_c)(O) : \pi^* \in \mathcal{A}, j \in \mathcal{J}, \tau \in \mathcal{T}\right)$$

The one-step TMLE involves updating the cause-specific hazards along the universal least favorable submodel. This is implemented by updating the hazards in small steps along the sequence of locally-least favorable submodels in the following manner:

$$\hat{\lambda}_{j,\epsilon_m}(t) = \hat{\lambda}^{SL}_j(t)\,\exp\left(\sum_{i=1}^{m} \frac{\left\langle \mathbb{P}_n D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O),\ h_{j,s}(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)\right\rangle_\Sigma}{||D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)||_\Sigma}\,\epsilon_i\right)$$

where

$$\langle x,y\rangle_\Sigma = x^\top \Sigma^{-1} y \quad , \quad ||x||_\Sigma = \sqrt{x^\top \Sigma^{-1} x}$$

The default value of ϵ in the software is 0.1, and the algorithm stops at ϵ_i when

$$\mathbb{P}_n D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O) \le \frac{\sqrt{\mathbb{P}_n\ D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)^2}}{\sqrt{n}\,\log(n)}$$

3.3 getOutput

`getOutput()` takes as an argument the "`ConcreteEst`" object returned by `doConcrete()` and returns the cause-specific risks, risk difference, and/or relative risks. Print and plot methods will be written.

```
ConcreteOut <- getOutput(ConcreteEst)
ConcreteRD <- ConcreteRD$RD[order(Estimator, Time, Event)]
```

4 Full Code Examples

4.1 Competing Risks pbc Analysis

We are often interested in the causal effect of interventions on the time until some outcome occurs. For instance the PBC (primary biliary cholangitis) data set resulted from the Mayo Clinic's randomized controlled trial aimed at determining if D-penicillamine was better than placebo at delaying the death of patients with PBC. In this trial, as in the real world, patients sometimes received life-saving liver transplants - a competing event which we wish to observe jointly with death, thus leading to the full data

$$X = (\tilde{T}^1, \Delta^1, \tilde{T}^0, \Delta^0, W : \tilde{T}^a \le t_{max},\ \Delta \in \{1,2\},\ a \in \{0,1\})$$

For illustration, we consider our causal estimand to be the cause-specific risk differences at at 90 day intervals from years 4 to 6.

$$\Psi^F = \left(P(\tilde{T}^1 \le t, \Delta = 1) - P(\tilde{T}^0 \le t, \Delta = 1), P(\tilde{T}^1 \le t, \Delta = 2) - P(\tilde{T}^0 \le t, \Delta = 2)\ :\ t = 1440, 1530, \ldots, 2160\right)$$

In the pbc trial, as in many time-to-event studies, the events of interest were not observed for many patients either because the study had ended or because they had dropped out of contact while they were still alive. The

occurence of such events, which obscure the observation of the event of interest and which researchers would have ideally prevented, we treat as right-censoring. The observed data then we write with the longitudinal formulation

$$O = N_1(t), N_2(t), N_c(t), A, W \ : \ t \leq T$$

For simplicity let's believe that the necessary identification assumptions are satistified after conditioning on the baseline variables of age, sex and albumin; then the post-intervention distribution for the treatment hypothetical is identified by

$$p_{a=1}(o) = p_0(w)\,\mathbf{1}(A=1)\,\mathbf{1}(\delta \neq 0)\,S_0(\tilde{t}\text{-} \mid A=1,\, w)\prod_{j=1}^{2}\lambda_{0,j}(\tilde{t} \mid A=1, w)^{\mathbf{1}(\delta=j)}$$

with the respective post-intervention distribution for the placebo hypothetical following an analagous structure with $A = 0$. This allows us to identify the causal parameter Ψ^F with the statistical estimand

$$\Psi = (\mathbb{E}\left[F_1(t \mid A=1, W) - F_1(t \mid A=0, W)\right], \mathbb{E}\left[F_2(t \mid A=1, W) - F_2(t \mid A=0, W)\right])$$

for $t = 90, 180, \ldots, 1440$. Let us further define the statistical model as imposing no structure other than what time-ordering would imply, and with this we have a complete statement of the estimation problem.

For initial estimation of nuisance parameters, we use a SuperLearner of `xgboost` and `glmnet` to estimate the propensity score and a Superlearner of main terms Cox as well as a Cox with 2-way interactions. The TMLE update step will be along the universal least favorable model using recursive update steps beginning with a step size of 0.1 with an unweighted efficient influence function norm.

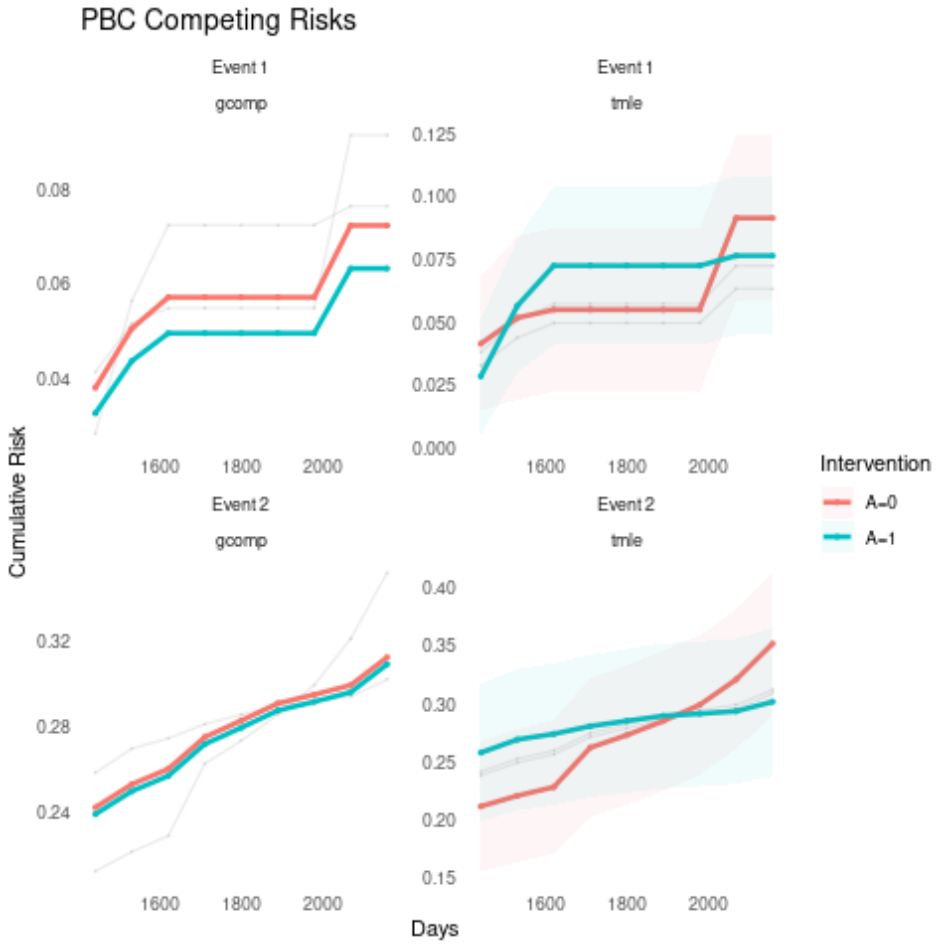
```
library(concrete)
library(data.table)
set.seed(12345)
data <- as.data.table(survival::pbc)
data <- data[!is.na(trt), ][, trt := trt - 1]
data <- data[, c("time", "status", "trt", "age", "sex", "albumin")]

ConcreteArgs <- formatArguments(DataTable = data,
                EventTime = "time",
                EventType = "status",
                Treatment = "trt",
                Intervention = 0:1,
                TargetTime = 90 * (16:24),
                TargetEvent = 1:2)
ConcreteArgs$Model$`1`$model3 <- "~.*."
ConcreteArgs$Model$`2`$model3 <- "~.*."
ConcreteArgs <- formatArguments(ConcreteArgs)

ConcreteEst <- doConcrete(ConcreteArgs)

ConcreteOut <- getOutput(ConcreteEst, "RD")$RD
head(ConcreteOut, 10)
```

Error in head(ConcreteOut, 10) : object 'ConcreteOut' not found



5 Appendix 1: Specific Code Examples

5.1 makeITT

```
ITT <- makeITT()
str(ITT, give.attr = FALSE)
```

The intervention function takes as inputs a vector of observed treatment assignments and data.table of covariates, and outputs a vector of desired treatment assignments. For example, in "A=1" the intervention function returns a vector of 1s the same length as the observed treatment vector.

```
ITT$`A=1`$intervention
```

The 'g.star' function takes as inputs a vector of treatment assignments and data.table of covariates, and outputs a vector of desired treatment probabilities for the provided vector of treatment assignments. In "A=1", the desired intervention is to assign everyone to treatment (i.e. trt = 1) with 100% probability and to control with 0% probability and the corresponding g.star function reflects this, returning 1 if the treatment assignment is 1 and 0 if the treatment assignment is 0.

```
ITT$`A=1`$g.star
```

For "A=0" the intervention function returns a vector of 0s and the treatment assignment probabilities are flipped so that a treatment assignment of 0 is given 100% probability while treatment assignments of 1 are given 0% probability.

```
ITT$`A=0`
```

5.2 Estimating Propensity Score using SuperLearner

```
library(SuperLearner)

# use Superlearner::listWrappers() to show the available models. For additional
    models see https://github.com/ecpolley/SuperLearnerExtra, or create new models
     by modifying "SL.template" or "screen.template"

# simple example
SLModel <- c("SL.glmnet", "SL.bayesglm", "SL.xgboost", "SL.polymars")
# example with screening
SLModel <- list(c("SL.ranger", "screen.corRank"), c("SL.glmnet", "All", "screen.
    randomForest"),
        c("SL.bayesglm", "screen.glmnet"), "SL.polymars")

ConcreteArgs[["Model"]][["trt"]] <- SLModel
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1,
TargetEvent = 1:2, TargetTime = (3:7)*500,
                CVArg = NULL, Model = ConcreteArgs[["Model"]],
                PropScoreBackend = "SuperLearner")
```

5.3 Estimating Propensity Scores using sl3

```
library(sl3)
# use sl3::sl3_list_learners() to show the available models. Use sl3_list_learners(
    properties = ) to list learners appropriate for "binomial", "categorical", or
    "continuous" depending on the type of Treatment variable in your data
sl3glmnet <- Lrnr_glmnet$new()
sl3hal <- Lrnr_hal9001$new()
sl3dbarts <- Lrnr_dbarts$new()

sl3Model <- Stack$new(sl3glmnet, sl3hal, sl3dbarts)
ConcreteArgs[["Model"]][["trt"]] <- sl3Model

ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = (3:7)*500,
                CVArg = NULL, Model = ConcreteArgs[["Model"]],
                PropScoreBackend = "sl3")
```

6 Appendix 2: Nice to have Concepts

6.1 Identification

In order to identify causal estimands such as absolute risk ratios and differences with functions of the observed data, some untestable structural assumptions must hold - namely the assumptions of consistency, positivity, randomization, and coarsening at random on the conditional density of the censoring mechanism.

1. The consistency assumption states that the observed outcome given a certain treatment decision is equal to the corresponding counterfactual outcome

$$T_j^d = T_j \text{ on the event that A = d(L)}$$

1. The positivity assumption states that the desired treatment regimes occur with non-zero probability in all observed covariate strata, and that remaining uncensored occurs with non-zero probability in all observed covariate strata at all times of interest t.

$$P_0\left(A = d(L) \mid W\right) > 0 \text{ , a.e.}$$
$$P(C \geq t \mid a, W) \text{ , a.e.}$$

1. The randomization assumption states that there is no unmeasured confounding between treatment and counterfactual outcomes

$$A \perp\!\!\!\perp (T_1^d, T_2^d) \mid W$$

1. Coarsening at random on censoring

$$C \perp\!\!\!\perp (T_1^d, T_2^d) \mid T > C, A, W$$

Given coarsening at random, the observed data distribution factorizes

$$p_0(O) = p_0(W)\, \pi_0(A \mid W)\, \lambda_{0,c}(\widetilde{T} \mid A,\, W)^{\mathbf{1}(\Delta=0)} S_{0,c}(\widetilde{T}\text{-} \mid A,\, W) \prod_{j=1}^{J} S_0(\widetilde{T}\text{-} \mid A,\, W)\, \lambda_{0,j}(\widetilde{T} \mid A,\, W)^{\mathbf{1}(\Delta=j)}$$

where $\lambda_{0,c}(t \mid A,\, W)$ is the true cause-specific hazard of the censoring process and $\lambda_{0,j}(t \mid A,\, W)$ is the true cause-specific hazard of the j^{th} event process. Additionally

$$S_{0,c}(t \mid a,\, w) = \exp\left(-\int_0^t \lambda_{0,c}(s \mid a,\, w)\, ds\right)$$

while in a pure competing risks setting

$$S_0(t \mid a,\, w) = \exp\left(-\int_0^t \sum_{j=1}^{J} \lambda_{0,j}(s \mid a,\, w)\, ds\right)$$

and

$$\begin{aligned} F_{0,j}(t \mid a,\, w) &= \int_0^t S(s\text{-} \mid a,\, w) \lambda_{0,j}(s \mid a,\, w)\, ds \\ &= \int_0^t \exp\left(-\int_0^s \sum_{j=1}^{J} \lambda_{0,j}(u \mid a,\, w)\, du\right) \lambda_{0,j}(s \mid a,\, w)\, ds. \end{aligned}$$

Under the above identification assumptions, the post-intervention distribution of O under intervention $A = d(a, w)$ in the world of no-censoring, i.e the distribution of $(W,\, T_j^d,\, \Delta_j^d : j = 1, \ldots, J)$, can be represented by the so-called G-computation formula. Let's denote this post-intervention probability distribution with P_d and the corresponding post-intervention random variable with O_d. The probability density of O_d follows from replacing $\pi_0(A \mid W)$ with the density that results from setting $A = d(a, l)$, $\pi_d(d(A, w) \mid W)$, and replacing the conditional probability of being censored at time t by no censoring with probability 1. In notation, $P(O_d = o)$ is given by

$$p_d(o) = p_0(w)\, \pi_d(d(a, w) \mid w)\, \mathbf{1}(\delta \neq 0) \prod_{j=1}^{J} \left[S_0(\tilde{t}\text{-} \mid A = d(a, w),\, w)\, \lambda_{0,j}(\tilde{t} \mid A = d(a, w), w)^{\mathbf{1}(\delta=j)}\right]$$

Recalling the censoring and cause-specific conditional hazards defined above in terms of observed data, we should note that given the identifiability assumptions they now identify their counterfactual counterparts, i.e.

$$\lambda_c(t \mid W,\, A) = \lim_{h \to 0} P(C < t + h \mid C \geq t,\, W,\, A)$$

$$\lambda_j(t \mid W,\, A) = \lim_{h \to 0} P(T < t + h, J = j \mid T \geq t, W, A)$$

Note that the cause-specific event hazards are not conditional on censoring once identifiability assumptions are met.

Since the density $P(O_d = o)$ implies any probability event about O_d, this g-computation formula for $P(O_d = o)$ also implies g-computation formulas for causal quantities such as event-free survival and cause-k absolute risk under intervention d.

7 Old Intro

We are often interested in the causal effect of interventions on the time until some outcome occurs. For instance the PBC (primary biliary cholangitis) data set resulted from the Mayo Clinic's randomized controlled trial aimed at determining if D-penicillamine was better than placebo at delaying the death of patients with PBC. In this trial, as in many time-to-event studies, the failure event (i.e. death) was not observed for many patients either because the study had ended or because they had dropped out of contact while they were still alive. The occurence of such events, which obscure the observation of the event of interest and which researchers would have ideally prevented, is common in time-to-event data and is referred to as right-censoring. On the other hand, other patients received a liver transplant during the study, which saved them from dying from PBC; This however was an outcome that researchers might not have wanted to prevent; after all, it could be important to know if D-penicillamine affects if and when patients receive life-saving transplants. In cases like this when mutually exclusive outcomes are jointly of interest to researchers, we have the case of competing events. With complex survival data, just formulating clear causal questions can be a serious undertaking. Fortunately, the formal causal frameworks developed in recent decades, such as the Neyman-Rubin language of counterfactuals, help us to define unambiguous causal questions and to determine what observed data is needed to answer them.

Of course the task is not finished after causal identification; an estimate must be computed and its uncertainty quantified. When precision and reliability are desired, the choice of appropriate estimators becomes important. The standard unadjusted Kaplan-Meier and Aalen-Johansen estimators are simple and reliable if censoring happens independently of failure events but are inefficient as they do not utilize covariate information and are susceptible to bias when the independence assumption is violated. Alternatively, parametric estimators such as the ubiquitous Cox model perform well but if real process lies within the parametric model; however, this is rarely a known fact in real world data. Instead, we might turn to semi-parametric efficient estimators such as TMLE.

Targeted maximum likelihood-based estimation (TMLE) is a framework for constructing regular and asymptotically linear estimators for pathwise-differential parameters in large statistical models. TMLE has been applied to causally interpretable parameters in many applications, including for survival analysis in discrete-time. The packages 'ltmle', 'stremr', and 'survtmle' can all be applied to discrete-time TMLE of survival estimands, but **concrete** is the first package to implement a continuous-time survival TMLE. 'ltmle' and 'stremr' handle longitudinal treatment and time-dependent confounding using the sequential regression TMLE, while 'survtmle' targets discrete survival outcomes with a TMLE of a discrete-time hazard-based influence function.

When real-world data is collected on a fine enough time scale to be considered more or less continuous, the choice of discretization becomes non-trivial. With longitudinal treatments and time-dependent confounding, the ramifications of discretization are serious to the point of re-defining the causal model. This remains an open practical problem as continuous-time TMLE for these longitudinal problems is not yet implemented. However, even in the single time-point intervention, non-longitudinal causal problems discretization is not a choice without ramifications. Overly coarse discretization may cost the estimator efficiency, while overly fine discretization may result in biased or non-converging estimates of small hazards. Perhaps equally problematic, the estimates resulting from different discretization choices can be different; what is the right way to interpret these different estimates and what is the correct a causal interpretation? Similarly to the choice of not assuming a small parametric model without ample justification, we believe that here we should continue to respect what is truly known about the data; if the data is truly continous, then we should analyze it with a continuous-time method. concrete is the first R package implementing a continuous-time TMLE for survival estimands.

Bibliography

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. [p1]

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. [p4]

P. J. Bickel, C. A. J. Klaassen, Y. Ritov, and J. A. Wellner. *Efficient and Adaptive Estimation for Semiparametric Models*. Springer-Verlag, New York, 1998. ISBN 978-0-387-98473-5. [p4]

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]. [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. [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. [p3]

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. [p4]

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. [p4]

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. [p3]

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. [p4]

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. [p2]

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. [p4]

H. C. Rytgaard, T. A. Gerds, and M. J. van der Laan. Continuous-time targeted minimum loss-based estimation of intervention-specific mean outcomes. *arXiv:2105.02088 [math, stat]*, May 2021. URL http://arxiv.org/abs/2105.02088. arXiv: 2105.02088. [p3, 4]

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, 4, 9, 11]

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. [p1]

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. [p1]

A. W. v. d. Vaart, S. Dudoit, and M. J. v. d. Laan. Oracle inequalities for multi-fold cross validation. *Statistics & Decisions*, 24(3), Jan. 2006. ISSN 0721-2631. doi: 10.1524/stnd.2006.24.3.351. [p4]

M. Wallace, E. E. M. Moodie, D. A. Stephens, and G. S. a. J. Schulz. DTRreg: DTR Estimation and Inference via G-Estimation, Dynamic WOLS, Q-Learning, and Dynamic Weighted Survival Modeling (DWSurv), Sept. 2020. URL https://CRAN.R-project.org/package=DTRreg. [p1]

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]. [p1]