concrete R Paper

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

Introduction

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 (Laan and Robins (2003)).

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.

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,

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).

Concepts

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.

In the continuous-time survival case, the targeted learning roadmap consists of:

1. causal considerations including defining a time zero and time horizon, specifying the event(s) of interest, identifying the intervention variable and specifying the desired interventions, and defining causal estimand(s) (e.g. evaluation time(s), difference vs. ratio)
2. statistical considerations such as identifying confounding variables, identifying right-censoring, establishing positivity for desired interventions and remaining uncensored, formalizing knowledge about the statistical model (e.g. dependency/independence structure, proportional hazards, etc.)
3. estimation using a pre-specified estimator

The Causal Model: Counterfactuals, Interventions, and Causal Estimands

Counterfactual survival data with J competing events takes the form:

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

where the superscript d indicates desired intervention, $\widetilde{T}^d \in \mathbb{R}^+$ is earliest occurence of any event under that intervention, $\Delta^d \in \{1, \ldots, J\}$ indicates which event occurs first under that intervention, and W is a set of covariates measured at baseline.

Equation (2) is the causal event-j absolute risk at time t under intervention d and Equation (3) is the causal event-free survival at time t under intervention d.

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

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

Observed Data

Observed survival data with J competing events takes the form:

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

where A is the intervention variable, $\widetilde{T} \in \mathbb{R}^+$ is the earlier of the time to first observed event or the onset of right-censoring, $\Delta \in \{0, \ldots, J\}$ indicates which event occurs (with 0 indicating right-censoring), and W is a set of covariates measured at baseline.

Given the identification assumptions of consistency, positivity for treatments and remaining uncensored, no unmeasured confounding, and coarsening at random on the censoring process, the causal survival (3) and risk (2) estimands can be identified by statistical estimands which are purely functions of the observed data. If the identification assumptions (stated in detail in Appendix 2.5.1) do not hold, then these statistical estimands still have an interpretation as standardized risks isolating the importance of the "intervention" variable (Laan (2006)).

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) which consists

of two major steps: an initial estimation of nuisance parameters utilizing flexible machine learning and a subsequent targeted update of initial estimators to solve the efficient influence function of the target statistical estimand (Laan and Robins (2003); Kennedy (2016)).

Adequate convergence is needed in the initial estimation stage so **concrete** uses 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 the cause-specific absolute risk at every target time $\tau$for every desired treatment regime π^* and every target event $j$is:

$$D^*_{\pi^*,j,\tau}(\hat{\lambda},\hat{\pi},\hat{S}_c)(O) = \sum_{l=1}^{J}\sum_{k=1}^{K} h_{\pi^*,j,l,\tau,s}(\hat{\lambda},\hat{\pi},\hat{S}_c)(O)\ \Big(\mathbf{1}(\Delta=j,\widetilde{T}=s_k) - \mathbf{1}(\widetilde{T}\geq s_K)\,\hat{\lambda}_l(s_k \mid A,W)\Big) + \sum_{a\in\mathcal{A}} F_j(\tau \mid A=a,W)\,\pi^*(a \mid W) - \Psi_{\pi^*,j,\tau}(P_0) \tag{5}$$

where

$$h_{\pi^*,j,l,\tau,s}(\hat{\lambda},\hat{\pi},\hat{S}_c)(O) = \frac{\pi^*(A \mid W)\,\mathbf{1}(s\leq\tau)}{\hat{\pi}^{SL}(A \mid W)\ \hat{S}_c(s\text{-} \mid A,W)}\left(\mathbf{1}(\Delta=j) - \frac{\hat{F}_j(\tau \mid A,W) - \hat{F}_j(s \mid A,W)}{\hat{S}(s \mid A,W)}\right) \tag{6}$$

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.

The one-step continuous-time survival TMLE updates the cause-specific hazards in small steps along the sequence of locally-least favorable submodels in the following manner:

$$\hat{\lambda}_{j,\epsilon_m}(t) = \hat{\lambda}^{SL}_j(t)\ \exp\left(\sum_{i=1}^{m}\frac{\langle \mathbb{P}_n D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O),\ h_{j,s}(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)\rangle_\Sigma}{||D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)||_\Sigma}\,\epsilon_i\right) \tag{7}$$

where

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

D^* is the vector of efficient influence functions

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

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

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

The one-step TMLE algorithm stops at the ϵ_i when

$$\mathbb{P}_n D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O) \leq \frac{\sqrt{\mathbb{P}_n\ D^*(\hat{\lambda}_{\epsilon_i},\hat{\pi},\hat{S}_c)(O)^2}}{\sqrt{n}\,\log(n)} \tag{8}$$

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`. Reflecting our vision of good statistical practice, the majority of user effort is directed into defining the desired analysis through 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 can be passed into getOutput to print, summarize, and plot cause-specific absolute risk derived estimands such as risk differences and relative risks.

formatArguments()

Arguments into `formatArguments` fall into 3 broad categories: specifying the observed data structure, specifying the target estimand, and specifying the estimation algorithm. `formatArguments()` will check its inputs and return errors, warnings, and messages as necessary. The output of `formatArguments` is an object of class `"ConcreteArgs"`, with fields that can 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

The general form of observed right-censored survival data, potentially with competing events, is

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

where $\widetilde{T}$ is the observed time to first event (censoring or otherwise), Δ indicates 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 columns containing patient information observed at baseline. A column containig uniquely identifying subject IDs can be passed into **concrete** as well. This data is passed into **concrete** as the following:

```
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 jointly targeting the cause-specific absolute risks at certain target times under some hypothetical treatments.

Treatment Let A be the intervention variable and d be an intervention on the variable. If A is binary and d is a static intervention setting everyone to 0 or setting everyone to 1, then the intervention can be passed into **concrete** through the argument `formatArguments(Intervention =` as "0" or "1" respectively. Both interventions can be specified by passing 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**.

Target Events In the pbc dataset, there are 3 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 `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 so `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 concrete:::getTargetTime(TargetTime = unique(obs[status > 0, : TargetTime must not target times after which all individuals are Censored, 4191

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 = 90 * (16:24))
```

Estimator Specification

The arguments involved in estimation are the cross-validation setup `CVArg`, the estimation algorithms `Model`, the software backends `PropScoreBackend` and `HazEstBackend`, and the TMLE specification choices `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 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 $Q_v^{\mathcal{V}} = \{O_i : B_n(i) \neq v\}$.

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 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 $L_\pi(\hat{\pi}, O) = -\log\left(\hat{\pi}(1 \mid W)^A \, \hat{\pi}(0 \mid W)\right)^{1-A}$. concrete uses a 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 $\pi_0(A \mid W)$.

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")`.

Estimating Event and Censoring Hazards 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 Cox models, $\{\hat{\lambda}_\delta : Q_n \to \hat{\lambda}_\delta(Q_n)\}$, for the censoring and cause-specific hazards and let L_δ be the log pseudo-likelihood loss function such that the risks $\mathbb{E}_0\left[L_\delta(\hat{\lambda}_\delta, O)\right]$ are minimized when $\hat{\lambda}_\delta = \lambda_{0,\delta}$. A discrete SuperLearner selector for each δ chooses the candidate $\mathcal{M}_\circ$ 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 $j = 1, \ldots, J$.

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 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) and summarized briefly here in Equations (7) and (8) in Section 2.2.4

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 (7) 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 can use the argument `formatArguments(MaxUpdateIter= )` to set the maximum number of small update recursions, i.e. i for ϵ_i in Equation (8). This argument takes positive integers and is set to a default of 100.

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 as separate arguments 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)
```

doConcrete

Once `formatArguments()` runs without errors, the resulting object of class `"ConcreteArgs"` should be a suitable input into the function `doConcrete()`. `doConcrete()` will then perform the specified estimation 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)
```

For a detailed and precise description of the one-step TMLE for continuous-time survival, see Rytgaard and van der Laan (2021) and Rytgaard et al. (2021). This manuscript briefly reviews the one-step TMLE for continuous-time survival in Section 2.2.4 and details the required steps in Section 2.3.1.3, subsections 2.3.1.3.1 through 2.3.1.3.5. Here we will list the functions called by `doConcrete()` which perform each of the steps in performing the one-step continuous-time survival TMLE.

The cross-validation scheme (Section 2.3.1.3.1) is checked by `formatArguments()` and if possible evaluated, outputting fold assignments as `.[["CVFolds"]]` of the `"ConcreteArgs"` object.

The initial estimation of nuisance parameters (Section 2.3.1.3.2) is performed by the function `concrete:::getInitialEstimate()`, with propensity estimation (Section 2.3.1.3.3) performed by `concrete:::getPropScore()` and hazard estimation (Section 2.3.1.3.4) performed by `concrete:::getHazEstimate()`.

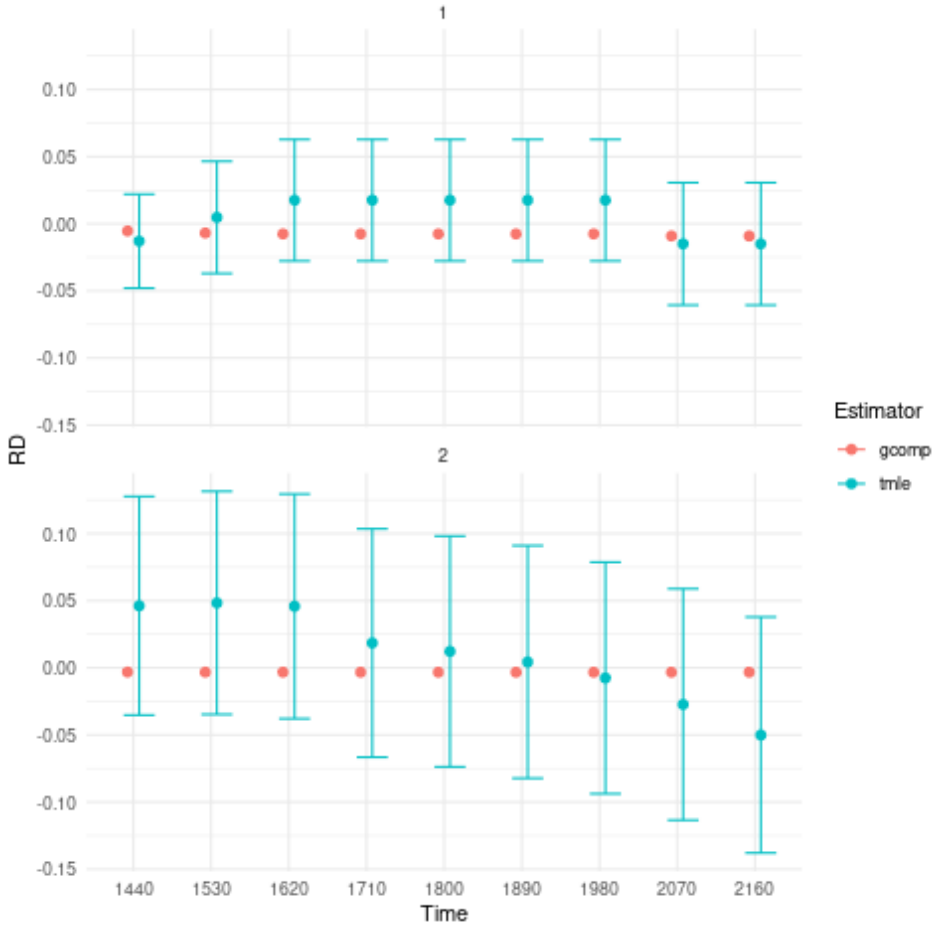
The one-step TMLE update procedure (Sections 2.2.4 and 2.3.1.3.5, Equations (5), (6), (7), and (8)) is performed by `concrete:::doTmleUpdate()` with `concrete:::getEIC()` computing the efficient influence curves (5).

getOutput

`getOutput()` takes as an argument the `"ConcreteEst"` object returned by `doConcrete()` and returns tables and plots of the cause-specific risks, risk differences, and/or relative risks.

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

Estimator	Event	Time	RD	se
tmle	1	1440	-0.012989172	0.01786192
tmle	2	1440	0.046210334	0.04154612
tmle	1	1530	0.004761007	0.02131398
tmle	2	1530	0.048368036	0.04239952
tmle	1	1620	0.017551051	0.02310111
tmle	2	1620	0.045792211	0.04269689
tmle	1	1710	0.017551051	0.02310111
tmle	2	1710	0.018490620	0.04338313
tmle	1	1800	0.017551051	0.02310111
tmle	2	1800	0.012207170	0.04385966



simulations?

concrete performance (vs SuperLearner g-formula and survtmle on discretized data)?

concrete performance with bad specification choices?

????

Appendix 2: Nice to have Concepts

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.

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. [p3]

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. [p3]

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]

E. H. Kennedy. Semiparametric theory and empirical processes in causal inference. In *Statistical causal inferences and their applications in public health research*, pages 141–167. Springer, 2016. [p3]

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. [p2]

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. [p3]

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. [p3]

M. J. v. d. Laan and J. M. Robins. *Unified Methods for Censored Longitudinal Data and Causality*. Springer Series in Statistics. Springer-Verlag, New York, 2003. ISBN 978-0-387-95556-8. [p1, 3]

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. [p3]

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. [p3]

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. [p2, 3, 8]

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, 2, 7, 8]

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. [p3]

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]