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 cause-specific absolute risks for time-to-event outcomes measured in continuous time. The point treatment 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

concrete is a statistical estimation package written to help researchers answer causal questions about time-to-event outcomes in continous time. When we analyze data and make predictions to guide future actions we are in essence asking causal questions, questions that can be made rigorous using the formal causal inference frameworks developed in recent decades (Pearl et al. (2016), Holland (1986)). To guide the sometimes treacherous transition from causal thinking into statistical estimation, it helps to follow a structured roadmap (Petersen and van der Laan (2014)) that begins with the causal model and causal estimands, moves through identification to a statistical model and estimands, and then finally performs estimation and provides inference. Detailed discussions of these first two stages can be found elsewhere but will only be briefly demonstrated in this manuscript, as **concrete** is fundamentally a statistical estimation package, albeit one that is written with this causal inference workflow in mind.

1.1 What is in this manuscript

We write for readers looking for a hands-on introduction to the one-step targeted maximum likelihood estimation (TMLE) method for continuous time survival analysis described in Rytgaard and van der Laan (2021), as well as for readers wishing to use this TMLE for their own continuous time survival or competing risks analyses. In Section 3 we briefly overview the one-step TMLE method as it is implemented in **concrete** and in Section 2 we show how to use **concrete** for a continuous-time competing risks analysis. For a full and rigorous treatment of the one-step TMLE for continuous-time survival and competing risks analyses, see Rytgaard and van der Laan (2021) and Rytgaard et al. (2021).

1.2 What can concrete be used for

concrete 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. The currently implemented estimators are the one-step TMLE and a g-formula plug-in, and both can be applied to data involving baseline covariate confounding, right-censoring and competing risks.

concrete cannot yet analyse multiple treatments, continuous or multinomial treatments, or stochastic interventions. Methods have not yet been implemented to account for paired or clustered data, time-dependent treatments (e.g. drop-in) or time-dependent confounding. Currently only Cox models can be used for initial estimation of conditional hazards; incorporation of estimators based on highly adaptive lasso (HAL) and penalized cox is planned for a future package version.

concrete is not meant to be used for left trunctation (i.e. delayed entry) or interval censored data and it will not check identification assumptions.

1.3 How concrete relates to other peoples work

concrete will be the first R package implementing TMLE for continuous-time survival. The **ltmle** (Schwab et al. (2020)), **stremr** (Sofrygin et al. (2017)), and **survtmle** (Benkeser and Hejazi (2019)) packages either natively or can be adapted to perform discrete-time TMLE for 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 work in discrete time and would necessitate discretizing continuous-time data, a practice which can in cases introduce bias and decrease estimator efficiency.

The *Causal Inference* CRAN Task View shows just **riskregression** (Gerds et al. (2022)) as expressly estimating treatment effect estimands in survival settings. **riskregression** implements the g-formula plug-in, IPTW, and double-robust AIPTW estimators. The *Survival* CRAN Task View shows no packages to be

implementing efficient semi-parametric estimators for survival estimands, while not yet available on CRAN are the packages adjustedCurves (Denz et al. (2022)) and CFsurvival (Westling et al. (2021)) which implement the AIPTW and a cross-fitted doubly-robust estimator respectively.

1.4 What concrete does

concrete implements the one-step TMLE developed in Rytgaard and van der Laan (2021) to estimate cause and intervention specific absolute risks, with asymptotic inference derived from the variance of the efficient influence curve (EIC) (Laan and Robins (2003)). **concrete** can also return risk differences and risk ratios computed from these absolute risks with inference derived using the delta method on the EICs of the constituent absolute risks.

Broadly speaking, 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)).

As it is often impossible to know in advance which estimator is best suited for a particular estimation problem, the use of a cross-validated machine learning ensemble with oracle guarantees (Super Learner Laan et al. (2007), Polley et al. (2021), Laan and Dudoit (2003), Vaart et al. (2006)) is helpful for the initial estimation of nuisance parameters. A Super Learner with a robust candidate library and a suitable loss function gives users the best chance of achieving the necessary nuisance estimation convergence rates for TMLE to be consistent and efficient. Guidance on how to best specify Superlearner estimators are discussed further in Phillips et al. (2022). Whenever possible, Super Learner libraries should include the highly adaptive lasso (HAL) which achieves the needed convergence rate (Laan (2017); Benkeser and Van Der Laan (2016); Rytgaard et al. (2021)) for TMLE's asymptotic properties.

The subsequent targeted update is based in semi-parametric efficiency theory (Laan and Rose (2011), Kennedy (2016)), specifically that a regular, asymptotically linear estimator of a statistical estimand is efficient if its influence function is equal to the target estimand's EIC. By updating initial estimators of nuisance parameters to solve the EIC, TMLE recovers asymptotically valid inference despite potentially using flexible algorithms without well understood limiting behaviour for initial estimation.

1.5 concrete in a nutshell

```
# Prepare Data
library(concrete)
library(data.table)
library(tidyverse)
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")]

# Specify Analysis
ConcreteArgs <- formatArguments(DataTable = data,
                EventTime = "time",
                EventType = "status",
                Treatment = "trt",
                Intervention = 0:1,
                TargetTime = 90 * (16:24),
                TargetEvent = 1:2)
# Compute
ConcreteEst <- doConcrete(ConcreteArgs)

# Return Output
ConcreteOut <- getOutput(ConcreteEst, c("RD", "Risk"))
```

2 Usage

concrete was written for causal analyses of time-to-event data, though it can also be used for purely statistical estimation problems. There are 3 main user-facing functions in **concrete**: `formatArguments()`, `doConcrete`, and `getOutput`. Specification of the estimation problem is done through input into `formatArguments()`, which checks the estimation specification and return errors, warnings, and messages as necessary. The output

of `formatArguments()` is a `"ConcreteArgs"` object which is then passed into `doConcrete()` to perform the specified continuous-time one-step survival TMLE. The output of `doConcrete()` is a `"ConcreteEst"` object which can be passed into getOutput to print, summarize, and plot cause-specific absolute risk derived estimands such as risk differences and relative risks. For example, a competing risks analysis of the pbc dataset is detailed below.

```
library(data.table)
library(concrete)
set.seed(12345)
data <- as.data.table(survival::pbc)[!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)

ConcreteEst <- doConcrete(ConcreteArgs)

ConcreteOut <- getOutput(ConcreteEst, c("RD", "Risk"))
```

2.1 formatArguments()

Most inputs into `formatArguments()` are involved in specifying one of three tasks: specifying the observed data structure, the target estimand, or the TMLE update procedure. `formatArguments` sanity checks its arguments, i.e. the specified analysis, and returns an object of class `"ConcreteArgs"` with elements that can and sometimes should be modified by the user before passing the `"ConcreteArgs"` object back through `formatArguments` to be re-checked. This process can be repeated as necessary until the full estimation problem is adequately specified.

Data

concrete requires data to be in the general form described in Eq. (4), with the observed time to first event (censoring or otherwise) $\widetilde{T}$, the indicator of which event occured (Δ, with $\Delta = 0$ indicating right-censoring), the intervention variable A, and a collection of baseline covariates W. This data must not include missing values; imputation of missing covariates should be done prior to passing data into **concrete** while data with missing treatment or outcome values (other than right-censoring) is not supported by **concrete**. Uniquely identifying subject IDs can be passed into `formatArguments()` through the `ID=` argument, though functionality for using subject IDs for analyzing clustered or longitudinal data has not yet been implemented.

Using the PBC dataset as our example, $\widetilde{T}$ is the column `"time"`, Δ is the column `"status"`, A is the column `"trt"`, and W consists of all the columns containing patient information observed at baseline. This data is passed into **concrete** as the following:

```
library(concrete)
library(data.table)
set.seed(0)
obs <- as.data.table(survival::pbc)
obs <- obs[,  c("time", "status", "trt", "id", "age", "albumin", "sex")]
obs <- obs[!is.na(trt), ]
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time",
                EventType = "status", Treatment = "trt",
                ID = "id", Intervention = 0:1)
```

Target Estimand

concrete implements a continuous-time one-step TMLE targeting absolute risk derived estimands indexed by interventions, target events, and target times.

Intervention For a binary A and static interventions d setting all observations to $A = 0$ or $A = 1$, then the intervention can specified `formatArguments(Intervention = c(0,1)`.

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

More complex dynamic interventions are passed into `formatArguments(Intervention =)` as 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**. These functions can take treatment and covariates as arguments and must produce either treatment assignments or treatment probabilities respectively, each with the same dimensions as the observed treatment. The function `makeITT()` creates list of functions corresponding to binary static interventions, which can be used as a template for more complex interventions.

Target Events The `TargetEvent =` argument specifies the event types of interest. For **concrete** event types must be integer valued, with 0 by default reserved for censoring. If no input is supplied and the default `TargetEvent = NULL` is used, then all observed non-zero event types will be targeted. If input is supplied for `TargetEvent =` , then all other observed event types will be treated as right-censoring.

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,

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

whereas for a competing risks analysis one could either leave `TargetEvent = NULL` or specify `TargetEvent = 1:2`

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

Target Time The `TargetTime=` argument specifies the time(s) at which estimates of the cause-specific absolute risks or event-free survival are desired. 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.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time",
                EventType = "status", Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = 90 * (16:24))
```

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`. It should be noted here that `Model` is used here to conform with common usage in statistical software, rather than to refer to statistical or causal models.

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 implement a simple 10-fold cross-validation scheme.

```
CVArgs <- list(n = nrow(obs), V = 10L, 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 = 90 * (16:24),
                CVArg = CVArgs)
```

Estimating Nuisance Parameters concrete accepts estimator specifications for estimating nuisance parameters through the argument `formatArguments(Model= )`. Inputs into the `Model=` argument must be named lists with one entry for the intervention variable, and for each of the event type including censoring. The list element corresponding to intervention must be named after the variable and the list elements corresponding to each event type must be named for the numeric value of the event type ("0" for censoring). If no input is provided for the `Model=` argument, `formatArguments()` will return a correctly formatted list, `.[["Model"]]`, containing default estimator specifications for each nuisance parameter, which can be then edited by the user.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = 90 * (16:24),
                CVArg = NULL, Model = NULL)
str(ConcreteArgs[["Model"]], give.attr = FALSE)
```

Propensity Score In concrete, propensity scores are by default estimated using the SuperLearner package `formatArguments(PropScoreBackend = "Superlearner")` with candidate algorithms `c("xgboost","glmnet")` implemented by packages xgboost and glmnet. Alternatively the sl3 package can be used by specifying `formatArguments(PropScoreBackend = "sl3")`.

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. 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.

```
DefaultHazardModels <- list("model1" = "~ trt",
                "model2" = "~ .")
```

One-step TMLE Specification As detailed by Eq. (12) and (13), the one-step TMLE update step involves recursively updating cause-specific hazards, summing along small steps ϵ_i. The value of ϵ is provided by the user as input into the argument `formatArguments(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 the sum in Equation (12) approximates an integral; therefore `OneStepEps` is halved whenever an update step would increase the norm of the efficient influence function.

The `formatArguments(MaxUpdateIter= )` argument is provided to provide a definite stop to the recursive TMLE update. This argument takes positive integers and is set to a default of 100. More updates may be needed when support for targeted estimands in the data is low and 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.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = 90 * (16:24),
                CVArg = NULL, Model = NULL,
                PropScoreBackend = "SuperLearner", HazEstBackend = "coxph",
                MaxUpdateIter = 100, OneStepEps = 0.1, MinNuisance = 0.05)
```

ConcreteArgs object `formatArguments()` returns a list object of class `"ConcreteArgs"`. This object can be modified by the user and then passed back through `formatArguments()` in lieu of supplying new inputs directly into `formatArguments()`.

```
ConcreteArgs <- formatArguments(DataTable = obs, EventTime = "time", EventType = "
    status",
                Treatment = "trt", ID = "id",
                Intervention = 0:1, TargetEvent = 1:2, TargetTime = 90 * (16:24),
                CVArg = NULL, Model = ConcreteArgs[["Model"]],
                PropScoreBackend = "SuperLearner", HazEstBackend = "coxph",
                MaxUpdateIter = 100, OneStepEps = 1, MinNuisance = 0.05)

ConcreteArgs <- formatArguments(ConcreteArgs)
```

2.2 doConcrete()

Once `formatArguments()` runs satisfactorily, the resulting object of class `"ConcreteArgs"` can be passed into the `doConcrete()` function. `doConcrete()` will then perform the specified TMLE algorithm and output an object of class `"ConcreteEst"` which will contain contains TMLE point estimates and influence curves for the cause-specific absolute risks for each targeted event at each targeted time. If `formatArguments(GComp=TRUE)`, then the `"ConcreteEst"` 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)
```

Detailed explanations of the one-step TMLE for continuous-time absolute risk derived estimands can be found in Rytgaard and van der Laan (2021) and Rytgaard et al. (2021). This manuscript briefly reviews this estimation procedure in Section 3.4 and details how a TMLE is specified in **concrete** in Section 2.1.3, subsections 2.1.3.1 through 2.1.3.5. Here we will name the specific functions called in `doConcrete()` which perform each of the steps of the one-step continuous-time survival TMLE procedure.

The cross-validation specification (Section 2.1.3.1) is checked and evaluated in `formatArguments()`, returning fold assignments as `.[["CVFolds"]]` of the `"ConcreteArgs"` object.

The initial estimation of nuisance parameters (Section 2.1.3.2) is performed by the function `getInitialEstimate()`; `getPropScore()` estimates propensity scores (Section 2.1.3.3) and `getHazEstimate()` estimates the conditional hazards (Section 2.1.3.4).

The one-step TMLE update procedure (Sections 3.4 and 2.1.3.5, Equations (10), (11), (12), and (13)) is performed by `doTmleUpdate()` with `getEIC()` computing the efficient influence curves (10).

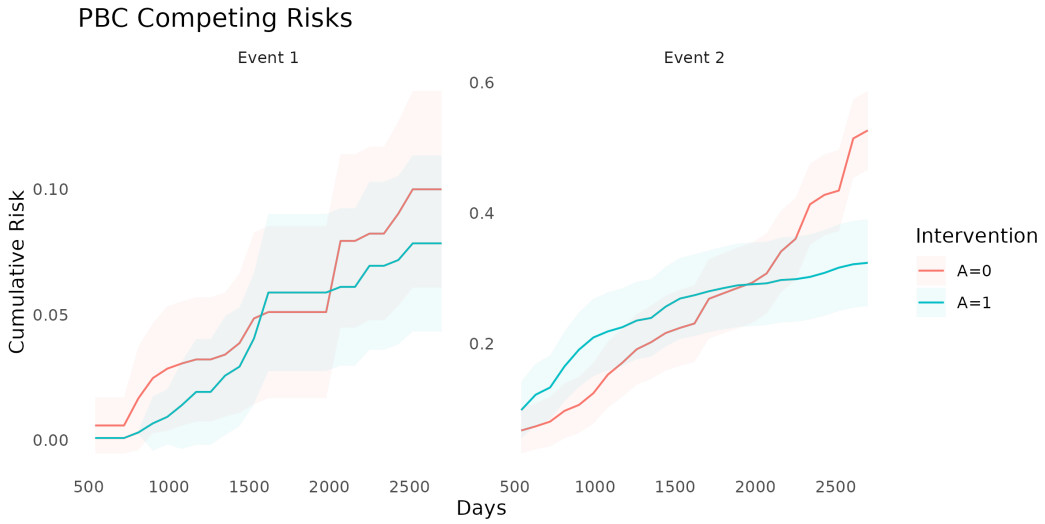
2.3 getOutput()

`getOutput()` takes as an argument the `"ConcreteEst"` object returned by `doConcrete()` and returns tables and produces plots of the cause-specific risks, risk differences, and/or relative risks. By default `getOutput()` returns a data.table with point estimates and pointwise standard errors.

```
library(tidyverse)
RD <- dplyr::filter(ConcreteOut, Estimand == "Risk Diff", Estimator == "tmle") %>%
  pivot_wider(names_from = Event, names_prefix = "J=",
          values_from = c("Pt Est", "se"), names_sep = ": ")
head(RD)
```

Estimand	Estimator	Time	Pt Est: J=1	Pt Est: J=2	se: J=1	se: J=2
Risk Diff	tmle	540	-0.005013989	0.03124389	0.005695705	0.02852565
Risk Diff	tmle	630	-0.005013989	0.04850332	0.005695705	0.03023138
Risk Diff	tmle	720	-0.005013989	0.05229562	0.005695705	0.03172624
Risk Diff	tmle	810	-0.013646841	0.06816388	0.010565078	0.03472190
Risk Diff	tmle	900	-0.018072829	0.08465917	0.012495621	0.03657222
Risk Diff	tmle	990	-0.019239700	0.08537494	0.013861139	0.03844921

From the above table of risk differences we can see that effect of treatment seems to be beneficial for the second event (death) and detrimental for the first event (transplant), though neither effect would be significant at 95% significance level. However, estimands that involve many time points results are often more easily interpreted with graphs.



In the above figures, we can more clearly see the effect of treatment on transplant and death and especially how the effect varies over time.

3 Appendix 1: Concepts

3.1 The Targeted Learning Roadmap

A basic targeted learning roadmap for analyzing continuous-time survival or competing risks consists of:

1. Defining the causal model and specifying the causal estimand (e.g. causal risk difference at time t). Considerations include defining a time zero and a time horizon, specifying the event(s) of interest, identifying the intervention (i.e. treatment) variable and specifying the desired intervention(s).
2. Stating the assumptions needed to define a statistical model with a statistical estimand that identifies the causal estimand. Considerations include identifying confounding variables and sources of right-censoring, establishing positivity for remaining uncensored and following desired interventions, and formalizing knowledge about the statistical model (e.g. dependency structure or functional structures such as proportional hazards)
3. Estimation and inference. Considerations include pre-specification, estimator consistency and efficiency within a desired class, and both theoretical and practical robustness and usability.

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

With time-to-event data, the counterfactual outcome is the time until some event(s) occur to subjects if they hypothetically received an 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 to set A to some value a, $d = a$. A more flexible dynamic treatment rule might assign treatments based on subjects' baseline covariates, while stochastic treatment rules may even depend on the natural treatment assignment mechanism. Whatever the desired intervention, let d represent the intervention of interest and let W represent a subject's baseline covariates. Then a counterfactual survival dataset with J competing events, an intervention d delivered at baseline time zero, and a time horizon of t_{max} takes the form:

$$X = \left(\widetilde{T}^d, \Delta^d, W\right) \quad , \quad T^d \in (0, t_{max}] \tag{1}$$

$\widetilde{T}^d \in \mathbb{R}^+$ is earliest occurence of any of the J events under intervention d and $\Delta^d \in \{1, \ldots, J\}$ shows which of the J events occured first under intervention d.

Importantly, counterfactual data does not incorporate censoring; the counterfactual events are only those events that experimenters would allow to occur in their ideal hypothetical experiment. If the ideal experiment would track just the incidence of one event, then the causal problem is one of classic survival; if multiple events would be allowed to compete, then the causal problem is one with competing risks.

Causal estimands can then be described as functions of this counterfactual data. For instance, to contrast the effects of interventions d^* and d^{**}, we can write the causal event-j relative risk at time t as Eq. (2) and the causal difference in event-free survival at time t as Eq. (3).

$$P(T_j^{d^*} \leq t, \Delta^{d^*} = j) / P(T_j^{d^{**}} \leq t, \Delta^{d^{**}} = j) \tag{2}$$

$$P(T_j^{d^*} < t) - P(T_j^{d^{**}} < t) \tag{3}$$

Researchers may also be interested in these estimands at multiple timepoints, potentially up to the full risk curves over some time interval, and in the case of competing risks potentially for multiple event types as well.

3.3 Observed Data, Identification, and Statistical Estimands

Observed survival data with J competing events takes the form:

$$O = \left(\widetilde{T},\, \widetilde{\Delta},\, A,\, W\right) \tag{4}$$

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

To link causal estimands such as Eq. (2) and (3) to statistical estimands, we need the following untestable identification assumptions to hold: consistency, positivity for treatments and remaining uncensored, no unmeasured confounding, and coarsening at random on the censoring process (details in Appendix 4.1). Given these assumptions, the cause-j absolute risk at time t under intervention d is identified by the g-computation formula

$$\begin{aligned} F_j^d(t) &= \mathbb{E}_{\mathcal{W}}\left[\mathbb{E}_{\pi^d}\left[F_j(t \mid a, w)\right]\right] \\ &= \mathbb{E}_{\mathcal{W}}\left[\int_{\mathcal{A}}\left[\int_0^t \lambda_j(s \mid a, w)\, S(s- \mid a, w)\, ds\right] \pi^d(a \mid w)\, da\right] \end{aligned} \tag{5}$$

where $\lambda_j(t \mid a, w) = \lim_{h \to 0} \frac{1}{h} P(\widetilde{T} \leq t + h, \widetilde{\Delta} = j \mid \widetilde{T} \geq t, a, w)$ is the cause-j conditional hazard, $S(t \mid a, w) = \exp\left(-\int_0^t \sum_{j=1}^{J} \lambda_j(s \mid a, w)\, ds\right)$ is the conditional event-free survival, and $\pi^d(a \mid w)$ is the treatment propensity implied by the intervention d. With the identification result in Eq. (5), the causal absolute risk (2) and survival (3) estimands can then be identified by statistical estimands (6) and (7).

$$\mathbb{E}_{\mathcal{W}}\left[\mathbb{E}_{\pi^*}\left[F_j(t \mid a, w)\right]\right] / \mathbb{E}_{\mathcal{W}}\left[\mathbb{E}_{\pi^{**}}\left[F_j(t \mid a, w)\right]\right] \tag{6}$$

$$\left[1 - \sum_{j=1}^{J} \mathbb{E}_{\mathcal{W}}\left[\mathbb{E}_{\pi^*}\left[F_j(t \mid a, w)\right]\right]\right] - \left[1 - \sum_{j=1}^{J} \mathbb{E}_{\mathcal{W}}\left[\mathbb{E}_{\pi^{**}}\left[F_j(t \mid a, w)\right]\right]\right] \tag{7}$$

It should be noted that even if the identification assumptions do not hold, these statistical estimands in Eq. (6) and (7) may still have valuable interpretations as standardized risks isolating the importance of the "intervention" variable (Laan (2006)).

3.4 Estimation

concrete implements the one-step continuous time TMLE for estimands derived from cause-specific, intervention-specific absolute risks, which begins with estimating the treatment propensity π, the conditional hazard of censoring λ_c and the conditional hazard of each event $\lambda_j : j = 1, \ldots, J$. Super Learners are used to estimate these nuisance parameters, which involves specifying the cross-validation scheme, libraries of candidate algorithms, the cross-validation loss functions, and the Super Learner meta-learner.

Cross-Validation

For a V-fold cross validation scheme, let $Q_n = \{O_i\}_{i=1}^n$ be the observed n i.i.d observations of $O \sim P_0$ and 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 a training set $Q_v^{\mathcal{T}} = \{O_i : B_n(i) = v\}$ with the corresponding validation set $Q_v^{\mathcal{V}} = \{O_i : B_n(i) \neq v\}$. For guidelines on selecting the number of folds and when stratification may be helpful, it may be helpful to consult Phillips et al. (2022).

Libraries should be constructed from candidate algorithms that vary in flexibility and respect pre-existing knowledge about the data-generating mechanism. For instance, Cox models should incorporate domain knowledge about which covariates may be most predictive of event times and if n is much greater than the

number of covariates then more flexible candidate Cox parameterizations along with flexible algorithms such as Highly Adaptive Lasso (HAL). If on the other hand the number of covariates is not much smaller n, then fewer and less flexible candidate algorithms should be used, potentially either including penalization natively or being paired with covariate screening algorithms.

Super Learner candidates should be evaluated on a cross-validated loss function that is minimized by the true data-generating process and for maximal robustness a discrete selector that simply selects the best performing candidate should be used as the Super Learner metalearner.

Initial Estimate of Treatment Propensity

For estimating the treatment propensity, let $\pi_0(\cdot \mid W)$ be the true conditional distribution of A given W, $\mathcal{M}_\pi = \{\hat{\pi} : Q_n \to \hat{\pi}(Q_n)\}$ be the candidate library of propensity score estimators, and L_π be a loss function such that the risk $\mathbb{E}_0\left[L_\pi(\hat{\pi}, O)\right]$ is minimized when $\hat{\pi} = \pi_0$. The discrete Superlearner estimator is then the the candidate propensity estimator $\hat{\pi} \in \mathcal{M}_\pi$ that has minimal cross validated risk

$$\hat{\pi}^{SL} = \underset{\hat{\pi} \in \mathcal{M}_\pi}{\operatorname{argmin}} \sum_{v=1}^{V} P_{Q_v^{\mathcal{V}}} \, L_\pi(\hat{\pi}(Q_v^{\mathcal{T}}), Q_v^{\mathcal{V}}) \tag{8}$$

Initial Estimate of Conditional Hazards

For estimating the conditional hazards, let $\lambda_{0,\delta} \; : \; \delta = 0, \ldots, J$ be the true conditional hazards for censoring ($\delta = 0$) and events ($\delta \in \{1, \ldots, J\}$). Let $\mathcal{M}_\delta = \{\hat{\lambda}_\delta : Q_n \to \mathbb{R}\}$ for $\delta = 0, \ldots, J$ be the libraries of candidate Cox hazard specifications for the censoring and cause-specific hazards and let $L_\delta(\beta) = -\sum_{i=1}^{n}\left[\beta W_i - \log\left[\sum_{h \in \mathcal{R}(\tilde{T}_h)} \exp(\beta W_h)\right]\right]$ be the negative log Cox partial-likelihood loss function. The discrete SuperLearner selector for each δ chooses the candidate $\hat{\lambda}_\delta \in \mathcal{M}_\delta$ that has minimal cross validated risk

$$\hat{\lambda}_\delta^{SL} = \underset{\hat{\lambda}_\delta \in \mathcal{M}_\delta}{\operatorname{argmin}} \sum_{v=1}^{V} P_{Q_v^{\mathcal{V}}} \, L_\pi(\hat{\lambda}_\delta(Q_v^{\mathcal{T}}), Q_v^{\mathcal{V}}) \; : \; \delta = 0, \ldots, J \tag{9}$$

Efficient Influence Curve

The treatment propensity estimator Eq. (8) and conditional hazard estimators Eq. (9) are used to estimate the nuisance parameters that make up the EICs of absolute-risk derived estimands like Eq. (6) and (7). Parameters that contrast multiple cause-specific absolute risks at multiple time points involve vector EICs comprised of the absolute risk EICs for each targeted event, each target time, and each intervention. For the event j, time t, and intervention propensity π^* absolute risk ($\Psi_{\pi^*,j,t}(P_0) = F_j^{\pi^*}(t)$), the corresponding vector EIC element is:

$$\begin{aligned} D^*_{\pi^*,j,t}(\lambda, \pi, S_c)(O) = \sum_{l=1}^{J} \int h_{\pi^*,j,l,t,s}(\lambda, \pi, S_c)(O) \, \left(N_l(ds) - \mathbf{1}(\widetilde{T} \geq s) \, \lambda_l(s \mid A, W)\right) \\ + \sum_{a \in \mathcal{A}} F_j(t \mid A = a, W) \, \pi^*(a \mid W) - \Psi_{\pi^*,j,t}(P_0) \end{aligned} \tag{10}$$

where $N_l(s) = \mathbf{1}\left\{\widetilde{T} \leq s, \widetilde{\Delta} = l\right\}$ are the cause-specific counting processes and $h_{\pi^*,j,l,t,s}(\lambda, \pi, S_c)(O)$ is the TMLE "clever covariate" with the form

$$h_{\pi^*,j,l,t,s}(\lambda, \pi, S_c)(O) = \frac{\pi^*(A \mid W) \, \mathbf{1}(s \leq t)}{\pi(A \mid W) \; S_c(s\text{-} \mid A, W)} \left(\mathbf{1}(\Delta = j) - \frac{F_j(t \mid A, W) - F_j(s \mid A, W)}{S(s \mid A, W)}\right) \tag{11}$$

where $F_j(t \mid a, w)$ is the conditional cause-j absolute risk, $S_c(t \mid A, W)$ is the conditional censoring survival, $S(t \mid A, W)$ is the conditional event-free survival, and $N_j(t) = \mathbf{1}\{\widetilde{T} \leq t, \Delta = l\}$ is the event-j counting process. The treatment propensity π and the conditional event and censoring hazard functions (λ_c, $\lambda_j \; : j = 1, \ldots, J$) are directly estimated with Eq. (8) and (9) while the conditional absolute risks and survivals are computed from the hazard estimates as described in Section 3.3. The clever covariate is a function of the intervention propensity, observed conditional distributions which are not changed by TMLE targeting, and lastly the outcome-related conditional distributions which are updated by targeting.

The one-step continuous-time survival TMLE updates the cause-specific hazards along the universally

least favorable submodel in the following manner:

$$\lambda_{j,\epsilon}(t) = \lambda_j(t)\ \exp\left(\int_0^{\epsilon} \frac{\left\langle \mathbb{P}_n \tilde{D}^*(\lambda_x, \pi, S_c)(O),\ h_{j,s}(\lambda_x, \pi, S_c)(O)\right\rangle_{\Sigma}}{||\tilde{D}^*(\lambda_x, \pi, S_c)(O)||_{\Sigma}}\ dx\right) \tag{12}$$

where

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

$\tilde{D}^*$ is the vector of efficient influence functions

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

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

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

In practice this integral is approximated by recursively summing along a series of locally least favorable models and ends when

$$\mathbb{P}_n D^*(\lambda_\epsilon, \pi, S_c)(O) \leq \frac{\sqrt{\mathbb{P}_n\ D^*(\lambda_\epsilon, \pi, S_c)(O)^2}}{\sqrt{n}\ \log(n)} \tag{13}$$

The variance of the EIC divided by the sample size, $\frac{\mathbb{P}_n\ D^*(\lambda_\epsilon,\pi,S_c)(O)^2}{n}$, is a consistent estimator of the asymptotic variance of asymptotically linear estimators, and is used in this package to estimate the variance of the TMLE cause-specific absolute risk estimator. In the presence of significant positivity violations however, this EIC-derived variance estimator will be anti-conservative.

4 Appendix 2: More Concepts

4.1 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.

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

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

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

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.

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. [p]

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. [p]

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]. [p]

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. [p]

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.]. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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]. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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. [p]

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]. [p]