Super Learner and Targeted Maximum Likelihood Estimation for Longitudinal Data Structures with Applications to Atrial Fibrillation

by

Jordan Chamberlain Brooks

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Biostatistics

in the

Graduate Division

of the

University of California, Berkeley

Spring 2012

Super Learner and Targeted Maximum Likelihood Estimation for Longitudinal Data Structures with Applications to Atrial Fibrillation

Abstract

Super Learner and Targeted Maximum Likelihood Estimation for Longitudinal Data Structures with Applications to Atrial Fibrillation

by

Jordan Chamberlain Brooks

Doctor of Philosophy in Biostatistics

University of California, Berkeley

Professor Mark J. van der Laan, Chair

This thesis discusses the Super Learner and Targeted Maximum Likelihood Estimation (TMLE) for longitudinal data structures in nonparametric statistical models. It focuses specifically on time-dependent data structures where the outcome of interest may be described as a counting process. A Super Learner for the conditional intensity of the counting process is proposed based on the minimization of squared error and negative Bernoulli loglikelihood risks. An analytic comparison of the oracle inequality for the cross validation selector of the squared error and negative Bernoulli loglikelihood risks is provided. TMLE is extended to enforce a calibration property, defined as an implicit constraint, on the Super Learner. Tradeoffs between calibration and risk minimization are explored through simulation. The final chapter discusses and implements a recently developed TMLE for the intervention-specific marginal mean in general longitudinal data structures. A modification of this general TMLE algorithm template is implemented to respect model constraints for the estimation of the cumulative event probability in survival data with time-dependent covariates and informative right-censoring. Each chapter is supplemented with practical examples of these estimators using data from the the Kaiser Permanente ATRIA-1 cohort study of adults with atrial fibrillation. The primary appendix presents a new Super Learner software implementation as a SAS (Statistical Analysis System) macro for data-adaptive machine learning. SAS macro code for the TMLEs are provided in secondary appendices.

To my family

Contents

List of Tables

Acknowledgments

Thank you to everyone who has supported me throughout my education. All my accomplishments are shared with family, friends, colleagues, and mentors. This dissertation represents the closing of an important chapter in my life. Professor Mark van der Laan has been a brilliant advisor and supportive mentor throughout the process. His ability to make the the most advanced concepts accessible and applicable is uncanny. I want to thank my committee members Alan Hubbard and John Colford for their help, and acknowledge the support of Kaiser Permanente Division of Research for providing the data and research questions that motivated much of the work presented here. I have had the pleasure of navigating the PhD program with and with the help of an exceptional group of fellow students and administrators. I could not have completed this journey without my family. My mother and her ongoing encouragement are blessings for which I am thankful everyday. Last but not least, to my Susan. The best for us is yet to come. I love you.

Chapter 1

Introduction

Nonparametric statistical models are characterized by a minimal set of assumptions on the nature of stochastic data. Because of this they allow realistically complex representations of true underlying probability distributions, and thus provide a very general approach to learning from any observed data structure. Not surprisingly, the flexibility of general nonparametric statistical models also makes estimation of their parameters a reasonably difficult task and an active area of statistical research. The Super Learner (van der Laan et al., 2007) and Targeted Maximum Likelihood Estimation (TMLE) (van der Laan and Rubin, 2006) are complimentary methods for parameter estimation in nonparametric statistical models for general data structures. The theory underlying these methods have been developed over the last eight years within Mark van der Laan's research group at the University of California, Berkeley. The Super Learner theory guides the construction of asymptotically optimal estimators of non-pathwise-differentiable parameters, e.g., prediction or density estimation, and the TMLE theory guides the construction of efficient estimators of finite dimensional pathwise-differentiable parameters, e.g., marginal means. This dissertation presents applications of these general estimation templates to the specific case of longitudinal time-dependent data structures where the outcome of interest can be characterized as a counting process.

The introductory material here presents an overview of estimation in nonparametric statistical models, followed by discussion of the general theory for Super Learner and TMLE. It concludes with a brief clinical overview of atrial fibrillation and a description of the ATRIA-1 cohort study used to illustrate the practical application of the methods presented herein. Chapter 2 presents the Super Learner for the conditional intensity of a counting process. Chapter 3 combines TMLE for and Super Learner for prediction function calibration, with illustrative examples in both point treatment data structures and time-dependent data structures with counting processes outcomes. Chapter 4 discusses TMLE for the estimation of intervention-specific marginal means in general longitudinal data structures, and presents an implementation for the estimation of the marginal cumulative event probability in right-censored survival data with time-dependent covariates. Each chapter investigates the statistical properties of the estimators presented therein and assesses their performance against competing estimators via simulation studies. Applications of the methods are demonstrated to answer research questions in the ATRIA-1 cohort study.

Derivations, proofs, and software are reserved for the appendices. Appendix A presents a SAS macro %SUPERLEARNER that was used to implement the Super Learners used throughout this dissertation. The motivation, concept, and design are discussed along with the practical inputs and outputs of the macro. Simulation studies are presented, and the full SAS code is provided. Appendix B provides a series of efficient influcence curves for the calibration parameters presented in Chapter 3 and SAS code to carry out the TMLE calibration procedure. Appendix C contains the the efficient influence curve for the intervention-specific marginal mean paramater in longitduinal data structures and provides SAS code to carry out the TMLE algorithm illustrated in Chapter 4.

1.1 Conventional versus nonparametric statistical models

Formally, a statistical model is a set of assumptions or constraints that defines a class of probability distributions. The model provides a rigorous framework on which to construct and assess estimators of the parameters of the distribution of the data. This section illustrates some of the key differences between estimation in conventional parametric and nonparametric statistical models using a simple point treatment data structure, $O = (W,A,Y) \sim P_0 \in \mathscr{M}$, where $W \in \mathbb{R}^d, A \in \{0,1\}$, and $Y \in \mathbb{R}$. This means our data O consists of random variables: W a d-dimensional vector of covariates, a binary treatment A, and a continuous outcome Y, with joint probability distribution, P_0. This joint distribution is a particular member of the class of distributions bounded by the assumptions of the statistical model $\mathscr{M}$.

Suppose now that we observe a set of n statistically independent data points, $O_1,\ldots,O_n \overset{\text{iid}}{\sim} P_0$. The two most common problems in applied statistics are prediction and effect estimation. In prediction, the goal is to estimate the density or conditional expectation of the outcome as a function of the covariate values. In effect estimation, the goal is to determine a measure of effect or association of the treatment on the outcome. In the illustrative example here, the prediction parameter will be the conditional expectation function and the effect parameter will be the mean additive change in Y for a one-unit change in A, controlling or adjusting for W.

The conventional approach for prediction and effect estimation in this data structure often works within a parametric linear regression model, $E_0[Y|A,W] = \alpha + A\beta + W\gamma$, for which prediction and effect estimation are trivial tasks. Prediction amounts to the estimation of the α, β, and γ coefficients, and the effect parameter is simply the β. If this model contains the truth, then efficient estimates can be computed with standard ordinary least squares regression software. Often, however, such a simple model is a poor representation of the true relationships. For example, the model assumes that the effect of A on the expectation of Y is equal for all levels of W. If this is not true, then the estimation of the β coefficient under this working model may result in bias for the true mean additive effect. Similarly, the additivity assumptions on the covariates may result in poor performance on the prediction task. The story becomes messier when working with binary outcomes, $Y \in \{0,1\}$, under the conventional parametric logistic regression model $E_0[Y|A,W] = \frac{1}{1+e^{-(\alpha+A\beta+W\gamma)}}$. This model is multiplicative in the odds, and has no coefficient to explicitly represent the mean additive effect parameter.

Nonparametric statistical models are general and place few if any restrictions on the class of probability distributions for the data. Unlike the parametric linear regression model, there are no coefficients that are sufficent to identify the parameters of interest. We must instead return to first principles and identify the parameter based on the axioms of probability. Without loss of generality, we can always factorize the joint density, or likelihood, or our data into a series of marginal and conditional densities. For notational purposes made apparent later, denote the factors of the likelihood:

$$\begin{aligned} dP_0(0) &= p_0(0) \\ &= p_0(W,A,Y) \\ &= p_0(W)p_0(A|W)p_0(Y|A,W) \\ &= Q_{W,0} \times g_{A,0} \times Q_{Y,0} \end{aligned}$$

In our approach the conditional densities of the covariates and outcome are denoted $Q_0 = (Q_{W,0}, Q_{Y,0})$ and the conditional density of the treatment or intervention is denoted $g_0 = g_{A,0}$. As before, the prediction task is the estimation of the conditional expectation function parameter, $\bar{Q}_{Y,0} = E_0[Y|A,W]$. Under the nonparametric model, however, we do not assume any particular functional form. Instead, we assume that the parameter is a member of a class of possibly infinite-dimensional functions (with mean 0 given the covariates) that map the treatment and covariate values to the real line. This is an extremely broad class of functions and finding an estimator that is closest to the truth is no trivial task. Prediction in nonparametric models has seen numerous contributions from both the statistical and machine learning communities. Examples include data-adaptive regressions, regularizers, recursive partitioning, neural networks, nearest neighbors, support vectors, and kernel methods, and some of the more flexible methods have even been proven to achieve asymptotic consistency. Nonetheless, the size of the class of potential prediction functions precludes estimation efficiency in the conventional sense. Instead the optimality of a prediction estimator is defined in terms of the rate at which it becomes in some sense "close" to $E_0[Y|A,W]$, where closeness is defined in terms of a formal dissimilarity. The same applies to a density estimator. The Super Learner is an asymptotically optimal estimator of conditional densities and conditional expectations in this sense.

Like the prediction task, effect estimation in the nonparametric model is not trivial. Without parametric assumptions on the functional form of the regression model, we must carefully define what "adjusting for W" really means. If this refers to the effect of A on Y averaged over all values of W, then the additive effect parameter can be written as the functional mapping $\psi_0 = \Psi(P_0) = E_0\big[E_0[Y|A=1,W] - E_0[Y,W|A=0,W]\big]$. This parameter only depends on the Q-factors $\Psi(P_0) = \Psi(Q_0) = E_{Q_W,0}[\bar{Q}_{Y,0}(A=1,W) - \bar{Q}_{Y,0}(A=0,W)\big]$, which immediately suggests a substitution estimator. It turns out that the mean additive effect parameter can also be identified as the solution to an estimation equation that involves the conditional density $g_{A,0}$. In general, substitution estimators and estimating equations represent the two most popular approaches estimation of finite-dimensional parameters in nonparametric statistical models. Note, however, that although the additive effect parameter is a scalar, it is identified as a functional mapping that involves the possibly infinite-dimensional density or conditional expectation function factors of the likelihood. Consistency of estimators for effect parameters depends on consistent estimation of the density factors used in their construction. Efficiency of estimation for such parameters is determined by the gradient of the functional mapping. In particular, an estimator is efficient if and only if the gradient of its functional mapping is asymptotically equal to the canonical gradient for the parameter mapping of interest. The canonical gradient is also called the efficient influence curve and plays a central role in the construction of efficient TMLE estimators presented in this dissertation.

1.2 Super Learner

1.2.1 Infinite-dimensional parameter, $Q(P_0)$, as a risk minimizer

The Super Learner is a data-adaptive machine learning approach to prediction and density estimation. In the nonparametric statistical model, tasks of this type correspond to the estimation of an infinite-dimensional parameter, which is a function of the distribution of the data, $Q_0 = Q(P_0)$. Consider from the previous section the prediction problem, i.e., estimation of the conditional expectation function $\bar{Q}_{Y,0} = \bar{Q}_{Y,0}(A,W) = E_0[Y|A,W]$ based on a set of n observations $O_1,\ldots,O_n \overset{\text{iid}}{\sim} P_0$. It is well-known that for some valid risk function, R, $\bar{Q}_{Y,0}$ can be identified in the nonparametric model as the functional mapping

$$\bar{Q}_{Y,0} = \underset{\bar{Q}_Y}{\operatorname{argmin}}\, R_0(\bar{Q}_Y)$$

That is, the conditional expectation is the minimizer of a valid risk function, $R_0 = R(P_0)$. A linear risk function is one that can be expressed as the expectation, with respect to P_0, of a loss function, $\mathscr{L}(O)$, of the unit data structure. A commonly used linear risk is the expectation of the squared error loss function.

$$R_0(\bar{Q}_Y) = E_0[\mathscr{L}(\bar{Q}_Y)(O)] = E_0[(Y - \bar{Q}_Y(A,W))^2]$$

Often there exists an entire class of valid risk functions for a particular infinite-dimensional parameter. For example, the conditional expectation of any outcome $Y \in [0,1]$, which includes binary Y as a special case, may be identified as the minimizer of the squared error risk above or as the minimizer of the negative Bernoulli loglikelihood risk.

1.2.2 Candidate estimators and the ”oracle”

Our goal is to construct an estimator mapping $\hat{\bar{Q}}_{Y,n} = \hat{\bar{Q}}_Y(P_n)$ induced by the empirical distribution, P_n, of our sample that is ”close” to the the true parameter mapping $\bar{Q}_{Y,0} = \bar{Q}_Y(P_0)$. The risk difference is a dissimilarity metric that provides a rigorous definition of closeness.

$$d(\bar{Q}_{Y,0}, \hat{\bar{Q}}_{Y,n}) = R_0(\bar{Q}_{Y,0}) - R_0(\hat{\bar{Q}}_{Y,n})$$

Consider a collection of K candidate estimators constructed based on the same n observations. We call this collection a library and index each candidate estimator $\hat{\bar{Q}}_{Y,n,1},\ldots,\hat{\bar{Q}}_{Y,n,K}$, and their corresponding risks, $R_0(\hat{\bar{Q}}_{Y,n,1}),\ldots,R_0(\hat{\bar{Q}}_{Y,n,K})$. The "oracle" selector is defined as a procedure that chooses the minimum risk candidate estimator, $\tilde{k}_n = \underset{k}{\operatorname{argmin}}\, d(\bar{Q}_{Y,0},\hat{\bar{Q}}_{Y,n,k}) = \underset{k}{\operatorname{argmin}}\, R_0(\hat{\bar{Q}}_{Y,n,k})$. The "oracle" estimator, $\hat{\bar{Q}}_{Y,n,\tilde{k}}$, depends on the unknown distribution of the data P_0, and in finite samples on the empirical distribution, P_n, of the n observations. Because of this, we typically never know the oracle for a particular problem a priori. The oracle does, however, provide a reasonable asymptotic target benchmark in practical data analysis. That is, we would like to construct an estimator that does as well as asymptotically as the oracle in terms of risk difference.

1.2.3 Cross validation

The empirical risk, $R_n(\hat{\bar{Q}}_{Y,n}) = \frac{1}{n}\sum_{i=1}^{n}\mathscr{L}(\hat{\bar{Q}}_{Y,n})(O_i)$, as a plugin estimator is a tempting approach to estimator selection. For flexible data-adaptive estimators, however, this empirical risk can severely underestimate the true risk. This downward (optimistic) bias is widely referred to as overfitting, and reflects the tendency of flexible estimators to incorporate random errors or noise from the empirical distribution P_n into their mappings.

Cross validation is the natural solution to this problem. In brief, the n observations are split into a training set used to construct the estimator mapping and a validation set used to evaluate the "cross validated" risk. This is a nearly unbiased procedure for the estimation of the true risk, even for flexible data-adaptive estimators. Formal definition of the cross validated risk requires a random vector $S_n \in \{0,1\}^n$ to split the observed data into training and validation subsamples. For the i^{th} subject

$$S_{n,i} = \begin{cases} 0 & \text{if the i}^{th}\text{ observation is in the training set} \\ 1 & \text{if the i}^{th}\text{ observation is in the validation set} \end{cases}.$$

If we denote the empirical distributions of the training and validation subsamples P^0_{n,S_n} and P^1_{n,S_n}, respectively, and let $p = \frac{\sum_{i=1}^{n} S_{n,i}}{n}$ be the proportion of observations in the validation set, then the cross validated risk is

$$R^{CV}_{n,p}(\hat{\bar{Q}}_Y) = E_{S_n}\left[\int \mathscr{L}(\hat{\bar{Q}}_{Y,P^0_{n,S_n}})(o)dP^1_{n,S_n}(o)\right] = \frac{1}{np}\sum_{i=1}^{n} S_{n,i}\mathscr{L}(\hat{\bar{Q}}_{Y,P^0_{n,S_n}})(O_i)$$

The cross validation selector is defined as the procedure that indexes the minimum cross validate risk candidate estimator, $\hat{k}_n = \underset{k}{\operatorname{argmin}}\, \frac{1}{np}\sum_{i=1}^{n} S_{n,i}\mathscr{L}(\hat{\bar{Q}}_{Y,P^0_{n,S_n},k})(O_i)$, and the cross validation estimator is $\hat{\bar{Q}}_{Y,n,\hat{k}_n}$. van der Laan and Dudoit (2003) proved that for linear risks based on bounded loss functions

$$E[d(\bar{Q}_{Y,0},\hat{\bar{Q}}_{Y,n,\hat{k}_n})] \leq (1+2\lambda)E[d(\bar{Q}_{Y,0},\hat{\bar{Q}}_{Y,n,\tilde{k}_n})] + 2C(\lambda)\frac{1+log(K(n))}{np}$$

where $K(n)$ is the size or cardinality of the library of candidate estimators for the training sample of size n. $\hat{k}_n$ is the cross validation selector, and $\tilde{k}_n$ is the oracle selector for the empirical distribution of our n observations. λ is a positive constant and $C(\lambda)$ is a constant defined by van der Laan and Dudoit (2003). This is the so-called oracle inequality for cross validation. It states that the cross validation selector performs asymptotically as well as the oracle selector. The last term means that we will have this property for any library whose size is allowed to grow at a rate that is polynomial in np. Further, if the oracle converges to the truth at a parametric rate, then the cross validation estimator will also converge at a $\frac{1+log(K(n))}{np}$ rate. This oracle result for the cross validation selector applies generally to several cross validation schemes including V-fold, Monte Carlo, and others, though not leave-one-out cross validation (van der Laan and Dudoit, 2003).

1.2.4 Super Learner

The Super Learner is an estimator based on a generalization of the oracle results for cross validation. Instead of working with a library of individual candidate estimators, van der Laan et al. (2007) extended the library to include an entire class of new candidates defined as convex weighted combinations of the individual candidate estimators. The original article proposed new candidates of a particular parametric form

$$\bar{Q}_{Y,n,\alpha} = m\big(\hat{\hat{Q}}_{Y,n,1}, \ldots, \hat{\hat{Q}}_{Y,n,K} | \alpha\big)$$

where $\alpha = (\alpha_1, \ldots, \alpha_K), \sum_{k=1}^{K} \alpha_k = 1, \alpha_k \geq 0 : k = 1, \ldots, K$. The constraints on α bound the extended library by the convex hull of the individual candidates in the original library. This convexity helps to ensure bounds on the loss, a necessary condition for the cross validation oracle result, and the framework has since proven to work well in practice. The convex Super Learner is then defined as the weighted combination that minimizes the V-fold cross validated risk, defined as follows. Start by defining a random vector $B \in \{1, \ldots, V\}^n$ with a multinomial distribution where each value has probability $\frac{1}{V}$. For the i^{th} subject

$$I(B_i = b) = \begin{cases} 0 & \text{if the i}^{th} \text{ observation is in the } b^{th} \text{ training set} \\ 1 & \text{if the i}^{th} \text{ observation is in the } b^{th} \text{ validation set} \end{cases} .$$

Formally let $\hat{\hat{Q}}_{Y,n,b,k} = \hat{\hat{Q}}_{Y,k}(P_{n,B \neq b})$ be the k^{th} candidate estimator induced by the b^{th} training set. For the i^{th} subject, define a new variable $Z_{b,i} = \{\hat{\hat{Q}}_{Y,n,b,k}(A_i, W_i)) : k = 1, \ldots, K\}$ to be the prediction from the k^{th} estimator induced by the b^{th} training set. Let $Z_i = \big(\sum_{b=1}^{V} I(B_i = b)\hat{\hat{Q}}_{Y,n,b,k}(A_i, W_i) : k = 1, \ldots, K\big)$ give the vector of cross validated predictions for each of the individual candidate estimators in the library. Doing this for all observations in our training sample we now have a matrix $Z_{n \times K}$ consisting of the V-fold cross validated predictions from the original K individual candidate estimators. The Super Learner weights are computed as the solution to the minimization problem

$$\alpha_n = \underset{\alpha}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}\big(m(Z_i|\alpha)\big)(O_i)$$

With the squared error risk the weights are selected as $\alpha_n = \underset{\alpha}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n}(Y_i - Z_i\alpha)^2$, and the convex Super Learner is defined $\hat{\bar{Q}}_{Y,n,SL} = \sum_{j=1}^{K} \alpha_{n,j}\hat{\bar{Q}}_{Y,n,j}$. If the outcome takes values in $[0,1]$, one might also consider weights selected as $\alpha_n = \underset{\alpha}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left[-2\big(logit(Z_i)\alpha(1-Y_i) + log(1+e^{-logit(Z_i)\alpha}\big)\right.$, where $logit(x) = log(\frac{x}{1-x})$. This corresponds with a Super Learner for the conditional expectation parameter defined as the minimizer of the negative Bernoulli loglikelihood risk. In this case the convex Super Learner estimator is $\hat{\bar{Q}}_{Y,n,SL} = \frac{1}{1+e^{-\sum_{j=1}^{K} logit(\hat{\bar{Q}}_{Y,n,j})\alpha_{n,j}}}$. The oracle result for the convex Super Learner is based on the inequality

$$\frac{1}{V}\sum_{v=1}^{V} E[d(\bar{Q}_{Y,\alpha_n}, \bar{Q}_{Y,0})] \leq (1+\delta)E[min_{\alpha}\frac{1}{V}\sum_{v=1}^{V} d(\bar{Q}_{Y,\alpha}, \bar{Q}_{Y,0})]$$

Generally, Super Learners can be constructed for any a parameter defined as a risk minimizer. Typical examples include prediction or conditional expectation functions, and density estimation. Each Super Learner is defined by:

- Library of candidate estimators
- Risk function
- Cross validation framework
- Function to combine the outputs of candidate estimators

While prediction and density estimation are important statistical problems in their own right, the Super Learners for such parameters are also critical components in the construction of estimators of finite dimensional parameters in nonparametric statistical models. The Super Learners discussed in the main text of this dissertation are specific implementations of this general template. The focus, however, is specific to Super Learners for nonpathwise differentiable parameters in longitudinal time-dependent data structures with an outcome characterized as a counting process. Chapter 2 in particular constructs a Super Learner for the conditional intensity of a counting process in right-censored survival data with time-dependent covariates.

1.3 Targeted Maximum Likelihood Estimator (TMLE)

1.3.1 Finite dimensional parameter, $\Psi(P_0)$, and efficient influence curve

Finite dimensional pathwise differentiable parameters are defined as a functional mapping from the statistical model to a vector of real numbers. $\psi_0 = \Psi(P_0), \Psi : \mathscr{M} \to \mathbb{R}^d$. For these types of parameters, the pathwise derivative evaluated at the distribution P_0 is a unique mean 0 random function called the efficient influence curve, D^*, of the parameter. The variance of D^* in the nonparametric statistical model is akin to the Fisher Information in a parametric model in that it defines the efficiency bound for regular estimators of ψ_0.

Often the parameter of interest depends on P_0 only through the Q-factors, $Q_0 = Q(P_0)$, i.e., the (conditional) densities of the covariates and the outcome. In such cases, the remaining factor, i.e., the conditional density of the treatment $g_0 = g(P_0)$, is a nuisance parameter. The additive effect of A on Y adjusted averaged over W from section 1.1 is an elementary example. Here, $\psi_0 = \Psi(Q_0) = \Psi(Q_{W,0}, \bar{Q}_{Y,0}) = E_{Q_{W,0}}\left[\bar{Q}_{Y,0}(1,W) - \bar{Q}_{Y,0}(0,W)\right]$. The efficient influence curve for this parameter, however, involves both the Q- and g-factors of the likelihood.

$$D^*_{Q_0,g_0} = \left(\frac{I(A=1)}{\bar{g}_{A,0}(W)} - \frac{I(A=0)}{1-\bar{g}_{A,0}(W)}\right)(Y - \bar{Q}_{Y,0}(A,W)) + \bar{Q}_{Y,0}(1,W) - \bar{Q}_{Y,0}(0,W) - \psi_0$$

1.3.2 Substitution estimators and estimating equations

The definition of our parameter as a functional mapping immediately suggests a substitution estimator based on plugins for $\bar{Q}_{Y,0}$ and $Q_{W,0}$. A major advantage of plugin substitution estimators is that they always respect global model constraints. Further, the maximum likelihood principle can be used as a criterion to select potential solutions. Consistency of the typical plugin estimator $\psi_n = \Psi(Q_{W,n}, \hat{\bar{Q}}_{Y,n})$ depends entirely on the consistency of the Q-factor plugins. Because of this, the construction of a Super Learner for each component might be considered a reasonable approach. However, it should be noted that the Super Learner minimizes the global risk for the Q-factor parameters without any regard for the bias-variance tradeoff for the estimation of ψ_0. It turns out that this procedure can and often does lead to bias for the additive effect parameter of interest. Another difficulty with this approach is that there is no theory that can be used to construct analytic estimates of the variance, and instead one must is forced to work with computationally intensive bootstrap procedures for inference.

Estimating equation methods provide a complimentary approach to the estimation of pathwise differentiable parameters. Here the parameter is identified as the solution to a particular estimating equation. Perhaps the most widely used is the Inverse Probability of Censoring Weighted (IPCW) influence curve estimating equation

$$\sum_{i=1}^{n}\left(\frac{I(A=1)}{\bar{g}_{A,0}(W)} - \frac{I(A=0)}{1-\bar{g}_{A,0}(W)}\right)Y - \psi_0 = 0$$

The IPCW estimator is consistent only if the a plugin estimate, for $\bar{g}_{A,0}(W)$ is consistent. Unlike the standard substitution esitmator described above, the IPCW is regular and asymptotically linear, which means that its variance may be estimated analytically as the estimated variance of its influence curve divided by the sample size. A disadvantage of the IPCW is that it is not locally efficient, but it can be made so by augmenting the estimating equation such that the solution solves the efficient influence curve estimating equation. The fact that the augmented IPCW solves the efficient influence curve implies that it is doubly robust in the sense of Bickel et al. (1993). In brief, the augmented IPCW is consistent if the estimates of either the Q- or g-factors of the likelihood are consistent, and it is locally efficient if both are consistent. Estimating equation approaches, however, suffer two major drawbacks. First, inconsistent estimates of the g-factors may lead to estimates that do not respect global model constraints. For example, it is not uncommon to estimate probabilities outside the interval [0,1]. Second, in the case where the estimating equation has several solutions, there is no obvious criterion for selecting amongst them.

1.3.3 Targeted maximum likelihood estimators

TMLE was first introduced by van der Laan and Rubin (2006). Broadly speaking, the TMLE is a two-stage estimator that seeks to combine the most favorable characteristics of estimating equation approaches with those of substitution-based maximum likelihood estimators. The TMLE is constructed around the efficient influence curve D^*, and, like the augmented IPCW, inherits the double robustness and local efficiency properties. The TMLE, however, does not use D^* to construct an efficient estimating equation. Instead a "least favorable" parametric submodel (Bickel et al., 1993) is constructed in such a way that its score (derivative of the loglikelihood) spans D^*. Standard parametric maximum likelihood is used to estimate the coefficients of the submodel that solve the efficient score equation. This approach allows TMLEs to be applied to a very broad class of parameters including those that cannot be written as the solution to an estimating equation.

Procedurally, the TMLE is constructed as follows. The first stage constructs an initial fit, Q_n^0, of the relevant Q-factors of likelihood. This is essentially the a typical substitution estimator, with a Super Learner for each component. The second stage "targets" the Q-factor fits obtained from the first stage to reduce bias for the parameter of interest. The least favorable submodel is $Q_0(\varepsilon) = Q_0 + \varepsilon H$, where ε is a finite dimensional parameter and H is a function of the data, such that the score (derivative of the loglikelihood) spans D^*. In the illustrative example used here, $H = H(A,W) = \frac{I(A=1)}{\bar{g}_0(W)} - \frac{I(A=0)}{1-\bar{g}_0(W)}$. Heuristically, the covariate H defines a direction in which we must fluctuate our initial estimator to remove bias for ψ_0, and is therefore called the "clever covariate." The parameter ε of the submodel represents the magnitude of the fluctuation. We now plug in our initial fit, Q_n^0, as the offset, compute H for all observations, and estimate ε using standard maximum likelihood regression software. This gives us our updated fit, $Q_n^* = Q_n(\varepsilon_n)$, and the TMLE is defined $\psi_n^* = \Psi(Q_n^*)$. The TMLE is regular and asymptotically linear, with variance given by the variance of the estimated efficient influence curve. It is a substitution estimator, and as such respects the global model bounds even if the g-factors of the likelihood are misspecified in the construction of the clever covariate H. In general, every TMLE procedure includes the following four ingredients:

1. Initial density estimator, Q_n^0
2. Choice of loss function, $\mathscr{L}(Q)$
3. Parametric fluctuation submodel, $Q_n(\varepsilon)^0 = Q_n^0 + \varepsilon H$

4. Updating step (possibly iterated), $Q_n^* = Q_n^0(\varepsilon_n)$

TMLEs have been developed for the estimation of marginal means or causal effect parameters in several data structures including point treatment (van der Laan and Rubin, 2006; van der Laan and Gruber, 2010; Porter, Gruber, van der Laan, and Sekhon, 2011), right-censored survival (Stitleman and van der Laan, 2010; Stitleman, Wester, De Gruttola, and van der Laan, 2011), longitudinal data structures with time-dependent covariates (van der Laan, 2010a,b), and case-control settings (van der Laan, 2008). TMLE has also been used to estimate variable importance measures (Tuglus and van der Laan, 2011). van der Laan and Rose (2011) provides a comprehensive overview of TMLE in a textbook format.

The TMLEs presented in main text of this dissertation are specific implementations of the general theory. The focus is, however, specific to TMLE for pathwise differentiable parameters in longitudinal time-dependent data structures with an outcome characterized as a counting process. Chapter 3 presents an extension of TMLE to impart calibration properties to a Super Learner for the conditional intensity, though it also covers the more simple point treatment data structure. Chapter 4 implements a recently developed (van der Laan and Gruber, 2011) TMLE for the intervention-specific marginal mean in general longitudinal data structures. The actual implementation is slightly modified to reflect the model constraints in right-censored survival data with time-dependent covariates.

1.4 Atrial fibrillation epidemiology

1.4.1 Atrial fibrillation

Atrial fibrillation (AF) is the most common cardiac arrhythmia in United States adults. As the name implies, AF is characterized by an abnormal fibrillation or quivering in the heart's upper (atrial) chambers. Empirical evidence clearly demonstrates that persons with AF are at markedly increased risk for thromboembolic stroke. The physiological mechanism underlying this increased risk has been described as follows. First, the abnormal quivering of the heart's chambers obstructs the normal flow of blood through the heart. The blood is left to pool in the left atrium, and without intervention this pooled blood may go on to form a hard clot, or thrombus. After some time, the thrombus may dislodge from the heart and move into the bloodstream. Now called a thromboembolism, the clot is free to travel from the larger vessels of the vascular system down to the smaller vessels where it can lodge and effectively block blood flow to end organs, which results in an ischemic injury due to lack of oxygen. Ischemic damage of this type to the brain is commonly called a thromboembolic stroke. The thromboembolic stroke may be contrasted with hemorrhagic strokes associated with ischemia due to bleeding. Brain injury in the form of stroke may be fatal or lead to devastating physical and cognitive disabilities in persons who survive the initial insult.

Warfarin is a commonly prescribed blood thinner that has been shown in clinical trials to be an effective treatment in the prevention of thromboembolism in patients with AF. The physiological mechanism works by interfering with normal clotting factors in the blood. It is therefore not surprising that warfarin usage is also associated with an increased incidence of excessive bleeding events. In fact, bleeding events that occur in the brain are a common cause of hemorrhagic stroke. This presents a difficult problem for clinicians who must weigh the potential benefits of preventing a thromboembolic stroke against the potential consequences of a hemorrhagic stroke via warfarin-induced bleeding. Current clinical guidelines suggest that warfarin treatment should be reserved for patients who are considered to be at high risk for thromboembolic stroke.

1.4.2 ATRIA-1 cohort

ATRIA-1 was a longitudinal follow-up study designed by Kaiser Permanente Northern California (KPNC) to answer several important questions regarding treatment choices in patients with AF. Data from 13,559 KPNC clients with with existing or new diagnoses of AF were collected during years 1997-2003. Information on each patient was collected via KPNC's network of large databases. Demographic variables including age, gender, race, education, and income were pulled from administrative databases. Medical diagnoses were extracted from the electronic medical records system. The diagnoses included hypertension, diabetes mellitus, retinopathy, dialysis, congestive heart failure, ischemic heart disease, coronary artery disease, peripheral arterial disease, history of prior stroke, dementia, falls, circulatory disease, gastrointestinal bleeding, other bleeding, hepatitis, herpes zoster, and seizures. Data on clinical biomarkers including creatinine, proteinuria, HgbA1C, total hemoglobin, total white blood cell count and international normalized ratio were drawn from lab databases. Finally an indicator for whether each patient was taking warfarin was inferred from pharmacy prescription logs. The medical diagnoses, lab values, and warfarin treatment indicators were treated as time-dependent covariates and were updated over the course of the study. The outcomes of interest were stroke or death observed during the follow-up period. The vast majority of participants lived stroke-free throughout the study period. A relatively small number of participants were lost to follow-up because they disenrolled from the KPNC health care system.

1.4.3 Applied research questions

Several practical research objectives from ATRIA-1 are addressed in this dissertation. Chapter 2 constructs a Super Learner for the conditional intensity of the thromboembolic stroke event for patients who are not taking warfarin, while Chapter 3 uses TMLE to calibrate the Super Learner predictions such that the resulting predictor is unbiased for the true stroke rates within the low, medium, and high risk subgroups defined by the predictor itself. That is, our method maps our prediction function into abn efficient estimate of the actual stroke rates amongst persons classified into ”low”, ”medium”, or ”high” risk subgroups. Chapter 4 estimates a causal effect of warfarin in the general AF population setting. This is of interest because the general AF patient population is generally sicker than clinical trial samples typically used to infer benefits. In particular Chapter 4 implements recently developed TMLE to estimate the causal effect of continuous warfarin on the marginal probability of stroke or death within a 1-year time-frame from the start of the study.

Chapter 2

Super Learner for the conditional intensity of a counting process

2.1 Introduction

The Super Learner is an optimal procedure for data-adaptive machine learning of infinite-dimensional nonpathwise differentiable parameters in a nonparametric statistical model. In brief, it is an ensemble estimator that combines the outputs from a collection of user-supplied candidate estimators (the library) that minimizes the expectation of the cross validated loss function. The library may include fully parametric estimators such as generalized linear regressions and/or data-adaptive machine learning algorithms such as decision trees, artificial neural networks, gradient boosting machines, etc. The Super Learner has been proven under mild regularity conditions to perform asymptotically as well as the öracleëstimator, which is defined as the candidate estimator in the user-supplied library that minimizes the true risk (van der Laan et al., 2007). Typically, the Super Learner is applied to prediction, i.e., estimation of the conditional expectation function, and density estimation. While prediction and density estimation are of interest in their own right, they also play a fundamental role in the estimation of several interesting finite dimensional parameters, e.g., marginal means or additive effects (van der Laan and Rose, 2011).

The conditional intensity of a counting process is a nonpathwise differentiable parameter in a nonparametric statistical model for longitudinal time-dependent data. In brief, the conditional intensity tells us the rate of change in an event counting process as a function of the covariate history and for all time-points. In the special case of right-censored survival data the conditional intensity is the hazard function.

This chapter describes and implements a Super Learner for the conditional intensity of a counting process. It begins with a formal definition of the data structure, the nonparametric statistical model, and the conditional intensity parameter as a risk minimizer mapping in this model. For completeness the conventional approaches based on parametric or semiparametric regression procedures is described, and followed with a description of the Super Learner procedure and the statistical properties of the resulting estimator. The chapter is supplemented with an application of the Super Learner for the conditional stroke intensity in atrial fibrillation patients from eh ATRIA-1 cohort study, and concludes with a general discussion of the presented material.

2.2 Data, model, and conditional intensity parameter, $\bar{Q}_{Y(t),0}$

Suppose we are given a time-dependent data structure with discrete time intervals indexed $t = 1,\ldots,\tau$. The discretization of time into intervals is a reasonable choice because in practice, covariates are updated at a particular latency, for example, once per day, week, month, or year. For the time-dependent data structure, the most natural choice for the length of the time interval in the discretized analysis is the unit of lowest latency. That is, if covariates are updated on a daily basis, then the natural long form data would contain one row per subject per day.

Let $L_1(t)$ be an event counting process that starts at 0 and increases by one unit for every event of interest observed. Let $L_2(t)$ denote a vector of covariates observed at time t. Finally let $R(t)$ be an indicator that the subject at is in the "risk set," i.e., we actually observed the subject at time t. By convention, we assume that within any particular time interval, t, the ordering of these is $L_1(t) \to L_2(t) \to R(t)$. As will be made clear, it is also helpful to define the change in the event count just after time t to be $Y(t) = L_1(t+1) - L_1(t)$.

The data for an individual subject grows as we move forward through time. Thus the observed data structure at some time t is given by $O(t) = \big(R(t), R(t)Y(t), R(t)\mathscr{F}_t\big)$, where $\mathscr{F}_t = (\bar{L}_1(t), \bar{L}_2(t))$ denotes the entire history of both L processes up to and including time t. This formulation makes it clear that we cannot observe a change in the event count just after time t if the person is not in the risk set at t. The complete data structure for an individual subject is then given by the collection of these time-specific data over all time points, $O = \big(O(t) : t = 1,\ldots,\tau\big)$. We propose that $O \sim P_0 \in \mathscr{M}$, which says that O is drawn from a particular probability distribution, P_0, which is a member of a class of distributions bounded by the assumptions of the statistical model $\mathscr{M}$, which we will assumed to be nonparametric.

Suppose we wish to estimate the conditional intensity of the event counting process, $L_1(t)$. Heuristically, the conditional intensity tells us the rate of change in the event count based on the covariate history and the time t. We already know with certainty that we will not observe a change in the event count if the subject is not in the risk set. We therefore define our conditional intensity parameter of interest here as the expected change in the counting process given the covariate history and conditional on being in the risk set.

$$\bar{Q}_{Y(t),0} = \bar{Q}_{Y(t)}(P_0) = \Big(E_0[Y(t)|R(t) = 1, \mathscr{F}_t] : t = 0,\ldots,\tau\Big)$$

where the subscript 0 indicates that the expectation is taken with respect to P_0. Note that in the special case of right-censored survival data, the conditional intensity corresponds to the hazard. For the general counting process framework the conditional intensity parameter, $\bar{Q}_{Y(t),0}$, can be identified in the nonparametric statistical model as the minimizer of a valid risk. A risk is a functional that maps our parameter to a real number. The risk is said to be valid if it is minimized by the true parameter $\bar{Q}_{Y(t),0}$. Commonly a valid risk is constructed by taking the expectation of a loss function. If, as is commonly the case $Y(t) \in \{0,1\}$, i.e., the event of interest can only occur only once per subject time point, then the negative Bernoulli loglikelihood risk is a natural choice. Consider this loss function for the intensity at a particular time point t.

$$\mathscr{L}_t(\bar{Q}_{Y(t)})(O(t)) = -2R(t)log\left\{\bar{Q}_{Y(t)}(R(t),\mathscr{F}_t)^{Y(t)}(1-\bar{Q}_{Y(t)}(R(t),\mathscr{F}_t))^{(1-Y(t))}\right\}$$

The conditional intensity as a function over all t, $\bar{Q}_{Y(t),0} = \bar{Q}_{Y(t),0}(t) : t = 1,\ldots,\tau$, may be identified as the minimizer of the risk defined by the expectation of the the loss function defined as the sum over such all t-specific loss functions.

$$\mathscr{L}(\bar{Q}_{Y(t)})(O) = \sum_t \mathscr{L}_t(\bar{Q}_{Y(t)})(O(t))$$

$$\bar{Q}_{Y(t),0} = \underset{\bar{Q}_{Y(t)}}{\text{argmin}}\, E_0[\mathscr{L}(\bar{Q}_{Y(t)})(O)]$$

In a more general counting process structure, where the event of interest may occur multiple times per subject per time point, the negative Bernoulli loglikelihood risk is not valid. In this case, one might consider a smaller interval for each time stamp or one may substitute the squared error loss which is valid for any $Y(t) \in \mathbb{R}$. To simplify the presentation here, we will assume that $Y(t) \in \{0,1\} : t = 0,\ldots,\tau$, i.e., that subjects can experience only one event per time interval.

2.3 Conventional approach

Suppose we observe a sample $O_1,\ldots,O_n \sim P_0$ from which we would like to construct an estimator of of the conditional intensity $\bar{Q}_{Y(t),0}$. When faced with this type of data, the conventional wisdom in applied work suggests a Cox proportional hazards regression (Cox and Oakes, 1984) for time-dependent covariates. This procedure maximizes the profile likelihood of the log hazard ratio parameter under a proportional hazards assumption. Violation of the proportional hazards assumption, which leads to biased estimates, is especially likely in high dimensional settings with multiple covariates. Further, the empirical baseline hazard estimator used in this regression requires smoothing to become a decent estimator of the conditional intensity, where the choice of smoother is secondary problem on its own.

Fully parametric approaches to hazard estimation do not require smoothing, per se, but instead rely on a set of model assumptions regarding its functional relationship to the covariates and the time point. Another method that has become increasingly popular is to work with a standard logistic regression, which allows the investigator to encode model assumptions via the function form of the linear component on the right hand side of the regression equation. Finally, machine learning learning techniques such as adaptive regression splines, generalized additive models, artificial neural networks, decision trees, and boosting algorithms have also become more common.

Procedurally, each approach works with time-dependent data in the long format matrix comprised of one row per subject per time point [Note: standard Cox regression software, however, often only requires one row for each time interval where covariate values have changed.] The estimation procedure essentially becomes one of predicting a binary outcome as a function of the covariate history and time. In general, the more parametric estimators exhibit stability at the cost of bias due to model misspecification, while the data-adaptive machine learning methods can be very unbiased but tend to overfit the random error in the observations used for estimation leading to high variability. In general the conventional approach involves the investigators favorite method possibly with some fine tuning to fit the data at hand.

For a given set of observations, the minimum risk estimator, called the ”oracle”, amongst a library of potential candidate estimators is never known a priori. It is dependent upon several factors including both the true probability distribution as well as the empirical distribution of the observations at hand. Because no single estimator is optimal for all true and empirical distributions, the conventional approach is not optimal for estimation in nonparametric models.

2.4 Cross validation

The general Super Learner approach to estimation in the nonparametric statistical model is to consider a broad class of candidate estimators and select a combination of these that minimizes an unbiased estimate of the risk. While the empirical risk is reasonably unbiased in a priori specified parametric regressions, it is widely known to underestimate the risk of more complex data-adaptive algorithms such as recursive partitioning trees or neural networks where the degrees of freedom are allowed to grow in sample size. The Super Learner presented here is built upon a V-fold cross validation scheme.

Consider a library of candidate estimators, $\hat{\bar{Q}}_{Y(t),1},\ldots,\hat{\bar{Q}}_{Y(t),K}$, as mappings from the covariate history and time to the interval $[0,1]$. For the k^{th} candidate estimator, let $\hat{\bar{Q}}_{Y(t),n,k} = \hat{\bar{Q}}_{Y(t),k}(P_n)$ denote the mapping induced by the empirical distribution of our sample. We can define the V-fold cross validated risk by first starting with a random vector $B \in \{1,\ldots,V\}^n$ with a multinomial distribution where each value has probability $\frac{1}{V}$. For the i^{th} subject

$$I(B_i = b) = \begin{cases} 0 & \text{if the i}^{th}\text{ observation is in the } b^{th} \text{ training set} \\ 1 & \text{if the i}^{th}\text{ observation is in the } b^{th} \text{ validation set} \end{cases}.$$

Let $\hat{\bar{Q}}_{Y(t),n,b,k} = \hat{\bar{Q}}_{Y(t),k}(P_{n,B\neq b})$ be the k^{th} candidate estimator induced by the b^{th} training set. An estimate of the V-fold cross validated risk can be computed

$$R^{CV}_{n,V}(\bar{Q}_{Y(t)}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\tau}\sum_{b=1}^{V} I(B_i = b)\mathscr{L}_t(\bar{Q}_{Y(t),n,b})(O_i(t))$$

The regular V-fold cross validation selector $\hat{k}_n = \underset{k\in\{1,\ldots,K\}}{\operatorname{argmin}} R^{CV}_{n,V}(\hat{\bar{Q}}_{Y(t),k})$ selects the single candidate estimator with the lowest cross validated risk. The cross validation selected estimator is

$\hat{\bar{Q}}_{Y(t),n,\hat{k}_n}$, i.e., the cross validation selected estimator induced on the entire n-sized sample. van der Laan and Dudoit (2003) proved that, for bounded loss functions, the cross validation selector is asymptotically equivalent with the "oracle" in terms of the risk difference dissimilarity with the the true parameter. This is given by the following oracle inequality

$$E[d(\bar{Q}_{Y,0},\hat{\bar{Q}}_{Y,n,\hat{k}_n})] \leq (1+2\lambda)E[d(\bar{Q}_{Y,0},\hat{\bar{Q}}_{Y,n,\tilde{k}_n})]+2C(\lambda)\frac{1+log(K(n))}{np}$$

where $K(n)$ is the size or cardinality of the library of candidate estimators for the training sample of size n. $\hat{k}_n$ is the cross validation selector, and $\tilde{k}_n$ is the oracle selector for the empirical distribution of our n observations. λ is a positive constant and $C(\lambda)$ is a constant defined by van der Laan and Dudoit (2003). It is helpful to understand the variability in $R^{CV}_{n,V}$ when using cross validation for estimator selection. Because $R^{CV}_{n,V}$ is an asymptotically linear estimator, an estimate of its standard error can be constructed based on its influence curve

$$D = \sum_{b=1}^{V} I(B_i = b)\Big(\sum_{t=0}^{\tau} \mathscr{L}_t(\bar{Q}_{Y(t),n,b})(O_i(t)) - R^{CV}_{n,V}\Big)$$

The standard error of the cross validated risk estimate is the square root of the variance of its influence curve divided by $\sqrt{n}$, and can be estimated with

$$\hat{SE}(R^{CV}_{n,V}) = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\sum_{b=1}^{V} I(B_i = b)\Big(\sum_{t=0}^{\tau} \mathscr{L}_t(\bar{Q}_{Y(t),n,b})(O_i(t)) - R^{CV}_{n,V}\Big)^2}$$

2.5 Super Learner

The Super Learner is an estimator based on a generalization of the oracle result for cross validation. Instead of the usual cross validation selector, van der Laan et al. (2007) extended the library to include new candidates defined as weighted combinations of the individual candidates. The original article proposed new candidates of a particular parametric form

$$\bar{Q}_{Y(t),n,\alpha} = m(\hat{\bar{Q}}_{Y(t),n,1},\dots,\hat{\bar{Q}}_{Y(t),n,K}|\alpha)$$

where $\alpha = (\alpha_1,\dots,\alpha_K), \sum_{k=1}^{K}\alpha_k = 1, \alpha_k \geq 0 : k = 1,\dots,K$. The constraints on α bound the extended library by the convex hull of the individual candidates in the original library. This convexity helps to ensure bounds on the risk, a necessary condition for the cross validation oracle result, and the framework has since proven to work well in practice. The convex Super Learner for the conditional intensity is then defined as the weighted combination that minimizes the V-fold cross validated risk, defined as follows.

$$\bar{Q}_{Y(t),n,SL} = m\big(\hat{\bar{Q}}_{Y(t),n,1},\dots,\hat{\bar{Q}}_{Y(t),n,K}|\alpha_n\big)$$

where

$$\alpha_n = \underset{\alpha}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{\tau} \sum_{b=1}^{V} I(B_i = b) \mathscr{L}_t \big(m(\hat{\hat{Q}}_{Y(t),n,b,1}, \ldots, \hat{\hat{Q}}_{Y(t),n,b,K} | \alpha) \big) \big(O_i(t) \big)$$

This can be simplified by defining $Z_i(t) = \big(\sum_{b=1}^{V} I(B_i = b) \hat{\hat{Q}}_{Y(t),n,b,k}(O_i(t)) : k = 1, \ldots, K \big)$ to be the vector of cross validated candidate predictions for the i^{th} subject at time point t. Doing so for all subjects yields a matrix Z with K columns and a number of rows equal to $\sum_{i=1}^{n} \sum_{t=0}^{\tau} R_i(t)$. That is, $Z_{i,k}(t)$ is the cross validated estimate from the k^{th} candidate estimator for the i^{th} subject at time-point t. With this setup, the Super Learner weights can be restated as

$$\alpha_n = \underset{\alpha}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{\tau} \mathscr{L}_t \big(m(Z_i(t) | \alpha) \big) \big(O_i(t) \big)$$

The Super Learner to minimize the negative Bernoulli loglikelihood risk uses

$$m\big(Z_i(t) | \alpha\big) = \big(1 + e^{-\sum_{j=1}^{K} logit(Z_{i,k}(t)) \alpha_{n,k}} \big)^{-1}$$

where $logit(x) = log(\frac{x}{1-x})$. The loss function is

$$\sum_t \mathscr{L}_t \big(m(Z(t) | \alpha) \big) \big(O(t) \big) = \sum_t -2 \Bigg((1 - Y(t)) \sum_k logit(Z_k(t)) \alpha_k + log(1 + e^{-\sum_k logit(Z_k(t)) \alpha_k}) \Bigg)$$

The weights can then be estimated with nonlinear optimization software, and the convex Super Learner is then defined $\hat{\hat{Q}}_{Y(t),n,SL} = \Big(1 + e^{-\sum_k logit(\hat{\hat{Q}}_{Y(t),n,k}) \alpha_{n,k}} \Big)^{-1}$. The Super Learner to minimize the squared error risk uses

$$m\big(Z_i(t) | \alpha\big) = \sum_{j=1}^{K} Z_{i,k}(t) \alpha_{n,k}$$

and the loss function

$$\sum_t \mathscr{L}_t \big(m(Z(t) | \alpha) \big) \big(O(t) \big) = \sum_t \big(Y(t) - \sum_k Z_k(t) \alpha_k \big)^2$$

The weights can again then be estimated with standard optimization software, and the convex Super Learner is then defined $\hat{\hat{Q}}_{Y(t),n,SL} = \sum_k \hat{\hat{Q}}_{Y(t),n,k} \alpha_{n,k}$.

2.6 Application: Conditional stroke intensity in ATRIA-1

Warfarin has been shown in clinical trials to be a prophylactic therapy that reduces the incidence of thromboembolic stroke in persons with a diagnosis of atrial fibrillation. The physiological mechanism involves interference with normal clotting factors in the blood. As a result, a major adverse side effect of warfarin therapy is uncontrolled bleeding events. Thus in persons who do take warfarin, seemingly minor falls or bumps to the head may result in severe brain injuries whose neurological consequences may be as bad or worse than what would have happened if the warfarin were never prescribed. The clinical decision of whether or not to prescribe warfarin is there complex and should take into account several factors.

Often the single most important factor in this decision is the individual patient's underlying propensity of thromboembolic stroke. Certainly the increased risk of adverse side effects should not be prescribed to patients who have a very low propensity of thromboembolic stroke. Conversely, the identification of high risk patients may lead to prevention through intervention with warfarin. Until relatively recently, an individual patient's underlying propensity for thromboembolic stroke was a purely subjective assessment made by an attending clinician. In 2000, however, researchers identified clinical prognostic factors and organized them into the 0-6 point $CHADS_2$ score. Stroke propensities for persons with a particular $CHADS_2$ score at baseline were then estimated empirically. This score is now widely used to estimate stroke propensity to aid in warfarin prescription decisions.

One of the major research objectives of Kaiser Permanente's ATRIA-1 study was to construct a state-of-the art stroke propensity assessment function using statistical learning methods applied to the longitudinal data in the medical histories of patients who were not taking warfarin. We therefore set about the estimation of the conditional intensity parameter with a Super Learner.

2.6.1 Observed data

ATRIA-1 included 13,559 individuals followed during years 1997-2003. The data set has a longitudinal right-censored survival data structure with both fixed and time-dependent covariates. The outcome of interest, thromoboembolic stroke, may be represented as a counting process. Right-censoring was present due in part to disenrollment from the Kaiser health care system, but mostly from administrative censoring at the end of the study. Fixed covariates included gender, race, income, and education. the time-dependent covariates included updated diagnoses of comorbidities taken from the electronic medical records, common biomarkers values taken from laboratory logs, and an indicator of warfarin prescription usage inferred from pharmacy databanks. The lowest latency between covariate and outcome updates was a single day.

The appropriate data representation is therefore $O = (R(t), R(t)\mathscr{F}_t, R(t)Y(t) : t = 1, \ldots, \tau) = (O(t) : t = 1, \ldots, \tau)$, where $Y(t)$ is the indicator that a person experienced the stroke event just after day t. $\mathscr{F}_t$ is the medical history which includes age, gender, race, education, income, diagnoses of various comorbidities including history of stroke at baseline, diabetes mellitus, heart failure, coronary artery disease, bleeding events, falls, dementia, seizures, hypertension, etc., most recent lab values for total hemoglobin, HgbA1C, total white blood cells, serum creatinine, estimated glomerular filtration rate, and proteinuria. $R(t)$ is the indicator that the person is in the "risk set" at time t. Persons who experienced the terminal stroke event or were censored before time t necessarily have $R(t) = 0$. Also, because we were only interested in the conditional stroke intensity for persons who were not currently on warfarin medication, $R(t)$ was set to 0 during time periods for which a person was taking warfarin medication. Our working dataset, comprised of all observation for which $R(t) = 1$, for the conditional intensity estimation problem involved a large data set with 40 covariates and an extremely sparse outcome vector that contained roughly 11.9 person-day observations with $Y(t) = 0$ and 687 with $Y(t) = 1$. This working data set amounted to just over 500MB as a compressed SAS data set in SAS v9.2.

2.6.2 Library of candidate estimators

We constructed a library of 21 candidate estimators that spanned a relatively broad class of functions, including the unconditional mean (PROC MEANS), several logistic regressions (PROC LOGISTIC), linear discriminant analyses (PROC DISCRIM), 3-layer perceptrons with 2 hidden units (PROC NEURAL), recursive partitioning decision trees (PROC ARBORETUM), and gradient boosting (PROC TREEBOOST). These are outlined below:

MEAN The unconditional mean of the outcome vector.

LOGIT Logistic regression using additive main terms

LOGIT_FOR Logistic regression using additive main terms selected via forward inclusion

LOGIT_BACK Logistic regression using additive main terms selected via backward elimination

LOGIT_STEP Logistic regression using additive main terms selected in a stepwise procedure

LOGIT_LASSO Logistic regression using additive main terms selected via squared error LASSO

LOGIT_TREE Logistic regression using additive main terms selected via decision tree importance

LOGIT_UNIP Logistic regression using additive main terms selected via univariate logistic p-value

LOGIT_LASSOX Logistic regression using additive main terms and 2-ways interactions selected via squared error LASSO

LOGIT_XP Logistic regression using additive main terms and 2-ways interactions selected via bivariate logistic p-value

LDA Linear discriminant analysis using additive main terms

LDA_LASSO Linear discriminant analysis using additive main terms selected via squared error LASSO

LDA_TREE Linear discriminant analysis using additive main terms selected via decision tree importance

LDA_UNIP Linear discriminant analysis using additive main terms selected via univariate logistic p-value

NN2 Artificial neural network 3-layer perceptron with 2 hidden units using additive main terms

NN2_LASSO Artificial neural network 3-layer perceptron with 2 hidden units using additive main terms selected via squared error LASSO

NN2_TREE Artificial neural network 3-layer perceptron with 2 hidden units using additive main terms selected via decision tree importance

NN2_UNIP Artificial neural network 3-layer perceptron with 2 hidden units using additive main terms selected via univariate logistic p-value

TREE Recursive partitioning decision tree using additive main terms

FOREST10 Mean of 10 bootstrapped recursive partitioning decision trees using additive main terms

BOOST50 Gradient boosting machine (treeboost) with 50 iterations

2.6.3 Cross validation, risk, and convex weights

The Super Learner was constructed using a 10-fold cross validation scheme. The individual patients were randomly assigned to one of 10 folds, and all of the t-specific person-day rows for the individual were used in the corresponding training and validation sets. We computed two Super Learners: one for the risk defined as the expectation of the sum loss over all t-specific negative log likelihood loss functions and the other for the risk defined as the expectation of the sum loss over all t-specific squared error loss functions. Estimation of the Super Learner convex weights themselves was carried out with the PROC NLP constrained estimation procedures in SAS/OR, where the usual cross validation selector was taken as the starting value for the weights in the optimization. A full description of the SAS software implementation of the Super Learner is given in the appendix of this dissertation. Standard errors for the 10-fold cross validated risks were computed based on the influence curve.

2.6.4 Results

The Super Learner for the negative Bernoulli loglikelihood risk had a 10-fold cross validated risk of 1.309, roughly equivalent to that of the two single best candidate estimators, logistic regression with variable selection based on univariate p-value < 0.05 and logistic regression with variables selected via a stepwise procedure. The convex Super Learner weights were split nearly evenly between these two candidates (34% and 32%, respectively) and and the gradient boosting machine (28%), with about 6% split between three other candidates. The remaining 15 candidates

Table 2.1: Super Learner for stroke intensity, 10-fold cross validated (CV) loglikelihood risk

Estimator	α_n	CV risk (SE)
Super Learner for loglikelihood	—	1.3090 (0.0439)
Super Learner for squared error	—	1.3150 (0.0437)
Unconditional mean (null model)	0.02	1.3538 (0.0453)
Logistic regression, all variables	0.00	1.3107 (0.0439)
Logistic regression, forward selection	0.00	1.3119 (0.0440)
Logistic regression, backward selection	0.00	1.3096 (0.0439)
Logistic regression, stepwise selection	0.32	1.3089 (0.0439)
Logistic regression, lasso selection	0.00	1.3140 (0.0440)
Logistic regression, decision tree selection	0.00	1.3142 (0.0440)
Logistic regression, univariate $p < 0.05$	0.34	1.3089 (0.0439)
Logistic regression, lasso selection on all 2-way interactions	0.00	1.3180 (0.0441)
Logistic regression, 2-way interaction $p < 0.05$	0.01	1.3132 (0.0440)
Linear discriminant analysis, all variables	0.00	1.3329 (0.0441)
Linear discriminant analysis, lasso selection	0.00	1.3310 (0.0442)
Linear discriminant analysis, decision tree selection	0.00	1.3292 (0.0441)
Linear discriminant analysis, univariate $p < 0.05$	0.00	1.3272 (0.0440)
Three-layer perceptron 2 hidden units, all variables	0.03	1.3593 (0.0442)
Three-layer perceptron 2 hidden units, lasso selection	0.00	1.3945 (0.0474)
Three-layer perceptron 2 hidden units, decision tree selection	0.00	1.3301 (0.0442)
Three-layer perceptron 2 hidden units, univariate $p < 0.05$	0.00	1.3414 (0.0440)
Decision tree, all variables	0.00	1.3298 (0.0443)
Mean of 10 bootstrapped decision trees, , all variables	0.00	1.3291 (0.0443)
Gradient boosting machine (treeboost), all variables	0.28	1.3133 (0.0438)

recieved no weight in the final Super Learner estimates.

The Super Learner for the squared error risk had a 10-fold cross validated risk of 0.0062906, which was worse than several of the candidates. The convex weights were again given to the best performing candidates logistic regression with variable selection based on univariate p-value < 0.05 (28%) and logistic regression with variables selected via a stepwise procedure (18%), and the gradient boosting machine (21%). A large weight (23%) was also given to the mean of the 10 decision bootstrapped decision trees, and the remaining 10% were split between three other candidates.

Interestingly, the Super Learner for the negative Bernoulli loglikelihood risk performed better than the Super Learner for the squared error risk with respect to the 10-fold cross validated squared error risk. The standard error for the difference in their 10-fold cross validated risks (estimated based on their influence curves) indicated that this difference was significant (p = 0.0005). Based

Table 2.2: Super Learner for stroke intensity, 10-fold cross validated (CV) squared error risk

Estimator	α_n	CV risk (SE)
Super Learner for loglikelihood	—	0.062900 (0.002323)
Super Learner for squared error	—	0.062906 (0.002323)
Unconditional mean (null model)	0.03	0.062903 (0.002323)
Logistic regression, all variables	0.00	0.062901 (0.002323)
Logistic regression, forward selection	0.00	0.062901 (0.002323)
Logistic regression, backward selection	0.00	0.062900 (0.002323)
Logistic regression, stepwise selection	0.18	0.062900 (0.002323)
Logistic regression, lasso selection	0.00	0.062900 (0.002323)
Logistic regression, decision tree selection	0.00	0.062901 (0.002323)
Logistic regression, univariate $p < 0.05$	0.28	0.062900 (0.002323)
Logistic regression, lasso selection on all 2-way interactions	0.02	0.062901 (0.002323)
Logistic regression, 2-way interaction $p < 0.05$	0.00	0.062901 (0.002323)
Linear discriminant analysis, all variables	0.00	0.062921 (0.002323)
Linear discriminant analysis, lasso selection	0.00	0.062913 (0.002323)
Linear discriminant analysis, decision tree selection	0.00	0.062909 (0.002323)
Linear discriminant analysis, univariate $p < 0.05$	0.00	0.062915 (0.002323)
Three-layer perceptron 2 hidden units, all variables	0.05	0.062909 (0.002323)
Three-layer perceptron 2 hidden units, lasso selection	0.00	0.063566 (0.002330)
Three-layer perceptron 2 hidden units, decision tree selection	0.00	0.062903 (0.002323)
Three-layer perceptron 2 hidden units, univariate $p < 0.05$	0.00	0.062906 (0.002323)
Decision tree, all variables	0.00	0.062901 (0.002323)
Mean of 10 bootstrapped decision trees, , all variables	0.23	0.062901 (0.002323)
Gradient boosting machine (treeboost), all variables	0.21	0.062901 (0.002323)

on this, we chose to work with the Super Learner for the negative Bernoulli log likelihood risk as our primary estimate of the conditional intensity for stroke in the ATRIA-1 data.

2.7 Is the negative loglikelihood a better loss function?

The results of the previous section motivated a further investigation of the properties of the Super Learners for squared error and loglikelihood risks. Recall that the Super Learner for log-likelihood risk achieved a 10-fold cross validated risk that was essentially equivalent with that of the best single candidate in the library. The Super Learner for squared error risk, on the other hand, achieved a 10-fold cross validated squared error risk that was beaten by the vast majority of candidates in the library. Further, the Super Learner for loglikelihood risk performed significantly better than the Super Learner for squared error risk, even with respect to the squared error risk assessment. This empirical result suggested to us that the Super Learner for loglikelihood risk was approaching its oracle faster than the Super Learner for squared error risk was approaching its oracle. On the assumption that these two oracles are roughly equivalent, it further suggests that the negative log loss is simply a better loss function for this particular parameter. In this section, we investigate analytically the properties of both squared error and loglikelihood loss in the prediction of rare binary outcomes in a point treatment setting.

Let $O = (Y,W) \sim P_0 \in \mathscr{M}$, and let $\bar{Q}_{Y,0} = \bar{Q}_Y(P_0) = E_0[Y|W]$. Let $\mathscr{L}_{KL}(\bar{Q}_Y) = -\big(Ylog(\bar{Q}_Y) + (1-Y)log(1-\bar{Q}_Y)\big)$, and note that $\bar{Q}_{Y,0} = \underset{\bar{Q}_Y}{\text{argmin}}\, E_0[\mathscr{L}_{KL}(\bar{Q}_Y)]$. Recall the oracle inequality (van der Laan and Dudoit, 2003)

$$E[d(\bar{Q}_{Y,0}, \hat{\bar{Q}}_{Y,n,\hat{k}_n})] \leq (1+2\delta)E[d(\bar{Q}_{Y,0}, \hat{\bar{Q}}_{Y,n,\tilde{k}_n})] + 2C(\delta)\frac{1+log(K(n))}{np}$$

where C actually depends on two constants

$$C(\delta) = C(M_1, M_2, \delta) = 2(1+\delta)^2\Big(\frac{M_1}{3} + \frac{M_2}{\delta}\Big)$$

Define $\mathscr{L}^*_{KL}(\bar{Q}_Y, \bar{Q}_{Y,0}) = \mathscr{L}_{KL}(\bar{Q}_Y) - \mathscr{L}_{KL}(\bar{Q}_{Y,0})$ to be the centered log-likelihood loss function. $M_1 = \sup_{\bar{Q}_Y} \sup_o \mid \mathscr{L}^*_{KL}(\bar{Q}_Y, \bar{Q}_{Y,0})(o) \mid$ and $M_2 = \sup_{\bar{Q}_Y} E_0[\mathscr{L}^*_{KL}(\bar{Q}_Y, \bar{Q}_{Y,0})^2]/E_0[\mathscr{L}^*_{KL}(\bar{Q}_Y, \bar{Q}_{Y,0})]$.

Suppose that we know that $\delta < \bar{Q}_Y < M$ for all $\bar{Q}_Y$ in the parameter space. Upper bounds have been provided in a previous article, but we suspect they are too conservative for practical use. We can bound $\mathscr{L}^*_{KL}(\bar{Q}_Y, \bar{Q}_{Y,0}) \leq \log M/\delta$, which is a bound that can be achieved by selecting $\bar{Q}_Y$ as a density that equals δ at a point at which $\bar{Q}_{Y,0}$ equals M. In a simulation study, one can determine an upper bound for M_1 w.r.t. the library of estimators, which might result in a much smaller M_1 if none of the candidate estimators will get close to such a least favorable choice. In other words, one determines an upper bound of $\max_k \sup_o \mid \mathscr{L}^*_{KL}(\hat{\bar{Q}}_{Y,k(P_n)}, \bar{Q}_{Y,0}) \mid$. Regarding M_2, the upper bound M/δ as presented earlier can be very large, and it is unclear if this upper bound cannot be much improved. Again, via simulation one could determine an upper bound for M_2 w.r.t. the library of candidate estimators.

The behavior of the bounds can be studied analytically in the special case where all candidate estimators, e.g., neural networks, k-nearest neighbors, recursive partitioning decision trees, etc., are uniformly consistent for $\bar{Q}_{Y,0}$. For that purpose, we consider the bounds when $\sqrt{d(\hat{\bar{Q}}_{Y,k(P_n)}, \bar{Q}_{Y,0})} \leq \varepsilon$ and its limit when $\varepsilon \to 0$. We suggest that we might use these asymptotic upper bounds for M_1 and M_2 to describe the asymptotic behavior of the finite sample oracle inequality.

Let $\bar{Q}^h_{Y,0}(\varepsilon) = \frac{1}{1+e^{-(logit(\bar{Q}_{Y,0})+\varepsilon h)}}$ denote a one dimensional fluctuation of density implied by $\bar{Q}_{Y,0}$ indexed by a score h that satisfies $E_0[h(O)|W] = 0$. This often called the "logistic" fluctuation and guarantees that the fluctuated densities remain in the model for P_0.

We wish to determine

$$\sup_h \lim_{\varepsilon \to 0} \frac{E_0[\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon), \bar{Q}_{Y,0})^2]}{E_0[\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon), \bar{Q}_{Y,0})]},$$

where the supremum is over all one-dimensional fluctuations of $\bar{Q}_{Y,0}$ allowed by parameter space/model.

The numerator and denominator equal zero at $\varepsilon = 0$. In addition, even the first derivative of numerator and denominator at $\varepsilon = 0$ both equal zero. Therefore, the limit w.r.t. $\varepsilon \to 0$ is given by the ratio of the second derivatives of the numerator and denominator w.r.t. ε at $\varepsilon = 0$.

Note that

$$\frac{\partial}{\partial \varepsilon}\bar{Q}^h_{Y,0}(\varepsilon) = h\bar{Q}^h_{Y,0}(\varepsilon)(1-\bar{Q}^h_{Y,0}(\varepsilon))$$

$$\frac{\partial}{\partial \varepsilon}\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0}) = h(\bar{Q}^h_{Y,0}(\varepsilon)-Y)$$

$$\frac{\partial^2}{\partial \varepsilon^2}\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0}) = h^2\bar{Q}^h_{Y,0}(\varepsilon)(1-\bar{Q}^h_{Y,0}(\varepsilon))$$

The second derivative of numerator is given by

$$\begin{aligned}
&\frac{\partial^2}{\partial \varepsilon^2}E_0\left[\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0})^2\right]|_{\varepsilon=0}\\
&\quad= 2E_0\left[\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0})\left(\frac{\partial^2}{\partial \varepsilon^2}\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0})\right) + \left(\frac{\partial}{\partial \varepsilon}\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0})\right)^2|_{\varepsilon=0}\right]\\
&\quad= 2E_0\left[\left(\frac{\partial}{\partial \varepsilon}\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0})\right)^2|_{\varepsilon=0}\right]\\
&\quad= 2E_0\left[h^2(\bar{Q}_{Y,0}-Y)^2\right]
\end{aligned}$$

The second derivative of the denominator is given by

$$\begin{aligned}
\frac{\partial^2}{\partial \varepsilon^2}E_0\left[\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0})\right] &= E_0\left[\frac{\partial}{\partial \varepsilon}h(\bar{Q}^h_{Y,0}(\varepsilon)-Y)|_{\varepsilon=0}\right]\\
&= E_0\left[h^2\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0})\right]
\end{aligned}$$

And the ratio is

$$\frac{2E_0\left[h^2(\bar{Q}_{Y,0}-Y)^2\right]}{E_0\left[h^2\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0})\right]}$$

By conditioning on W, it becomes clear that the expectations in the numerator and denominator are in fact equal since $E_0\left[(\bar{Q}_{Y,0}-Y)^2|W\right] = \bar{Q}_{Y,0}(1-\bar{Q}_{Y,0})$. Thus we obtain

$$\lim_{\varepsilon\to 0}\frac{E_0[\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0})^2]}{E_0[\mathscr{L}^*_{KL}(\bar{Q}_{Y,0}(\varepsilon),\bar{Q}_{Y,0})]} = 2,$$

and also the supremum over h equals 2. We suggest using $M_2 = 2$ in the oracle inequality to describe behavior of the cross-validation selector for a library of uniformly consistent estimators. We also suggest that this proof can be refined to establish:

$$\lim_{\varepsilon\to 0}\sup_h \frac{E_0[\mathscr{L}^*_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0})^2]}{E_0[\mathscr{L}^*_{KL}(\bar{Q}_{Y,0}(\varepsilon),\bar{Q}_{Y,0})]} = 2.$$

This would then also establish that

$$\lim_{\varepsilon\to 0}\sup_{\bar{Q}_Y:\sqrt{d_0(\bar{Q}_Y,\bar{Q}_{Y,0})}<\varepsilon} \frac{E_0[\mathscr{L}^*_{KL}(\bar{Q}_Y,\bar{Q}_{Y,0})^2]}{E_0[\mathscr{L}^*_{KL}(\bar{Q}_Y,\bar{Q}_{Y,0})]} = 2,$$

where $d_0(\bar{Q}_Y,\bar{Q}_{Y,0}) = E_0[\mathscr{L}^*_{KL}(\bar{Q}_Y,\bar{Q}_{Y,0})]$ is the Kullback-Leibler dissimilarity (whose square root corresponds with an L^2-norm for densities). We also note that $M_1(\varepsilon) = O(\varepsilon)$, so that we can select $M_1 = 0$ to represent the asymptotic behavior of the constant M_1 in oracle inequality for a library of uniformly consistent estimators.

Now consider the squared error loss, $\mathscr{L}_{L2}(\bar{Q}_Y)(O) = (Y-\bar{Q}_Y)^2$, and note that our parameter can also be identified as $\bar{Q}_{Y,0} = \underset{\bar{Q}_Y}{\operatorname{argmin}}\, E_0[\mathscr{L}_{L2}(\bar{Q}_Y)]$. We define $\mathscr{L}^*_{L2}(\bar{Q}_Y,\bar{Q}_{Y,0}) = \mathscr{L}_{L2}(\bar{Q}_Y) - \mathscr{L}_{L2}(\bar{Q}_{Y,0})$ to be the centered squared error loss function. For this loss function it can be shown that

$$\lim_{\varepsilon\to 0}\frac{E_0[\mathscr{L}^*_{L2}(\bar{Q}_{Y,0}(\varepsilon),\bar{Q}_{Y,0})^2]}{E_0[\mathscr{L}^*_{L2}(\bar{Q}_{Y,0}(\varepsilon),\bar{Q}_{Y,0})]} = 4\sup_w \sigma^2_0(w)$$

where $\sigma^2_0 = E_0[((Y-\bar{Q}_{Y,0}(W))^2 \mid W)] = \bar{Q}_{Y,0}(1-\bar{Q}_{Y,0})$. Note that a sharp upper bound for $\sigma^2_0(W)$ is given by $\frac{1}{4}$. Thus, $M_2 = 1$ for the squared error loss and, again, $M_1(\varepsilon) = O(\varepsilon)$ so that we can select $M_1 = 0$.

Suppose that the oracle selected estimator for the squared error loss and log-likelihood loss are identical. In that case, we need to compare

$$C(0,1,\delta)\frac{\log(1+K(n))}{npE_0E_{B_n}d_{L2}(\hat{\bar{Q}}_{Y,\tilde{k}}(P_{n,B\neq b}),\bar{Q}_{Y,0})} \text{ and } C(0,2,\delta)\frac{\log(1+K(n))}{npE_0E_{B_n}d_{KL}(\hat{\bar{Q}}_{Y,\tilde{k}}(P_{n,B\neq b}),\bar{Q}_{Y,0})}.$$

Using a second order Taylor expansion we can compute

$$\begin{aligned} d_{KL}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0}) &= E_0[h^2\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0})]\frac{\varepsilon^2}{2} \\ d_{L2}(\bar{Q}^h_{Y,0}(\varepsilon),\bar{Q}_{Y,0}) &= E_0[h^2(\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0}))^2]\varepsilon^2. \end{aligned}$$

Thus, the two second order terms for the squared error and log-likelihood oracle inequality, representing the oracle selected estimator as $\bar{Q}^h_{Y,0}(\varepsilon)$ for some choice of h and ε, are given by:

$$C(0,1,\delta)\frac{\log(1+K(n))}{npE_0[h^2(\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0}))^2]\varepsilon^2} \text{ and } C(0,2,\delta)\frac{\log(1+K(n))}{npE_0[h^2\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0})]\frac{\varepsilon^2}{2}}.$$

Note $C(0,1,\delta)/C(0,2,\delta)=\frac{1}{2}$.

Thus, the relative size of the squared error and log-likelihood second order term is given by:

$$\frac{1}{4}\frac{E_0[h^2\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0})]}{E_0[h^2(\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0}))^2]}\geq 1.$$

For rare outcomes, we have that $\bar{Q}_{Y,0}(1-\bar{Q}_{Y,0})\approx\bar{Q}_{Y,0}$ so that this ratio could be very large. For example, if $\bar{Q}_{Y,0}$ is bounded from below by δ, could behave as $\frac{1}{4\delta(1-\delta)}$. Thus, we conclude that the behavior of the second order term for estimators that approximate $\bar{Q}_{Y,0}$ is much worse for the squared error loss than for the log-likelihood loss. This demonstrates that the log-likelihood cross-validation selector will have much better behavior relative to its oracle selected estimator than the squared error cross-validation selector relative to its oracle selected estimator, even asymptotically when all candidate estimators approximate the truth.

The empirical results in the previous section suggest that the loglikelihood may also be better than the squared error loss for estimation of the conditional intensity of a counting process. Indeed, for right-censored survival data, $Y(t)$ the indicator that the event counting process jumps just after time t is typically rare. However, the analytic results discussed here do not immediately apply because the loss functions for the conditional intensity involve a sum over all t of the t-specific losses. Analytic comparisons of loss functions for this parameter is an area of future research.

2.8 Discussion

This chapter presented the Super Learner for the conditional intensity of a counting process in a longitudinal time-dependent data structure. This is a particular example of the general Super Learner estimation template for nonpathwise differentiable parameters in a nonparametric statistical model. A practical implementation of the Super Learner was developed in the SAS software platform and a Super Learner for the conditional stroke intensity in ATRIA-1 was constructed to serve as a guide in the clinical decision of whether or not to prescribed warfarin in persons with atrial fibrillation. The Super Learner for loglikelihood risk performed as well as the best estimators in its library with respect to 10-fold cross validated risk, but the Super Learner for the squared error risk did not. We showed analytically that in the prediction of rare events in a point treatment setting, such behavior can be explained by the remainder of the finite sample oracle inequality for the cross validation selector. Extending the analytic results to other parameters, including the conditional intensity of a time-depednent counting process is an area for future research.

Though the primary motivation for the Super Learner presented here was a prediction-type task, the Super Learner for the conditional intensity may also be used as a component in the estimation of intervention-specific means and under certain assumptions causal effects in time-dependent data structures. An example of this is presented in Chapter 4, where two Super Learners are fit - one for the conditional intensity of an event process and a second one for the conditional intensity of a censoring process - as components in the estimation of the casual effect of warfarin on the probability of stroke of death within a 1-year time frame.

While the parameter identification and Super Learner procedure are relatively straightforward examples of the general templates, the practical implementation in ATRIA-1 proved difficult for several reasons. The sheer size of the data set was a challenge on its own, that precluded out-of-the-box use of existing Super Learner software as implemented in the R language package *SuperLearner*. Initially, we worked with smaller version of the data set via case-control type sampling and used a Super Learner with an inverse probability of sampling weighted risk. Though this approach will also result in a Super Learner that is asymptotically equivalent with the ”oracle”, we found that the additional sampling error increased the cross validated risk estimates (as computed on the full prospective data) beyond any gains from using a broader library. We therefore chose to develop a scalable Super Learner on the SAS platform that is not bound by memory limitations in processing big data. This software development project comprised a major component of the research presented here. The concept and design is discussed in the the appendix here and the full SAS code for the %SUPERLEARNER SAS macro as well as that for several of the candidate estimators presented in this chapter are reproduced.

Chapter 3

Targeted maximum likelihood estimation for prediction calibration

3.1 Introduction

A reasonable and often used statistical prediction function parameter is the conditional expectation of the outcome given the covariates. In a nonparametric statistical model, the conditional expectation function parameter can be identified as the minimizer of the risk, i.e., the expectation of a loss function under the true probability distribution of the data (van der Laan and Dudoit, 2003). Valid loss functions for this parameter include the squared error loss, negative loglikelihood loss, and several others. Typically, the form of this conditional expectation function is unknown and must be learned or estimated from observational data. When working within a nonparametric statistical model a reasonable approach is to consider several classes of potentially flexible estimators and then select one that minimizes an unbiased estimate of the risk. This risk minimization involves trade offs with respect to certain statistical properties. For example, mean squared error risk can be decomposed into a variance term and a term for the squared bias, and its minimization is thus viewed as a global bias-variance trade off. With most regular estimators the balance of this trade off is determined by the complexity; those with higher complexity have less bias and more variance and vice versa. Each estimator may therefore be tuned to achieve the best global bias-variance trade off corresponding to optimal prediction error.

Suppose, however, that particular subgroups of the data are of particular interest. The global risk minimization as described above, will generally result in bias for the conditional expectation of the outcome for the specified subgroups in return for better prediction error over the entire data distribution. One approach for unbiased estimation is to simply take the empirical mean of the outcome for each subgroup. Empirical means are unbiased and efficient nonparametric maximum likelihood estimators for subgroup-specific conditional expectations, though they are unlikely to perform well as an estimator of the prediction function. This motivates the construction of estimators of the prediction function that, in addition to performing well in terms of risk, also maps into the empirical means for the subgroups of interest. Certainly for non-technical and technical researchers alike, it is often reassuring to see that the empirical mean of the predictions is equal to observed empirical mean of the outcome in sample data. In other words, the expected matches the observed. This is property has been called *calibration* (Harrell Jr., Lee, and Mark, 1996), and is often taken into consideration in the assessment of predictions.

Several methods have been suggested to assess calibration defined in this way (Hosmer, Hosmer, Cessie, and Lemeshow, 1997; Tsiatis, 1980). The Hosmer and Lemeshow goodness-of-fit test, for example, partitions observations in a sample by the deciles of the prediction space and then compares the observed outcomes within each decile with the expected outcomes under the estimated model. Tsiatis' suggested test is largely similar except that the partitioning is constructed in the covariate space, rather than the prediction space. In each test, the null hypothesis is that the conditional expectation of the estimator's prediction is unbiased for the conditional expectation of the outcome given the partitioning. Moving beyond the assessment of calibration, the literature also discusses methods for imparting the calibration property both in the context of the parametric logistic regression (Steyerberg, Borsboom, van Houwelingen, Eijkemans, and Habbema, 2004; Harrell Jr., Lee, and Mark, 1996) and also in data-adaptive machine learning (Vinterbo and Ohno-Machado, 1999).

In the present article we propose a targeted maximum likelihood estimator (TMLE) van der Laan and Rose (2011) for the calibration of a prediction function estimator. Specifically, our procedure uses the TMLE to update an initial prediction function estimator such that the updated predictions map into an unbiased and efficient estimator of subgroup-specific conditional expectations. This is readily accomplished because subgroup-specific conditional expectations are themselves smooth pathwise-differentiable parameters, which can be characterized as a function of the distribution of the data. The TMLE works by fluctuating an initial prediction function estimator in such a way that the updated prediction function estimator solves the efficient influence curve estimating equation for these smooth features. As we show, the subgroups may be defined via a partitioning of the data in either the covariate space or the prediction space of the estimator itself. In the first case, the TMLE converges in a single step. In the latter case the TMLE enforces an implicit constraint on the prediction function estimator itself, which requires an iterative procedure. Our work is a novel application of the TMLE in the context of prediction. Our TMLE procedure also has important consequences for the use of some standard goodness-of-fit tests often used in the context of prediction. In particular, we show that *any* initial estimator can be updated such that the test statistic for a Hosmer-and-Lemeshow or Tsiatis-type goodness-of-fit test will be exactly 0 for the data on which the estimator was fit, which implies that the calibration property is insufficient for model selection. We explore through simulation the impact of enforcing implicit calibration constraints on prediction performance defined in terms of a valid loss function.

The organization of this chapter is as follows. Section 3.2 provides a brief overview of targeted maximum likelihood estimation. Section 3.3 develops our new TMLE in the context of a single time-point prediction of an outcome given covariates. We discuss in turn two separate single step TMLEs for calibration of prediction functions estimator for both univariate and multivariate features, i.e., the conditional expectation of the outcome for single or multiple subgroups. We then show how this same approach may be incorporated into an iterative procedure to enforce implicit constraints on the prediction function estimator. Illustrative examples focus on a binary outcome, though the results trivially extend to the continuous case. Section 3.4 extends the ideas in Section 3.3 to develop a TMLE for the calibration of an estimator of the conditional intensity function under a time-dependent counting process framework. Again, we demonstrate the approach for calibration in single or multiple subgroups, and illustrate calibration for parameters defined as weighted averages of the time-specific intensities. Section 3.5 presents simulation results to investigate both (1) the impact of the TMLE on predictive performance assessed in terms of a valid loss function, and (2) bias reduction for the calibration parameter in a large independent validation data set. Section 3.6 provides results for the real-world project that motivated the development of the new TMLE methodology.

3.2 TMLE for calibration of the conditional expectation, $\bar{Q}_Y$

In this section we present the TMLE for calibration of the conditional expectation function. We begin with a definition of the data structure, a statistical model for the data, and the conditional expectation of the outcome formally defined as a parameter mapping from the statistical model to a space consisting of functions of the covariates. We then identify the calibration parameter as the conditional expectation of the outcome given a particular partitioning of the data. This is the parameter targeted by the TMLE for calibration. The TMLE starts with an initial estimator of the conditional expectation of the outcome as a function of the covariates, and then uses a targeted updating step to remove bias for the second parameter. The resulting TMLE maps our initial estimator into the set of empirical means within each data partition, which is indeed an unbiased and efficient estimator of the conditional expectation of the outcome given the partitioning. For brevity we only discuss the TMLE procedure along with the properties of the resulting estimator, reserving details on the derivation of influence curves for the Appendix. For the sake of presentation, we will use a working assumption that all random variables are discrete, with the understanding that all results may be generalized to case involving continuous random variables and their densities by defining an appropriate dominating measure.

3.2.1 Data, model, and conditional expectation parameter

Suppose we observe n independently and identically distributed copies of a data structure given by $O = (W,Y) \sim P_0 \in \mathscr{M}$, where $W \in \mathbb{R}^d$ is a d-dimensional vector of covariates and $Y \in \{0,1\}$ is a binary outcome. $\mathscr{M}$ represents the collection of all possible probability distributions of the data, P, and the subscript "0" on P_0 denotes the single true probability distribution. The distribution P_0 may be decomposed into orthogonal components given by marginal distribution of W denoted $Q_{W,0}$ and the conditional distribution of Y given W denoted $Q_{Y,0}$.

$$P_0(O) = P_0(Y,W) = P_0(Y|W)P_0(W) = Q_{Y,0}Q_{W,0}$$

We defined the prediction function parameter as the conditional expectation of Y given W, here denoted $\bar{Q}_{Y,0}$, which is a function of W, i.e., $\bar{Q}_{Y,0} = \bar{Q}_{Y,0}(W) = E_0[Y|W]$. Consider the negative loglikelihood loss function.

$$\mathscr{L}(\bar{Q}_Y)(O) = \mathscr{L}(\bar{Q}_Y)(W,Y) = -log[\bar{Q}_Y(W)^Y(1-\bar{Q}_Y(W))^{1-Y}]$$

The true parameter can now be identified as

$$\bar{Q}_{Y,0} = \underset{\bar{Q}_Y}{\operatorname{argmin}}\, E_0[\mathscr{L}(\bar{Q}_Y)(O)]$$

where E_0 is the expectation under the true distribution P_0.

3.2.2 Initial estimator of the conditional expectation, $\bar{Q}^0_{Y,n}$

In the nonparametric statistical model, one can never be sure a priori of the optimal estimator for the prediction function $\bar{Q}_{Y,0}$. In practice, several candidate estimators of this parameter may be worth consideration. For example, we may propose several parametric estimators characterized by different 'functional forms', or we may propose several semiparametric or nonparametric estimators characterized by different search strategies and tuning parameters. We call this collection of candidate estimators a library. There are several proposed methods for combining the predictions from candidate estimators in such a library. These are known as ensemble methods in the machine learning literature and several ensembles have been shown to outperform individual candidate estimators in several practical settings.

The Super Learner is an ensemble method that assigns weights to each of the candidate estimators in the library such that the resulting weighted ensemble minimizes the cross-validated risk. The asymptotic optimality of this procedure is based on "oracle" inequality results for cross validation proven in (van der Laan, Dudoit, and Keles, 2004). In brief, the "oracle" is defined as the weighted combination of candidate estimators contained in the library that achieves the lowest true risk. In practice this can never be known with certainty because we never know the true data distribution P_0 with certainty. However, it turns out that if the none of the estimators in the library converge to the true parameter at a parametric, $\sqrt{n}$, rate then the the Super Learner is asymptotically equivalent with the "oracle" selector with respect to cross validated risk. On the other hand, if one of the candidate estimators does converge to the true parameter at a parametric rate, the Super Learner converge will converge at the near-parametric, $\frac{log(K)}{n}$, rate where K is the number of candidate estimators contained in the library (van der Laan, Polley, and Hubbard, 2007).

3.2.3 Calibration parameter $\psi_0 = \Psi(Q_{W,0}, \bar{Q}_{Y,0})$

Now suppose we would like to construct an estimate of the prediction function that has the calibration property for some particular subgroup of the population. That is, we want our estimator to map into an unbiased estimate of the conditional expectation of the outcome within the subgroup.

Start by defining $S = S(W)$ to be a real-valued summary measure of the covariates W, and let $I_{(\mathscr{A})}\{S\}$ be the indicator that S lies in the set defined by $\mathscr{A}$.

$$\psi_0 = \Psi(P_0) = \Psi(Q_{W,0}, \bar{Q}_{Y,0}) = E_{Q_{W,0}}[Y|I_{(\mathscr{A})}\{S\} = 1] = E_{Q_{W,0}}[\bar{Q}_{Y,0}|I_{(\mathscr{A})}\{S\} = 1]$$

This makes explicit that ψ_0 is a scalar computed as a mapping, $\Psi : \mathscr{M} \to \mathbb{R}$, applied to P_0. Further, the mapping can be represented two ways, the first involving the outcome Y, and the second involving the prediction function $\bar{Q}_{Y,0}$.

Our goal is now to construct a substitution TMLE, $\Psi(Q_{W,n}, \bar{Q}^*_{Y,n})$, such that $\bar{Q}^*_{Y,n}$ performs well as an estimator of $\bar{Q}_{Y,0}$, and $\Psi(Q_{W,n}, \bar{Q}^*_{Y,n})$ is unbiased and efficient for ψ_0. As discussed previously the TMLE requires an initial estimator, a choice of (valid) loss function, a parametric fluctuation submodel, and an updating step. We will discuss these in turn. But first a discussion of the nonparametric maximum likelihood estimator for ψ_0 is worthwhile.

3.2.4 The empirical estimator of ψ_0

First consider the nonparametric maximum likelihood empirical estimator of ψ_0.

$$\psi_n = \frac{\frac{1}{n}\sum_{i=1}^n Y_i I_{\mathscr{A}}\{S_i\}}{\frac{1}{n}\sum_{i=1}^n I_{\mathscr{A}}\{S_i\}}$$

This estimator is unbiased and efficient for ψ_0 in the nonparametric statistical model. It follows that its influence curve is equal to the efficient influence curve, $D^*(P_0)(O)$, for the mapping $\Psi : \mathscr{M} \to \mathbb{R}$, except that we replace the empirical distribution P_n with the true distribution P_0. For brevity, we reserved a detailed description of $D^*(P_0)(O)$ for the Appendix. Although ψ_n is unbiased and efficient for ψ_0, it is often not a particularly good estimate of $\bar{Q}_{Y,0}$. And this, in part motivates our TMLE procedure.

3.2.5 Initial estimator $\psi_n^0 = \Psi(Q_{W,n}, \bar{Q}^0_{Y,n})$

Note that our initial estimator has two components. The first corresponds to the marginal distribution of W. We choose the empirical estimator $Q_{W,n}$, which places probability mass $\frac{1}{n}$ on every observation. It turns out that this is already targeted towards the estimation of ψ_0. Thus our initial estimator takes the form

$$\psi_n^0 = \Psi(Q_{W,n}, \bar{Q}^0_{Y,n}) = \frac{\frac{1}{n}\sum_{i=1}^n \bar{Q}^0_{Y,n}(W_i) I_{\mathscr{A}}\{S_i\}}{\frac{1}{n}\sum_{i=1}^n I_{\mathscr{A}}\{S_i\}}$$

The second component, $\bar{Q}^0_{Y,n}$, is our initial estimator of the prediction function. Recall that our estimator $\bar{Q}^0_{Y,n}$ uses Super Learning to construct an optimal estimate of $\bar{Q}_{Y,0}$, but not ψ_0. Thus, our initial estimator $\Psi(Q_{W,n}, \bar{Q}^0_{Y,n})$ is likely to have at least some bias with respect to ψ_0.

3.2.6 TMLE $\psi_n^* = \Psi(Q_{W,n}, \bar{Q}_{Y,n}^*)$

Our goal now is to construct a TMLE that updates our initial estimator $\Psi(Q_{W,n}, \bar{Q}_{Y,n}^0)$ to $\Psi(Q_{W,n}, \bar{Q}_{Y,n}^*)$ such that the updated estimator is unbiased and efficient for ψ_0, the conditional expectation of the outcome Y for a particular subgroup of the data.
Start with the negative Bernoulli loglikelihood loss function

$$\mathscr{L}(\bar{Q}_Y)(O) = \mathscr{L}(\bar{Q}_Y)(W,Y) = -log[\bar{Q}_Y(W)^Y(1-\bar{Q}_Y(W))^{1-Y}]$$

Then define a fluctuation through our initial estimator, $\bar{Q}_{Y,n}^0(W)$, with a parametric submodel, indexed by the univariate parameter ε.

$$Logit(\bar{Q}_{Y,n}^0(\varepsilon)) = Logit(\bar{Q}_{Y,n}^0) + \varepsilon \frac{I_{(\mathscr{A})}\{S\}}{E_{Q_{W,0}}[I_{(\mathscr{A})}\{S\}]}$$

where $Logit(\bar{Q}_{Y,n}^0)$ serves as a fixed offset in the linear predictor and $\frac{I_{(\mathscr{A})}\{S\}}{E_{Q_{W,0}}[I_{(\mathscr{A})}\{S\}]}$ serves as a covariate. Because the $E_{Q_{W,0}}[I_{(\mathscr{A})}\{S\}]$ in the denominator of this covariate is a constant, its value can be subsumed in the estimation of ε, and we may define the submodel more simply as

$$Logit(\bar{Q}_{Y,n}^0(\varepsilon)) = Logit(\bar{Q}_{Y,n}^0) + \varepsilon H(S)$$

where $H(S) = I_{(\mathscr{A})}\{S\}$. Heuristically, H this is the direction of the fluctuation, and ε is the magnitude, which is determined with maximum likelihood estimation, or equivalently, with minimization of the empirical negative Bernoulli loglikelihood loss function.

$$\varepsilon_n = \underset{\varepsilon}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}(\bar{Q}_{Y,n}^0)(\varepsilon)(O_i)$$

This can be achieved with standard univariate logistic regression software, and converges in a single step. The targeted update is

$$\bar{Q}_{Y,n}^* = \bar{Q}_{Y,n}^0(\varepsilon_n)$$

The TMLE $\Psi(Q_{W,n}, \bar{Q}_{Y,n}^*)$ now solves the efficient influence curve estimating equation $\sum_{i=1}^n D^*(Q_{W,n}, \bar{Q}_{Y,n}^*)(O_i) = 0$ for ψ_0. This also implies that $\Psi(Q_{W,n}, \bar{Q}_{Y,n}^*)$ is unbiased and efficient for ψ_0 and is equivalent to the empirical estimator, i.e., the empirical mean of Y amongst observations for which $S(W) \in \mathscr{A}$. It is also worth noting that because the TMLE is an asymptotically linear estimator, we can estimate its variance with the empirical variance of the estimated influence curve $\frac{1}{n}\sum_{i=1}^n [D^*(Q_{W,n}, \bar{Q}_{Y,n}^*)(O_i)]^2$.

3.2.7 TMLE for calibration for several subgroups

Now say we want a prediction function estimator $\bar{Q}_{Y,n}^*(W)$ that is calibrated with respect to several subgroups of the data. In medical outcome prediction, for example, these subgroups might be defined in terms of the covariate space, e.g., men over age 65, women with a history of comorbidities, etc. Formally, we want the estimator to map into an unbiased and efficient estimator of a vector calibration parameter. Let $I_{(\mathscr{A}_j)}\{S\} = S(W) \in \mathscr{A}_j : j \in 1,..,J$ corresponding with a partitioning of the outcomes space for S. Our goal now is to construct an estimator $\bar{Q}^*_{Y,n}$ of $\bar{Q}_{Y,0}$ that maps into an unbiased estimator of the J-dimensional parameter vector

$$\Psi(P_0) = \begin{bmatrix} \psi_{0,1} \\ \vdots \\ \psi_{0,J} \end{bmatrix} = \begin{bmatrix} E_0[Y|I_{(\mathscr{A}_1)}\{S\} = 1] \\ \vdots \\ E_0[Y|I_{(\mathscr{A}_J)}\{S\} = 1] \end{bmatrix}$$

The TMLE for this is quite similar to that for a scalar parameter. Again we will take the empirical distribution and the Super Learner, $(Q_{W,n}, \bar{Q}^0_{Y,n})$, as the initial estimator and the negative Bernoulli loglikelihood as the loss function. The only difference in the procedure is that our parametric submodel is now indexed by a J-dimensional parameter vector $\varepsilon = \{\varepsilon_1, \ldots, \varepsilon_J\}$ and a multivariate covariate $H_j(S) = I_{(\mathscr{A}_j)}\{S\} : j \in 1,..,J$, such that the score of ε spans the J-dimensional vector efficient influence curve. This submodel is

$$Logit(\bar{Q}^0_{Y,n}(\varepsilon)) = Logit(\bar{Q}^0_{Y,n}) + \varepsilon_1 H_1(S) + \cdots + \varepsilon_J H_J(S)$$

The TMLE update step is then

$$\varepsilon_n = \underset{\varepsilon}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(\bar{Q}^0_{Y,n}(\varepsilon))(O_i)$$

This can be achieved with standard multiple logistic regression software and converges in a single step. The targeted update is then

$$\bar{Q}^*_{Y,n} = \bar{Q}^0_{Y,n}(\varepsilon_n)$$

Note also that the single step convergence is true even for a partitioning of the outcomes space for S that contains overlapping subsets. The resulting estimator solves the J-component vector efficient influence curve estimating function, and is therefore unbiased and efficient for the vector parameter. Again, the TMLE will equal the empirical estimators for each of the components of the parameter vector.

3.2.8 TMLE for implicit constraints on $\bar{Q}^*_{Y,n}$

Now suppose we want to construct a prediction function estimator $\bar{Q}^*_{Y,n}(W)$ that will be used to classify patients as ”low”, ”medium”, or ”high” risk subsets according to some predetermined clinical cut points in the prediction space. In this scenario it is reassuring to see that, within each subset, the expected proportion of events according to the prediction function estimator is equivalent to the actual proportion of events. That is, we would like to achieve the calibration property for subgroups defined in the prediction space of the estimator. This involves enforcing an implicit constraint on the estimator $\bar{Q}^*_{Y,n}$ itself.

Formally, let $(I_{(a_j,b_j)}\{\bar{Q}^*_{Y,n}\} : j \in 1,..,J) = (\bar{Q}^*_{Y,n}(W) \in (a_j,b_j) : j \in 1,..,J)$ corresponding with a partitioning of the prediction space for the estimator of the conditional mean of Y given W, i.e., where for some j, (a_j,b_j) is an interval that defines a set of real numbers. The goal is to construct an estimator $\bar{Q}^*_{Y,n}$ of $\bar{Q}_{Y,0}$ that is unbiased and efficient for the vector parameter

$$\Psi(P_0) = \begin{bmatrix} E_{Q_{W,0}}[\bar{Q}_{Y,0}|\bar{Q}^*_{Y,n}(W) \in (a_1,b_1)] \\ \vdots \\ E_{Q_{W,0}}[\bar{Q}_{Y,0}|\bar{Q}^*_{Y,n}(W) \in (a_J,b_J)] \end{bmatrix} = \begin{bmatrix} E_{Q_{W,0}}[Y|\bar{Q}^*_{Y,n}(W) \in (a_1,b_1)] \\ \vdots \\ E_{Q_{W,0}}[Y|\bar{Q}^*_{Y,n}(W) \in (a_J,b_J)] \end{bmatrix}$$

The TMLE procedure is largely the same as that outlined in the previous section, except that here, the multivariate covariate in the parametric submodel depends on the estimator itself, $(H_j(\bar{Q}^k_{Y,n}) : j \in 1,..,J) = (I_{(a_j,b_j)}\{\bar{Q}^k_{Y,n}(W)\} : j \in 1,..,J)$. Thus the TMLE algorithm requires iteration as follows.

Initialize:

$$Logit(\bar{Q}^0_{Y,n}(\varepsilon)) = Logit(\bar{Q}^0_{Y,n}) + \varepsilon_1 H_1(\bar{Q}^0_{Y,n}) + \cdots + \varepsilon_J H_J(\bar{Q}^0_{Y,n})$$

$$\varepsilon^0_n = \underset{\varepsilon}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}(\bar{Q}^0_{Y,n}(\varepsilon)(O_i))$$

$$\bar{Q}^1_{Y,n} = \bar{Q}^0_{Y,n}(\varepsilon^0_n)$$

Iterate:

$$Logit(\bar{Q}^k_{Y,n}(\varepsilon)) = Logit(\bar{Q}^k_{Y,n}) + \varepsilon_1 H_1(\bar{Q}^k_{Y,n}) + \cdots + \varepsilon_J H_J(\bar{Q}^k_{Y,n})$$

$$\varepsilon^k_n = \underset{\varepsilon}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}(\bar{Q}^k_{Y,n}(\varepsilon)(O_i))$$

$$\bar{Q}^{k+1}_{Y,n} = \bar{Q}^k_{Y,n}(\varepsilon^k_n)$$

Stop when:

$$\|\varepsilon^k_n\| \approx 0$$

Again, the estimation of ε at each step can be achieved with standard software for multivariate logistic regression, and iterations are easily programmed with a loop. The final targeted update is $\bar{Q}^*_{Y,n}$. The TMLE $\Psi(Q_{W,n}, \bar{Q}^*_{Y,n})$ solves the vector efficient influence curve estimating equation and is therefore unbiased and efficient. This also implies that the TMLE equals the empirical mean of Y given $\bar{Q}^*_{Y,n}(W) \in (a_j,b_j) : j \in 1,\ldots,J$.

3.3 TMLE for calibration of the conditional intensity of a counting process, $\bar{Q}_{Y(t)}$

In this section we present the TMLE for calibration of the conditional intensity function within a counting process framework. This includes, as special cases, the hazard function in right-censored survival data with or without time-dependent covariates. We begin with description of the data structure, the nonparametric statistical model for the data, and the conditional intensity function parameter, defined as a mapping from the statistical model to a parameter space consisting of functions of the event history, covariates, being "at risk", and the time point, t. We identify several calibration parameters corresponding to calibration of (1) the t-specific conditional intensity; (2) the conditional intensity function over all time points; and (3) a weighted average of the t-specific intensities. These parameters are expressed as mappings applied to the statistical model. Each TMLE starts with an initial estimator of the conditional intensity function of the event history, and then uses a TMLE updating step to remove bias for the calibration parameters. Again, for brevity we discuss the TMLE procedure along with the properties of the estimator, reserving details on the influence curves for the Appendix.

3.3.1 Data, model, and conditional intensity parameter

Here we work with the counting process framework to describe time-dependent data structures that include time-dependent covariates. For the sake of presentation, we will assume that all random variables are discrete and that time is measured in discrete units. The latter assumption is especially reasonable in practical data analysis where time is measured in discrete units like seconds, days, years, etc. The truly continuous time case may be approximated by decreasing the length of the discrete time interval.

Consider a random variable T that represents the time until some event of interest. Let $L_1(t) = I(T < t)$ be the event counting process. Let $Y(t)$ be the indicator that this event counting process jumps just after time t, i.e., $Y(t) = L_1(t+1) - L_1(t)$. $R(t)$ is the indicator that a subject is in the "risk set" at time t. In right-censored survival data, for example, a person is not included in the risk set after they have experienced the failure event or after they have been censored. The counting process framework is general and allows a subject to enter, exit, or re-enter the "risk set" for any time intervals prior to the failure event or censoring. Finally let $\mathscr{F}_t$ is the history up to time t. It includes both baseline (time-independent) and time-dependent covariates. The observed data for any single subject at some time-point t can be represented as $O(t) = (R(t), R(t)\mathscr{F}_t, R(t)Y(t))$, and the observed data over all time-points is $O = (O(t) : t = 1, \ldots \tau)$. Suppose we observe n independently and identically distributed copies from $O \sim P_0 \in \mathscr{M}$. Let the statistical model $\mathscr{M}$ containing P_0 be nonparametric.

Consider the t-specific conditional intensity $\bar{Q}_{Y(t),0}(t)$, i.e., the conditional expectation of $Y(t)$ given $\mathscr{F}_t$ and $R(t) = 1$ for some t. The negative Bernoulli loglikelihood loss function for this

t-specific intensity is

$$\mathscr{L}(\bar{Q}_{Y(t)}(t))(O(t)) = -R(t)log[\bar{Q}_{Y(t)}(t)(\mathscr{F}_t,R(t))^{Y(t)}(1-\bar{Q}_{Y(t)}(t)(\mathscr{F}_t,R(t)))^{1-Y(t)}]$$

Note that $R(t)$ appears outside of the loglikelihood because we are only interested in the intensity for persons who are in the risk set at time t, i.e., conditional on $R(t)=1$. The t-specific intensity $\bar{Q}_{Y(t),0}(t)$ can be identified as the minimizer of the t-specific risk

$$\bar{Q}_{Y(t),0}(t) = \underset{\bar{Q}_{Y(t)}(t)}{\operatorname{argmin}}\, E_0[\mathscr{L}(\bar{Q}_{Y(t)}(t))(O(t))]$$

However, we want to estimate a prediction function for every t, i.e., the intensity function. $\bar{Q}_{Y,0} = (E_0[Y(t)|\mathscr{F}_t,R(t)=1] : t=1,\ldots,\tau)$. Consider the sum loss function over all the t-specific negative loglikelihood losses

$$\mathscr{L}(\bar{Q}_{Y(t)})(O) = \sum_{t=1}^{\tau} \mathscr{L}(\bar{Q}_{Y(t)}(t))(O(t))$$

Our intensity function parameter can be identified as the minimizer of the expectation of this sum loss function,

$$\bar{Q}_{Y(t),0} = \underset{\bar{Q}_{Y(t)}}{\operatorname{argmin}}\, E_0[\mathscr{L}(\bar{Q}_{Y(t)})(O)]$$

3.3.2 Initial estimator of conditional intensity, $\bar{Q}^0_{Y(t),n}$

We use as an initial estimator a Super Learner described in Chapter 2. In brief, we construct a long format data set with one row per subject per time-point for which $R(t)=1$. We construct a library of candidate estimators for the conditional intensity. The library may include, for example, pooled logistic regressions, artificial neural networks, recursive partitioning trees, or any other reasonable estimator for the conditional expectation of a binary outcome. We then find a convex weighted combination of the candidate that minimizes the V-fold cross validated risk, where the risk is defined as the expected sum loss over all t-specific negative Bernoulli loglikelihood loss functions.

3.3.3 Calibration of the intensity at a particular time-point, $\phi_0(t) = \Phi(P_0)$

Now suppose we would like our estimate of the prediction function to be calibrated with respect to a particular data subgroup at a particular time-point t. This means that our estimator should map into an unbiased estimate of some t-specific scalar parameter of the distribution of the data. Let $S_t = S(\mathscr{F}_t)$ be a real-valued summary measure of the history at a time t. Consider the scalar t-specific parameter

$$\phi_0(t) = \Phi(P_0) = E_0[\bar{Q}_{Y(t),0}(t)|I_{(\mathscr{A})}\{S_t\}=1,R(t)=1] = E_0[Y(t)|I_{(\mathscr{A})}\{S_t\}=1,R(t)=1]$$

where $I_{(\mathscr{A})}\{S_t\}$ is the indicator that the summary S_t measure falls in a set $\mathscr{A}$ in the outcome space for S.

As you might expect, the TMLE for this t-specific parameter is largely the same as the single-time point binary outcome case discussed in section 3.2.6. With the Super Learner fit as the initial estimator $\bar{Q}^0_{Y(t),n}$ and the t-specific negative Bernoulli loglikelihood as the loss function, the TMLE algorithm proceeds as follows:

$$Logit(\bar{Q}^0_{Y(t),n}(\varepsilon)) = Logit(\bar{Q}^0_{Y(t),n}) + \varepsilon_t H_t(R(t),S_t)$$

where $H_t(R(t),S_t) = \frac{R(t)}{E_0[I_{(\mathscr{A})}\{S_t\},R(t)]}$

$$\varepsilon_n = argmin_{\varepsilon}\frac{1}{n}\sum_{i=1}^{n}\mathscr{L}(\bar{Q}^0_{Y(t),n}(\varepsilon)(O_i)$$

This TMLE converges in a single step

$$\bar{Q}^*_{Y(t),n} = \bar{Q}^0_{Y(t),n}(\varepsilon_n)$$

3.3.4 Calibration of the intensity over all time-points, $(\phi_0(t) : t = 1,\ldots,\tau)$

Now suppose we wish to calibrate our estimator for a particular data subgroup at every time-point. This means our estimator should map into an unbiased and efficient estimate a parameter that is a function of t. Again, let $S_t = S(\mathscr{F}_t)$ be some real-valued summary measure of the history at a time t. Consider the function parameter $(\phi_0(t) : t = 1,\ldots,\tau) = \Phi(P_0) = (E_0[\bar{Q}_{Y(t),0}(t)|I_{(\mathscr{A})}\{S_t\} = 1,R(t) = 1] : t = 1,\ldots,\tau) = (E_0[Y(t)|I_{(\mathscr{A})}\{S_t\} = 1,R(t) = 1] : t = 1,\ldots,\tau)$ where $I_{(\mathscr{A})}\{S_t\}$ is the indicator that the summary S_t measure falls in a set $\mathscr{A}$ in the outcome space for S. This can also be viewed as a (possibly high-dimensional) vector parameter. The TMLE for this parameter must solve the efficient influence curve estimating equations at all $t = 1,\ldots,\tau$. The necessary fluctuation is then given by a τ-dimensional parametric submodel given by

$$Logit(\bar{Q}^0_{Y(t),n}(\varepsilon)) = Logit(\bar{Q}^0_{Y(t),n}) + \varepsilon_1 H_1(R(1),S_1) + \ldots + \varepsilon_\tau H_\tau(R(\tau),S_\tau)$$

where $H_t(R(t),S_t) = \frac{R(t)}{E_0[I_{(\mathscr{A})}\{S_t\},R(t)]}$

$$\varepsilon_n = argmin_{\varepsilon}\frac{1}{n}\sum_{i=1}^{n}\mathscr{L}(\bar{Q}^0_{Y(t),n}(\varepsilon))(O_i)$$

This TMLE converges in a single step

$$\bar{Q}^*_{Y(t),n} = \bar{Q}^0_{Y(t),n}(\varepsilon_n)$$

This TMLE may, however, be impractical if events are rare or there are many time points. In the latter case, the parametric submodel used to fluctuate our initial estimator becomes very high dimensional, and this may lead to practical difficulties in the estimation of the fluctuation submodel itself. It may therefore seem more reasonable to target a less ambitious parameter such as a weighted average of $(\phi_0(t) : t = 1,\ldots,\tau)$.

3.3.5 The "crude rate" $\bar{\phi}_0$ as a weighted average of $(\phi_0(t) : t = 1, \ldots, \tau)$

In right-censored survival data, an estimate of the "crude rate" is often defined as the number of observed events divided by total amount of observed person-time "at risk". It turns out that this crude rate can be expressed as a weighted average of the t-specific intensities, where the weights are determined by the number of persons "at risk" at each time point. Consider the parameter $\bar{\phi}_0$ corresponding to a weighted average of $\phi_0(t)$ over all $t = 1, \ldots, \tau$.

$$\bar{\phi}_0 = \frac{1}{\sum_t E_0[I_{(\mathscr{A})}\{S_t\}, R(t)]} \sum_t E_0[I_{(\mathscr{A})}\{S_t\}, R(t)]\phi_0(t)$$

We show below how this is equivalent to the common definition of "crude rate."

$$\frac{1}{\sum_t E_n[I_{(\mathscr{A})}\{S_t\}, R(t)]} \sum_t E_n[I_{(\mathscr{A})}\{S_t\}, R(t)]\phi_0(t)$$

$$= \frac{\sum_t E_n[I_{(\mathscr{A})}\{S_t\}, R(t)]\Phi(Q_{\mathscr{F}_t,n}(t), \bar{Q}^*_{Y(t),n}(t))}{\sum_t E_n[I_{(\mathscr{A})}\{S_t\}, R(t)]}$$

$$= \frac{\sum_t E_n[I_{(\mathscr{A})}\{S_t\}, R(t)] \frac{\frac{1}{n}\sum_{i=1}^n Y_i(t) I_{(\mathscr{A})}\{S_{t,i}\} R_i(t)}{\frac{1}{n}\sum_{i=1}^n I_{(\mathscr{A})}\{S_{t,i}\} R_i(t)}}{\sum_t E_n[I_{(\mathscr{A})}\{S_t\}, R(t)]}$$

$$= \frac{\sum_t \frac{1}{n}\sum_{i=1}^n [I_{(\mathscr{A})}\{S_{t,i}\} R_i(t)] \frac{\frac{1}{n}\sum_{i=1}^n Y_i(t) I_{(\mathscr{A})}\{S_{t,i}\} R_i(t)}{\frac{1}{n}\sum_{i=1}^n I_{(\mathscr{A})}\{S_{t,i}\} R_i(t)}}{\sum_t \frac{1}{n}\sum_{i=1}^n [I_{(\mathscr{A})}\{S_{t,i}\} R_i(t)]}$$

$$= \frac{\sum_t \sum_{i=1}^n Y_i(t) I_{(\mathscr{A})}\{S_{t,i}\} R_i(t)}{\sum_t \sum_{i=1}^n I_{(\mathscr{A})}\{S_{t,i}\} R_i(t)}$$

3.3.6 TMLE for the crude rate, $\bar{\phi}^*_n = \bar{\Phi}(Q_{\mathscr{F}_t,n}, \bar{Q}^*_{Y(t),n})$

We must now construct a targeted estimator, $\bar{Q}^*_{Y(t),n} = \{\bar{Q}^*_{Y(t),n}(\mathscr{F}_t, R(t) = 1)(t) : t = 1, \ldots, \tau\}$ via a fluctuation of our initial estimator $\bar{Q}^0_{Y(t),n}$. Recall the sum loss over all t-specific negative loglikelihood loss functions

$$\mathscr{L}(\bar{Q}_Y)(O) = \sum_t \mathscr{L}(\bar{Q}_{Y(t)}(t))(O(t)) = -\sum_t R(t) log\left\{ (\bar{Q}_{Y(t)})^{Y(t)} (1 - \bar{Q}_{Y(t)})^{(1-Y(t))} \right\}$$

In this case, the parametric fluctuation submodel is a univariate logistic regression pooled over all t

$$Logit(\bar{Q}^0_{Y(t),n}(\varepsilon)) = Logit(\bar{Q}^0_{Y(t),n}) + \varepsilon C H(R(t), S_t)$$

where $H(R(t), S_t) = R(t) I_{(\mathscr{A})}\{S_t\}$, and $C = \frac{1}{\sum_t E_0[R(t), I_{(\mathscr{A})}\{S_t\}]}$. Because C is a constant, its value can be subsumed in the estimation of ε, and the submodel can be expressed more simply as

$$Logit(\bar{Q}^0_{Y(t),n}(\varepsilon)) = Logit(\bar{Q}^0_{Y(t),n}) + \varepsilon H(R(t), S_t)$$

$$\varepsilon_n = \underset{\varepsilon}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(\bar{Q}^0_{Y(t),n}(\varepsilon)(O_i)$$

This TMLE converges in a single step

$$\bar{Q}^*_{Y(t),n} = \bar{Q}^0_{Y(t),n}(\varepsilon_n)$$

The targeted update $\bar{Q}^*_{Y(t),n}$, along with the empirical marginal distributions $(Q_{\mathscr{F}_t,n} : t = 1, \ldots, \tau)$, solve the efficient influence curve estimating equation for $\bar{\phi}_0$ and can therefore be mapped into an unbiased and efficient estimator of the weighted average intensity given $I_{(\mathscr{A})}\{S_t\} = 1$. It also implies that TMLE mapping will be equal to the empirical "crude rate".

3.3.7 TMLE for a vector of crude rates

In practice we may want our intensity function estimator to map into an unbiased and efficient estimation of the "crude rate" for several subpopulations according to their histories. For example, in medical outcome prediction we may desire an estimator that maps into an unbiased estimator estimator for the crude rates for several key patient subgroups. This corresponds with the unbiased estimation of a vector of weighted average parameter. The TMLE for the scalar weighted average discussed in the previous section is easily extended to a J-dimensional vector weighted average parameter corresponding to a partitioning of the outcomes space for $S(\mathscr{F}_t)$. The only difference is that the parametric submodel becomes J-dimensional with a covariate corresponding to each subgroup defined by the partitioning. The parametric fluctuation submodel is pooled over all t

$$Logit(\bar{Q}^0_{Y(t),n}(\varepsilon)) = Logit(\bar{Q}^0_{Y(t),n}) + \varepsilon_1 H_1(R(t), S_t) + \ldots + \varepsilon_J H_J(R(t), S_t)$$

where $H_j(R(t), S_t) = R(t) I_{(\mathscr{A}_j)}\{S_t\}$.

$$\varepsilon_n = \underset{\varepsilon}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(\bar{Q}^0_{Y(t),n}(\varepsilon)(O_i)$$

This TMLE converges in a single step

$$\bar{Q}^*_{Y(t),n} = \bar{Q}^0_{Y(t),n}(\varepsilon_n)$$

The targeted update $\bar{Q}^*_{Y(t),n}$, along with the empirical marginal distributions $(Q_{\mathscr{F}_t,n} : t = 1,\ldots,\tau)$, solves the efficient influence curve estimating equation for the J-dimensional vector parameter $\bar{\phi}_0$ and therefore maps into an unbiased and efficient estimator of the weighted average intensities corresponding to the specific partitioning. That is, the TMLE mapping will be equal to the empirical "crude rate" for each of the subgroups defined by the partitioning.

3.3.8 TMLE to enforce a vector of implicit constraints on $\bar{Q}^*_{Y(t),n}$

Suppose we now use our conditional intensity function estimator to classify patients as "low", "medium", or "high" risk subsets according to the predictions from our estimator of $\bar{Q}_{Y(t),0}$. It is reassuring to know that the empirical mean of the predictions within each subset is unbiased and efficient for the true conditional intensity within each subset. To achieve this, we enforce an implicit constraint on the estimator, $\bar{Q}^*_{Y,n}$, itself. The TMLE described in the previous section may be extended to achieve this through an iterative procedure.

Let $I_{(a_j,b_j)}\{\bar{Q}^*_{Y(t),n}\}$ be the indicator that $\bar{Q}^*_{Y(t),n} \in (a_j,b_j) : j = 1,\ldots,J$ corresponding with a partitioning of the prediction space for our estimator of the conditional intensity, i.e., where (a_j,b_j) is an interval that defines a set of real numbers. The goal is now to construct an estimator $\bar{Q}^*_{Y(t),n}$ of $\bar{Q}_{Y(t),0}$ that maps into an unbiased and efficient estimator of the vector parameter

$$\Phi(P_0) = (E_0[Y(t)|I_{(a_j,b_j)}\{\bar{Q}^*_{Y(t),n}\} = 1, R(t) = 1] : j = 1,,J)$$

This time, let the covariate of the parametric fluctuation model be

$$(H_j(\mathscr{F}_t, R(t)) : j = 1,,J) = (R(t)I_{(a_j,b_j)}\{\bar{Q}^*_{Y(t),n}\} : j = 1,,J)$$

Because this involves the estimator itself the TMLE procedure must be iterated until convergence.

Step 1:

$$Logit(\bar{Q}^0_{Y(t),n}(\varepsilon)) = Logit(\bar{Q}^0_{Y(t),n}) + \varepsilon_1 H_1(R(t),\bar{Q}^0_{Y(t),n}) + \ldots + \varepsilon_J H_J(R(t),\bar{Q}^0_{Y(t),n})$$

$$\varepsilon_n = \underset{\varepsilon}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}(\bar{Q}^0_{Y,n}(\varepsilon)(O_i)$$

$$\bar{Q}^1_{Y(t),n} = \bar{Q}^0_{Y(t),n}(\varepsilon_n)$$

Iterate:

$$Logit(\bar{Q}^k_{Y(t),n}(\varepsilon)) = Logit(\bar{Q}^k_{Y(t),n}) + \varepsilon_1 H_1(R(t),\bar{Q}^k_{Y(t),n}) + \ldots + \varepsilon_J H_J(R(t),\bar{Q}^k_{Y(t),n})$$

$$\varepsilon^k_n = \underset{\varepsilon}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}(\bar{Q}^k_{Y(t),n}(\varepsilon)(O_i)$$

$$\bar{Q}^{k+1}_{Y(t),n} = \bar{Q}^{k}_{Y(t),n}(\varepsilon^k_n)$$

Stop when:

$$\|\varepsilon^k_n\| \approx 0$$

3.3.9 TMLE in the exponential model

Consider a restricted model where $\bar{Q}_{Y(t),0}(t) : t = 1, \ldots, \tau$ does not depend on t. This implies that our calibration parameter $(\phi_0(t) : t = 1, \ldots, \tau) = \phi_0$ also does not depend on t. Readers familiar with parametric survival analysis will recognize that this assumption corresponds with the conditional exponential distribution of survival times, and that the "crude rate" defined above is in fact the parametric maximum likelihood estimate of the conditional intensity or hazard function under this model. We claim that the TMLE for $\bar{\phi}_0$, the weighted average in the nonparametric model, is also the TMLE for ϕ_0 in this restricted exponential model. To see this, note that neither the initial estimator $\bar{Q}^0_{Y(t),n}$ nor the clever covariate H depends on t. As a result, the targeted update $\bar{Q}^*_{Y(t),n}$ also does not depend on t and thus stays in the exponential model.

3.4 Simulation Results

As shown in the previous sections of this paper the TMLE updating procedure achieves the calibration property with respect to the empirical distribution of a given sample for any initial estimator of the prediction function. This result suggests that calibration with respect to certain subgroups should not be viewed as a measure of predictive performance, but instead as a reassuring constraint that we as investigators may choose to impose. Further, it forces us to think carefully about how we should go about the estimation of the constrained prediction function.

In this section we consider two approaches, A and B. Both approaches are two step procedures based in part on data-adaptive Super Learning. Each approach begins with a library of three candidate estimators (parametric linear discriminant analysis, additive logistic regression, and a recursive partitioning tree) of the prediction function. Approach A first constructs a convex combination of these estimators via data-adaptive Super Learning and then updates the convex combination Super Learner fit using the TMLE procedure for calibration. In approach B we first apply the TMLE update procedure of calibration to each candidate estimator individually, and then we choose the calibrated estimator with the lowest cross validated risk. The latter step is commonly known as model selection via cross validation, which is a (discrete) special case of the general Super Learner methodology. Approach B more closely follows the guidelines indicated by Super Learning theory, while approach A is more computationally convenient.

The important questions to ask are: (1) Which approach, A or B, achieves the best predictive performance as assessed by a valid loss or risk function? (2) How does the TMLE update for calibration affect the overall predictive performance of the initial estimator? and (3) Does the

TMLE-calibrated prediction function reduce bias for the target calibration parameter as assessed in a large independent validation data set?

Both approaches are applied to a training data set consisting of 10,000 observations, $O_1, \ldots, O_n$, from the data structure $O = (Y, W_1, W_2, W_3, W_4) \sim P_0$. The outcome $Y \in \{0,1\}$ is binary. The first 2 covariates, $W_1, W_2 \sim \mathscr{N}(0,1)$ are standard normal variates, and the last 2 covariates, $W_3, W_4 \sim \mathscr{B}(0.5)$ are Bernoulli variates with probability 0.5. The prediction function is

$$\begin{aligned}\bar{Q}_{Y,0} &= E_0[Y|W_1,W_2,W_3,W_4] \\ &= expit(0.001W_1 + 0.01W_2 - 0.5W_3 + 0.5W_4 \\ &\qquad - 0.2W_3W_1 + 0.05W_1^3 - sin(W_2W_3))\end{aligned}$$

where $expit(x) = \frac{1}{1+e^{-x}}$. The risk, here defined as the expectation of the negative Bernoulli log-likelihood function, of this true prediction function is approximately 1.30.

Denote the initial estimator of this function for each approach, $\bar{Q}_{Y,n,A}$ and $\bar{Q}_{Y,n,B}$, respectively. For approach A we define 5 non overlapping data subgroups defined by $\{\bar{Q}_{Y,n,A} \in (a_j, b_j) : j = 1, \ldots, 5\}$. We then use TMLE to construct the updated estimator $\bar{Q}^*_{Y,n,A}$ with the calibration property $\{E_n[\bar{Q}^*_{Y,n,A}|\bar{Q}^*_{Y,n,A} \in (a_j,b_j)] = E_n[Y|\bar{Q}^*_{Y,n,A} \in (a_j,b_j)] : j = 1, \ldots, 5\}$. We do the same thing for approach B. The calibration subgroups used here were defined in the outcome space by predicted probabilities in the the intervals [0, 0.37), [0.37, 0.50), [0.50, 0.56), [0.56, 0.62), [0.62, 1.0), which correspond roughly to the quintiles of the true prediction function values. The resulting calibrated estimators $\bar{Q}^*_{Y,n,A}$ and $\bar{Q}^*_{Y,n,B}$ are compared with respect to the risk on a large validation sample consisting of 1,000,000 observations.

Table 3.1: Calibration with Super Learner: Comparison of two approaches

Competing Approaches	Risk on validation data
A. TMLE update applied to convex Super Learner	1.330
B. Discrete Super Learner applied to TMLE-updated library	1.335

Though the implementations of approaches A and B are considerably different, in this simulation their risks were close enough not to make any material distinctions in the results. This suggests that the computationally convenient approach A, is reasonable to use in practice. In theory, one could consider the estimator in approach A to be a particular additional calibrated candidate estimator to be used in the library for approach B. The risk of the convex Super Learner used in approach A before the TMLE calibration update was 1.334. This is not markedly different from - and actually somewhat higher than - the risk after the TMLE calibration update. Though somewhat unexpected, in this simulation the TMLE update to enforce the calibration property slightly helped the predictive performance of the initial estimator.

The TMLE update reduces bias for the calibration parameter on the large independent data set. As discussed in the derivation in this paper, the reason is that the TMLE is exactly equal to the empirical mean (in the training data) of the outcome within each subgroup, and these empirical means are unbiased and efficient estimators.

Table 3.2: Calibration property before and after TMLE update

Interval $\{a_j, b_j\}$	$E_0[Y - \bar{Q}_{Y,n,A} \vert \bar{Q}_{Y,n,A} \in \{a_j, b_j\}]$	$E_0[Y - \bar{Q}^*_{Y,n,A} \vert \bar{Q}^*_{Y,n,A} \in \{a_j, b_j\}]$
$[0, 0.37)$	-0.052	0.011
$[0.37, 0.50)$	0.013	0.005
$[0.50, 0.56)$	0.031	-0.012
$[0.56, 0.62)$	0.020	-0.000
$[0.62, 1.0)$	-0.041	-0.019

3.5 Application: Calibration of the conditional stroke intensity Super Learner

One of the research objectives in ATRIA-1 was to develop a scheme to classify patients who were not currently on warfarin as ”low”, ”medium”, or ”high” stroke risk on the basis of their medical history. This classification could then provide a simple summary to assist clinical warfarin prescription decisions. The first step, whose implementation was described in Chapter 2, was to construct an estimator the conditional stroke intensity function that mapped patient medical histories into an annualized stroke rate. The predicted stroke rates were then classified as ”low”, ”medium”, or ”high” according to specified clinical cut points. While minimization of the negative Bernoulli loglikelihood risk provided a valid objective for the estimation of the stroke intensity function, the co-investigators wanted to make sure that the expected stroke rates for the Super Learner were in fact close to the actual stroke rates within each classification level. Thus, calibration in the sense described in section 3.3.8 was a desired property.

3.5.1 Data, statistical model, and conditional stroke intensity parameter

The ATRIA-1 dataset includes time-dependent indicators of whether a patient was on warfarin therapy, presence of certain comorbidities, and lab value measures. The data structure is that of Section 4, namely, $O = (R(t), R(t)\mathscr{F}_t, R(t)Y(t) : t = 1, \ldots, \tau) = (O(t) : t = 1, \ldots, \tau)$. $Y(t)$ is the indicator that a person experienced the stroke event on day t. $\mathscr{F}_t$ is the medical history which includes age, gender, race, education, income, diagnoses of various comorbidities including prior stroke, diabetes mellitus, heart failure, coronary artery disease, bleeding events, falls, dementia, seizures, hypertension, etc., most recent lab values for total hemoglobin, HgbA1C, total white blood cells, serum creatinine, estimated glomerular filtration rate, and proteinuria. $R(t)$ is the indicator that the person is in the ”at risk” at time t. Persons who experienced the event or were censored before time t necessarily have $R(t) = 0$. Also, because we were only interested in the conditional stroke intensity in persons who were not currently on warfarin medication, $R(t)$ was set to 0 during time periods for which a person was on warfarin.

We allow the statistical model P_0 of O to be nonparametric, with a conditional stroke intensity parameter defined as the minimizer:

$$\bar{Q}_{Y,0} = \underset{\bar{Q}_Y}{\text{argmin}}\, E_0[\mathscr{L}(\bar{Q}_Y)(O)]$$

where $\mathscr{L}(\bar{Q}_Y)(O) = \sum_t \mathscr{L}(\bar{Q}_{Y(t)}(t)(O(t))$, and

$$\mathscr{L}(\bar{Q}_{Y(t)}(t))(O(t)) = -R(t)log\left\{\bar{Q}_{Y(t)}(\mathscr{F}_t, R(t))^{Y(t)}(1-\bar{Q}_{Y(t)}(\mathscr{F}_t, R(t)))^{(1-Y(t))}\right\}$$

3.5.2 Super Learner for the conditional stroke intensity

The initial estimator of the conditional intensity function, $\bar{Q}^0_{Y(t),n}$ was fitted with the Super Learner described in Chapter 2. In brief, the Super Learner methodology was implemented in SAS software and consisted of 21 candidate estimators. These included the null (unconditional mean) estimator, logistic regression, linear discriminant analysis, artificial neural network multilayer perceptrons, decision trees, and a boosting algorithm. In addition, several of the candidate estimators also included four strategies for explanatory variable selection: all main terms, main terms for which univariate logistic regression gave a p-value < 0.05, main terms for variables with a positive variable importance based on a decision tree, and main terms selected by a lasso-type (L1-regularization) algorithm.

3.5.3 Crude rate calibration for three patient subgroups

For completeness, we start with a relatively simple demonstration of the TMLE calibration of the initial Super Learner estimator with respect to the crude stroke rate in three broad patient subgroups: (1) men older than 75 years of age, (2) women with prior history of stroke, and (3) persons with diabetes mellitus. This corresponds with targeting the the 3-dimensional weighted average parameter

$$\bar{\phi}_0 = \frac{1}{\sum_t E_0[R(t), I_{(\mathscr{A}_j)}\{S_t\}]} \sum_t E_0[R(t), I_{(\mathscr{A}_j)}\{S_t\}]\phi_0(t) : j = 1,2,3$$

The TMLE for this vector parameter converges in a single step. The empirical means of the initial estimator, the TMLE, and the nonparametric empirical estimator are given below.

3.5.4 Crude rate calibration for ”low”, ”medium”, and ”high” risk groups

We now present the calibration for the three-level ”low”, ”medium”, and ”high” stroke incidence/intensity classification system. Under the current clinical guidelines ”low” was defined as an annualized intensity of less than 1 stroke per 100 patients per year, ”medium” was between 1 and 2 strokes per 100 patients per year, and ”high” was greater than 2 strokes per 100 patients per year.

Table 3.3: Calibration of annualized intensities (strokes/person-year) by patient subgroups

Patient subgroup	Mean of initial estimator	Mean of TMLE estimator	Empirical crude ”stroke rate”
Men, age > 75	2.52	2.47	2.47
Women with prior stroke	7.02	8.05	8.05
Persons with diabetes	2.98	3.01	3.01

The within class mean predictions from our initial Super Learner estimator were reasonably close to the within class empirical stroke rates, but they were not completely unbiased for this calibration parameter. We used the TMLE procedure outlined in section 3.3.8 to calibrate our initial Super Learner estimator, such that it mapped into a completely unbiased and efficient estimator of the within-class stroke rates. The results are given below.

Table 3.4: Calibration of annualized intensities (strokes/person-year) by risk class

Risk	Initial estimator		TMLE	
Class	Crude rate	Mean prediction	Crude rate	Mean prediction
”*low*”	0.36	0.58	0.40	0.40
”*medium*”	1.30	1.46	1.42	1.42
”*high*”	4.29	3.87	4.29	4.29

The TMLE calibration procedure achieved convergence after 3 iterations. As shown above, the final estimator is perfectly calibrated in that it maps into the empirical stroke rate within each risk class, and is therefore an unbiased and efficient estimator of the within-class stroke rate parameter. Here one of the interesting aspects of the implicit constraints becomes apparent. Note that because clinical risk classes are defined by the estimator itself, as the estimator is updated, the empirical stroke rates within each class are also updated. This is why the TMLE for implicit constraints on the estimator itself requires iteration. It also shows explicitly, why Hosmer and Lemeshow type goodness-of-fit tests may not be particularly useful in the context of model selection, particularly when applied to the data used to fit the prediction function estimator. As shown above, our procedure will ensure the expected equal to the observed which necessarily makes the test statistic equal to 0.

3.6 Discussion

In this article we presented a new TMLE procedure for use in the calibration of a prediction function estimator. We demonstrated how an initial estimator of a prediction function may be updated through targeted fluctuations in such a way that the resulting targeted estimator may be mapped into an unbiased and efficient estimator of certain pre-specified features of the distribution of the data. We showed that, when iterated, the same procedure can be used to enforce implicit constraints on the prediction function estimator itself. We developed the TMLE procedure in the context of the conditional expectation of single-time point binary outcome and in the context of the estimation of the conditional intensity function in right-censored survival data or the more general counting process framework, and showed how to enforce both scalar and vector constraints. We explored through simulation the impact of enforcing calibration constraints on predictive performance as assessed by loss function methods. Finally we demonstrated calibration of an estimator of the conditional stroke intensity prediction function using real-world data on individuals with atrial fibrillation.

The methodology presented has important implications for the practice and assessment of statistical prediction. At the most fundamental level, our new procedure augments the conventional wisdom of global bias-variance trade offs, by providing a means to remove bias for a priori specified local features of the data distribution related to the prediction function. This may prove to be particularly useful in medical risk prediction where unchecked bias could lead to inferior decision making for particular patient subgroups. Our iterative TMLE procedure, which can be used to enforce implicit constraints on prediction function estimators, represents a novel use for TMLE and is of interest in its own right for several reasons. Firstly, it solves any calibration problem defined by comparing the mean of the predictions generated by an estimator and observed frequencies of the outcome according to strata defined by intervals of predictions themselves. Taking this further, however, the fact that we can achieve this calibration property for *any* choice of initial estimator calls into question the validity of the Hosmer and Lemeshow type goodness-of-fit tests. In particular, using the procedures described here it is possible to construct an entire library of TMLE-calibrated estimators that all achieve a Hosmer and Lemeshow test statistic exactly equal to 0, but whose predictive performance assessed in terms of a valid loss function may be wildly different.

In our view, a valid risk function should be primary assessment metric for prediction function estimators in nonparametric statistical models. It was reassuring to see in our simulation that the TMLE calibration procedure actually improved predictive performance assessed in terms of loglikelihood risk. It should be noted, however, that such improvements are not always guaranteed and imparting a calibration constraint may result in decreased predictive performance assessed in terms of a risk function. If, as in the present article, calibration or unbiasedness for other specific data features is a desired property, candidate estimators that achieve this should be combined with risk-based methods, e.g., Super Learning, to ensure adequate predictive performance. Finally we note that the theory underlying the TMLE procedures illustrated here in the context of the conditional expectation of binary outcomes is general and may be easily adapted to other types of outcomes including continuous may be generalized rather to any number of other scenarios, including the conditional expectation continuous outcomes or conditional survival probabilities given covariates.

Chapter 4

Targeted maximum likelihood estimation for the treatment-specific marginal mean in right-censored survival data

4.1 Introduction

The estimation of causal effects of therapeutic interventions on survival is central to evidence-based medicine. Ideally, we would like to know whether a particular therapeutic intervention will induce a beneficial or adverse effect on survival for each individual patient. The problem of individualized prediction is, however, notoriously difficult. In light of the difficulties with individualized predictions, investigators often focus on the less ambitious task of estimating the marginal causal effect of an intervention, e.g., the effect of an intervention on the cumulative probability of deaths at various time points for entire population. For example, the marginal causal effect of an intervention may be defined as the overall proportion of individuals who die within one year if everybody were to receive the intervention minus the overall proportion of individuals who die within one year if nobody were to receive the intervention.

The gold standard for the estimation of causal effects of medical interventions is the randomized controlled trial (RCT). In theory, RCTs may achieve consistent estimates of marginal causal effects in nonparametric statistical models via randomization, a process in which the intervention is assigned independently of other covariates. Heuristically, randomization seeks to achieve balance between intervention groups with respect to other factors that may be predictive of the outcome of interest. If randomization is successful then the difference in the average health outcome between the treatment and control intervention arms may be attributed specifically to the differences in the interventions assigned.

While the importance of RCTs for causal inference cannot be understated, there are several practical concerns that require attention in the interpretation of their results. First, the theoretical benefits of randomization may not be realized in a particular sample of patients. That is, some covariates may prove to be unbalanced across intervention groups. If these same covariates are related to the outcome, a phenomenon known as 'empirical confounding', the marginal causal effect estimate from the RCT will be biased. A second and equally important concern in the context of time-to-event outcomes is study dropout. Often RCTs for time-to-event outcomes use the well-known Kaplan-Meier (KM) (Kaplan and Meier, 1958) product limit estimator to compute estimates of a causal effect on the marginal survival probabilities. The consistency of this estimator relies not only on successful randomization, but also on the assumption that right-censoring due to study dropout is also independent. In medical trials, however, this assumption rarely holds in practice. Typically, the sickest subjects are more likely to drop out of the study simply because it is too difficult or unethical to continue. In this case the KM overestimates the survival probability. Conversely if the healthiest subjects do not perceive any benefit of the intervention, and because of this choose to drop out, the KM underestimates the survival probability.

In the last two decades several estimators have been proposed that, under certain assumptions, achieve consistent estimation under both 'empirical confounding' and dependent censoring. These relatively new estimators achieve this by incorporating information from an observed fixed or time-dependent covariate process to adjust for imbalance in the treatment groups or for dependent censoring. Their consistency relies on the assumptions that (1) the distribution of intervention assignments and censoring times can be estimated consistently as a function of the observed covariate process, or (2) the distribution of the outcome can be estimated consistently as a function of the observed covariate process. For example, the consistency of the Inverse Probability of Censoring Weighted (IPCW) estimator (van der Laan and Robins, 2003), which is constructed as the solution to the IPCW estimating equation, relies strictly on (1). The consistency of Maximum Likelihood Estimation-based (MLE) substitution estimators relies strictly on (2). Augmented IPCW (van der Laan and Robins, 2003) and Targeted Maximum Likelihood Estimators (TMLE) (van der Laan and Rose, 2011) are doubly robust in the sense that they are consistent under (1) or (2) and are efficient if both (1) and (2).

A separate but equally important issue, aside from consistent estimation within RCTs themselves, is the concern that causal effect estimates from RCT populations may not generalize to the clinical patient population in which approved drugs are routinely prescribed. Trial participants are, on average, healthier and have fewer comorbidities than typical patients for whom prescriptions are routinely written. Further, the medical care provided in the RCT may be more intensive than what is typical in the general population and could potentially bias results. In light of this, there has been increasing interest in the estimation of marginal causal effects with observational data from typical patient populations. Fortunately, the same methods used to address 'empirical confounding' and dependent censoring in RCTs may also be applied to the estimation of causal effects from observational data. The extent to which causal effects are identifiable in observational data is dependent on the measurement of all potential confounders and on having a positive probability of intervention for each patient in the population under study.

In this chapter, we implement the recent TMLE developed in van der Laan and Gruber (2011) for estimation of intervention-specific marginal means in general longitudinal data structures. In brief, the algorithm computes the estimator by exploiting iterative law of conditional expectation, also known as the tower property. It starts with an initial estimator of the conditional expectation at the final time-point, and updates this to reduce bias for the intervention-specific marginal mean parameter. The updated conditional expectation then serves as the outcome in for the next conditional expectation in the time-ordered sequence, and it too is updated. The process iterates across all time-points until the zeroeth, at which point the resulting mapping defines the TMLE. The implementation here is specific to right-censored survival data with time-dependent covariates. The marginal mean parameter here corresponds to the cumulative probability of the terminal event by a specific time-point. The right-censored survival data model imposes particular constraints on the probability distribution, which allows a more computationally efficient implementation than that for the general algorithm presented in van der Laan and Gruber (2011).

Section 4.2 lays out a road map for statistical and causal effect estimation. We start with a formal definition of the data structure, write down its likelihood, and commit to a nonparametric statistical model. We define our statistical parameter as a mapping applied to the likelihood of the observed data. In particular, the mapping defines the marginal cumulative event probability at a specific time-point via recursive applications of the law of iterated conditional expectation. We conclude the section by outlining additional assumptions under which estimation of the statistical parameter is equivalent estimation of the causal parameter in a formal structural causal model. Section 4.3 presents in detail the van der Laan and Gruber (2011) TMLE algorithm for the marginal cumulative event probability. The implementation is slightly modified from that in the original van der Laan and Gruber (2011) article to fully exploit information in the nonparametric statistical model for right-censored survival data. We reserve technical details on the efficient influence curve for the Appendix. In Section 4.4, we simulate a data structure with right-censored survival times, a fixed treatment, a covariate process comprised of both fixed and time-dependent covariates, and a censoring process that is dependent on the treatment and covariate processes. We apply the new TMLE algorithm and compare the results with the well-known KM and more recent IPCW estimators under positivity violations and model misspecification. Section 4.5 applies the TMLE to the ATRIA-1 research study to estimate the additive causal effect of sustained warfarin therapy versus no warfarin therapy on the marginal cumulative probability of stroke or death by 1 year, and the results are compared to those obtained with the KM and IPCW estimators.

4.2 Data, model, and parameter

4.2.1 Data structure, O

Consider a right-censored survival data structure in which subject covariates are assessed as some baseline time-point. After this assessment, the subjects are assigned a treatment and are then followed up until they experience some terminal event of interest or are censored, for example, being lost to follow-up, study drop out, or simply administrative censoring at the end of the study. In many situations, covariates whose values may change over time, i.e., time-dependent covariates, are collected throughout the follow-up period. These time-dependent covariates may be predictive of the survival/death outcome or may be predictive of censoring. It is convenient, as in the previous chapters, to provide a formal representation of this data as a random variable in a counting process framework. Again, for notational convenience, assume that time-points are discrete.

Let t be the daily time counter. $(L_1(t) \in \{0,1\} : t = 0,\dots,\tau)$ is the event counting process. In brief, it is a random variable that takes value 0 until the terminal event is observed. If the event is observed at time t the value switches to 1 at all subsequent time points. We assume that $L_1(0) = 0$, i.e., everybody is alive at the beginning of follow-up. In the previous chapters, we characterized the data structure with a general history denoted $\mathscr{F}_t$ and an "at risk" indicator $R(t)$, Here it is helpful to refine these objects into specific components. In particular, let $(L_2(t) \in \mathbb{R}^d : t = 0,\dots,\tau)$ be the covariate process, which includes both fixed and time-dependent covariates. $(A_1(t) \in \{0,1\} : t = 0,\dots,\tau)$ is the treatment process. It is a random variable that takes value 0 if the subject is not treated at time t and 1 if he is treated. $A_2(t)$ is the counting process for censoring. Like the event counting process this is a random variable that takes value 0 before the subject is censored and then becomes fixed to value 1 from the time that censoring is observed. By convention, we assume that within a particular time t, the order of observation is $L_1(t) \to L_2(t) \to A_1(t) \to A_2(t)$.

For notational convenience we define $\bar{L}(t)$ to be the history of the event and covariate processes up to time point t. We define $\bar{A}(t)$ to be the the history of the treatment and the censoring processes. Doing so allows a definition of the time-specific data structure $O(t) = \Big(A_2(t), (1-A_2(t))\bar{A}(t), (1-A_2(t))\bar{L}(t)\Big)$. That is, if a person has not been censored, we observed their event, covariate, and treatment histories up to time t. The full data for an individual subject is then the collection of these time-specific structures taken over all time-points, $O = \Big(O(t) : t = 0,...,\tau\Big)$.

4.2.2 Factorization of the likelihood, P_0

Assume that our data is a random variable that follows a particular likelihood, $O \sim P_0$. This likelihood is the joint density of the random variables that make up O. As will be seen, it is useful to factorize P_0 into orthogonal components indexed by the time-ordering t. Start by noting that the joint density of O can be factorized into the conditional densities of all time specific data-structures, $O(t) : t = 0,\dots,\tau$, given the history.

$$P_0(O) = \prod_{t=0}^{\tau} P_0\Big(O(t)\Big|Pa\big(O(t)\big)\Big)$$

where $Pa\big(O(t)\big)$ denote the parents, i.e., all random variables that are time-ordered before $O(t)$. Now, each conditional density of each time-specific data structure given the history can be further factorized into the conditional densities of the event, covariate, treatment, and censoring processes given the history.

$$= \prod_{t=0}^{\tau} P_0\Big(L_1(t)\Big|Pa\big(L_1(t)\big)\Big) P_0\Big(L_2(t)\Big|Pa\big(L_2(t)\big)\Big)$$

$$\prod_{t=0}^{\tau} P_0\Big(A_1(t)\Big|Pa\big(A_1(t)\big)\Big)P_0\Big(A_2(t)\Big|Pa\big(A_2(t)\big)\Big)$$

From here on out we will refer to the conditional densities of the event and covariate processes as Q-factors and the conditional densities of the treatment and censoring processes as g-factors of the likelihood. We will denote the conditional expectations with the bar notation, $\bar{Q}$ and $\bar{g}$.

$$P_0(O) = \prod_{t=0}^{\tau} Q_{0,L_1(t)}(O(t))Q_{0,L_2(t)}(O(t))$$
$$\prod_{t=0}^{\tau} g_{0,A_1(t)}(O(t))g_{0,A_2(t)}(O(t))$$

4.2.3 Statistical model, $\mathscr{M}$

At this point we must commit to a statistical model, $\mathscr{M}$, which is a set of assumption we wish to impose on P_0. Formally, $\mathscr{M}$ represents a collection of potential probability distributions (or likelihoods) of which P_0 is the particular member that generates our data as the random variable O.

$$O \sim P_0 \in \mathscr{M}$$

In general we would like to avoid strong distributional assumptions on P_0, thus allowing our statistical model, M, to be as nonparametric as possible. There are, however, four constraints that are implicit in right-censored survival data. The first two constraints are that a subject cannot re-enter the "risk set" after experiencing the event or censoring:

$$\bar{Q}_{0,L_1(t)}\big(L_1(t-1)=1,\bar{L}(t-1),\bar{A}(t-1)\big) = 1$$

$$\bar{g}_{0,A_2(t)}\big(\bar{L}(t-1),A_2(t-1)=1,\bar{A}(t-2)\big) = 1$$

An event cannot be observed after censoring has been observed:

$$\bar{Q}_{0,L_1(t)}\big(\bar{L}(t-1),A_2(t-1)=1,\bar{A}(t-1)\big) = 0$$

Finally censoring cannot be observed after the event has been observed:

$$\bar{g}_{0,A_2(t)}\big(L_1(t-1)=1,\bar{L}(t-1),\bar{A}(t-2)\big) = 0$$

4.2.4 Statistical parameter $\psi_0 = \Psi(P_0)$

Our statistical parameter of interest, ψ_0, is the marginal cumulative event probability (or cumulative incidence) at time-point $K+1$ according to a specific intervention, $\bar{A}(K) = \bar{a}$. Here we define the parameter as a mapping applied to the probability distribution in the statistical model, $\psi_0 = \Psi(P_0), \Psi :\to \mathscr{M}$. This parameter is identifiable under the positivity assumption,

$P_0\Big(\bar{A}(k) = \bar{a}(k)\Big|Pa\big(A(k)\big)\Big) > 0, a.e.$, i.e., the specific intervention must have a positive probability for all subject histories. Intuitively, if a certain patient subgroup will never get the specific intervention, then a marginal intervention-specific outcome that includes this subgroup makes no sense. Because the parameter is defined by a specific intervention then, if the positivity assumption holds, the parameter mapping is only a function of the Q-factors of P_0, i.e., $\psi_0 = \Psi(P_0) = \Psi(Q_0)$. This mapping is given below. For notational convenience here we drop the 0 subscript from the Q-factors.

Let $Y = L_1(K+1)$

By the law of iterated conditional expectation:

$$\bar{Q}_{Y|\bar{L}(K-0),\bar{a}} = E\Big[Y\big|\bar{L}(K-0), \bar{A}(K-0) = \bar{a}\Big]$$

$$\bar{Q}_{Y|\bar{L}(K-1),\bar{a}} = E\Big[\bar{Q}_{Y|\bar{L}(K-0),\bar{a}}\big|\bar{L}(K-1), \bar{A}(K-1) = \bar{a}\Big]$$

$$\bar{Q}_{Y|\bar{L}(K-2),\bar{a}} = E\Big[\bar{Q}_{Y|\bar{L}(K-1),\bar{a}}\big|\bar{L}(K-2), \bar{A}(K-2) = \bar{a}\Big]$$

$$\vdots$$

$$\bar{Q}_{Y|\bar{L}(1),\bar{a}} = E\Big[\bar{Q}_{Y|\bar{L}(2),\bar{a}}\big|\bar{L}(1), A(0) = \bar{a}\Big]$$

$$\bar{Q}_{Y|L(0),\bar{a}} = E\Big[\bar{Q}_{Y|\bar{L}(1),\bar{a}}\big|L(0)\Big]$$

$$\psi = E\Big[\bar{Q}_{Y|L(0),\bar{a}}\Big]$$

$$\psi = \Psi(Q) = \Psi\Big(\bar{Q}_{Y|L(0),\bar{a}}, \bar{Q}_{Y|L(1),\bar{a}}, \ldots, \bar{Q}_{Y|L(K),\bar{a}}\Big)$$

As noted above, the true parameter, ψ_0, is simply this same mapping applied to the true Q_0 distribution, i.e., $\psi_0 = \Psi(Q_0)$. This representation makes it clear that our parameter is only a function of the Q-factors through iterated conditional means at each time-point. This is very convenient from an estimation perspective. In general, the estimation of conditional means is much easier problem than the estimation of conditional densities.

4.2.5 Causal treatment-specific mean parameter, $\psi^{\bar{a}}$

We now move from the statistical parameter to the causal parameter. The estimation of a causal effect parameter requires a causal model. Consider that our counting process data structure can also be written as a directed acyclic graph in which each random variable at each time point is represented as a node. Because our model is largely nonparametric, each node has incoming directed edges from all other nodes that precede it according to the time-ordering. An equivalent causal model can be constructed as a nonparametric structural equation model (NPSEM) in which every node is a deterministic function of its parents and an exogenous random error (Pearl, 2009).

Now consider an intervention, $\bar{a}(K)$, which corresponds with a fixed intervention on the treatment and censoring processes. For example, we may choose

$$\bar{a}(K) = \big(A_1(0) = A_1(1) = \ldots = A_1(K) = 1, A_2(0) = A_2(1) = \ldots = A_2(K) = 0\big)$$

corresponding to a fixed treatment and no censoring up until at least time point K. Setting the values of all intervention nodes according to this intervention results in a system with a new probability density, $P^{\bar{a}}$, and our causal parameter may now be defined as the marginal cumulative event probability at time-point $t = K+1$ in this new system. Equivalently, the parameter is identified as a mapping applied to the intervention-specific probability distribution $P^{\bar{a}}$.

$$\psi^{\bar{a}} = \Psi(P^{\bar{a}}) = \Psi(Q^{\bar{a}}) = E^{\bar{a}}\Big[L_1(K+1)\Big]$$

Our observed data O is not generated by $P^{\bar{a}}$, but rather from P_0. Fortunately, we are under certain assumptions able to identify $\psi^{\bar{a}}$ as the mapping applied to P_0 described in the previous section. The first assumption concerns consistency of the structural causal model. The validity of this assumption is largely guaranteed by the time-ordering of the data structure and the NPSEM, which puts no substantive restrictions on causal relationships. The second assumption required for a causal parameter interpretation is that of sequential randomization, $A(k) \perp L^{\bar{a}(k)} | Pa\big(A(k)\big)$. The heuristic translation is that there are no unmeasured confounding variables. This assumption is untestable in practice, and therefore represents the major point of controversy in causal effect estimation. It is important to note, however, that even if this assumption is not valid, we may still arrive at an interesting parameter, best described as the association of the intervention with the outcome adjusted for a subset of confounders. If both assumptions are met then consistent estimation of the statistical parameter is equivalent to consistent estimation of the causal parameter.

4.2.6 Efficient influence curve D^*

Before constructing a TMLE, we must understand the efficient influence curve

$$D^*_{Q_0,g_0}(O) = \frac{I(\bar{A}(K) = \bar{a}(K))}{g_{0,K}}\big(L_1(K+1) - \bar{Q}_{Y|\bar{L}(K)}\big) + \sum_{k=0}^{K-1} \frac{I(\bar{A}(k) = \bar{a}(k))}{g_{0,k}}\big(\bar{Q}_{Y|\bar{L}(k+1)} - \bar{Q}_{Y|\bar{L}(k)}\big) + \bar{Q}_{Y|L(0)} - \psi_0$$

It is easy to see that the mean of D^* is 0. The variance of D^* divided by the sample size defines the efficiency bound for the etsimation of this parameter in the nonparametric statistical model. It follows that an estimator is efficient if and only its influence curve is equal to the efficient influence curve. The derivation of D^* is presented in the Appendix of this dissertation. As shown in the next section, the TMLE solves an efficient (targeted maximum likelihood) score equation, which spans D^*. As we show, this will also solve the efficient influence curve estimating equation, which is constructed in the usual way.

4.3 TMLE for the marginal cumulative event probability

The TMLE presented here is a special case of the general algorithmic template given in van der Laan and Gruber (2011) for the estimation of intervention-specific means in general longitudinal data structures. The latter was inspired in part by the estimating equation methods proposed by Bang and Robins (2005). The particular TMLE implementation discussed here is appropriate for right-censored survival data structures with a baseline treatment intervention and a covariate process that may include both fixed or time-dependent covariates. It allows for both treatment and censoring mechanisms that are dependent on either fixed or time-dependent covariates or on time itself. This TMLE requires the following ingredients:

1. g-factor estimator: $\left(\left(\bar{g}_{n,A_1(t)},\bar{g}_{n,A_2(t)}\right):t=0,\ldots,K\right)$

2. Q-factor initial estimator: $\left(\bar{Q}^{\bar{a},0}_{n,Y|L(0)},\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(1)},\ldots,\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)}\right)$

3. Loss function: Sum of Q-factor specific log-losses

4. Parametric submodel: Univariate (intervention-specific mean is a scalar)

5. Updating step: Maximum Likelihood Estimation

4.3.1 TMLE algorithm

As discussed in van der Laan and Gruber (2011) the TMLE algorithm iteratively solves the efficient score equation for each Q-component of the likelihood. It starts by estimating the conditional expectation of the outcome given the history up to the penultimate time-point. The recursive iterated conditional expectation at each time-point results in a TMLE that converges in a single pass through the observed data. The TMLE update for the Q-component at each time-point converges in a single step. The implementation presented here highlights how to fully exploit information in the statistical model for right-censored survival data. The algorithm is as follows.

Step $K+1$:

Initial Estimator: Propose an initial estimator,$\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)}$, of the innermost conditional expectation, $Q^{a}_{0,Y|L(K)}$. While one may be tempted to fit a regression straight away, some simple algebra allows more insight. Note that

$$\begin{aligned}\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)} = I(L_1(K) = 1) \\ + \\ I(L_1(K) = 0)E_n[Y^{\bar{a}}|\bar{L}^{\bar{a}}_2(K), L^{\bar{a}}_1(K) = 0]\end{aligned}$$

The first term reflects our knowledge that if the failure occurred at or before time K then event counting process is fixed at value 1 for all subsequent time points with probability 1, which implies that no update is required for observations with $I(L_1(K) = 1) = 1$. The second term is the expectation of the event process for persons who have not yet had the event and (due to the intervention settings) are not censored. In other words, this term corresponds to the intensity or hazard at time $t = K + 1$. A reasonable approach to constructing the initial estimator here might include estimation of intensity/hazard by smoothing over all time points and substituting the estimate at time $t = K$. This could be accomplished using a simple pooled logistic regression or with a Super Learner.

Log-loss function: Here we use the log-loss function for estimation. The log-loss at this step is given by

$$\mathscr{L}(\bar{Q}^{\bar{a}}_{Y|\bar{L}(K)}) = -log\Big\{ \big(\bar{Q}^{\bar{a}}_{Y|\bar{L}(K)}\big)^{Y} \big(1 - \bar{Q}^{\bar{a}}_{Y|\bar{L}(K)}\big)^{1-Y} \Big\}$$

Because $Y \in \{0, 1\}$, the log-loss is valid in the sense that its expectation is minimized at the true conditional expectation $\bar{Q}^{\bar{a}}_{Y|\bar{L}(K)} = \bar{Q}^{\bar{a}}_{0,Y|\bar{L}(K)}$.

Parametric submodel:

$$Logit\Big(\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)}(\varepsilon)\Big) = Logit\Big(\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)}\Big) + \varepsilon_{K+1}H_{K+1}$$

where $Logit\Big(\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)}\Big)$ is used as an offset.
Note that the score of this submodel at $\varepsilon_{K+1} = 0$ is

$$H\Big(Y - \bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)}(\varepsilon)\Big)$$

where $H_{K+1} = \frac{I(\bar{A}(K)=\bar{a}(K))}{g_{n,K}}$, i.e., the inverse probability of censoring weights constructed from our estimate of the g-factors of the likelihood. H_{K+1} is called the "clever covariate" because it is constructed so that the above score at $\varepsilon = 0$ spans the $(K+1)^{th}$ component of the efficient influence curve for $\Psi(P_0)$. The parameter estimate $\varepsilon_{n,K+1}$ is computed by minimizing the log-loss, or equivalently, by maximum likelihood estimation.

$$\varepsilon_{n,K+1} = \underset{\varepsilon_{K+1}}{\text{argmin}}\, \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}\Big(\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)}(\varepsilon_{K+1})\Big)(O_i)$$

Targeted update: The targeted update is then computed by plugging in our estimate, $\varepsilon_{n,K+1}$, into the parametric submodel.

$$\bar{Q}^{\bar{a},*}_{n,Y|\bar{L}(K)} = \bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K)}(\varepsilon_{n,K+1})$$

This solves the $(K+1)^{th}$ component of the efficient influence curve estimating equation.

Step K:

Initial Estimator: Now we must construct an initial estimator, $Q^{\bar{a},0}_{n,Y|\bar{L}(K-1)}$, of the iterated conditional expectation. That is the conditional expectation (at time K) of the conditional expectation at time $K+1$ given the history up to time $K-1$. Note that

$$\begin{aligned}
Q^{\bar{a},0}_{n,Y|\bar{L}(K-1)} &= E_n\big[Q^{\bar{a},*}_{n,Y|\bar{L}(K)}\big|\bar{L}^{\bar{a}}(K-1)\big] \\
&= I(L_1(K-1)=1) \\
&\quad + \\
&\quad I(L_1(K-1)=0)E_n\Big[Q^{\bar{a},*}_{n,Y|\bar{L}(K)}\Big|\bar{L}^{\bar{a}}_2(K-1), L^{\bar{a}}_1(K-1)=0\Big]
\end{aligned}$$

Here again the first term reflects our knowledge that if the failure occurred at or before time $K-1$ then event counting process is fixed at value 1 for all subsequent time points with probability 1. We do not need to update this value in the targeted maximum likelihood step. Like the previous step, the second term here corresponds to people who have not yet had the event. Unlike the previous step, however, it is no longer simply the hazard or intensity at time K. It can be thought of as a random variable that consists of probabilities and 1s. This initial estimator could be from a traditional regression of the previously targeted $Q^{\bar{a},*}_{n,Y|\bar{L}(K)}$ fit on the history or some summary measure of the history, or from a Super Learner.

Log-loss function: Here again we use the log-loss function for estimation. Because the outcome $Q^{\bar{a},*}_{n,Y|\bar{L}(K)}$ consists of probabilities and 1s we know that it is always bounded in the interval $[0,1]$. This in turn implies that the log-loss is valid. The log-loss at this step is given by

$$\mathscr{L}(Q^{\bar{a}}_{Y|\bar{L}(K-1)}) = -log\Big\{\big(Q^{\bar{a}}_{Y|\bar{L}(K-1)}\big)^{Q^{\bar{a},*}_{n,Y|\bar{L}(K)}}\big(1-Q^{\bar{a}}_{Y|\bar{L}(K-1)}\big)^{1-Q^{\bar{a},*}_{n,Y|\bar{L}(K-1)}}\Big\}$$

Parametric submodel:

$$Logit\Big(\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K-1)}(\varepsilon)\Big) = Logit\Big(\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K-1)}\Big) + \varepsilon_K H_K$$

where the Logit of our initial estimator, $Logit\Big(\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K-1)}\Big)$, is used as an offset, and $H_K = \frac{I(\bar{A}(K-1)=\bar{a}(K-1))}{g_{K-1}}$, i.e., the inverse probability of censoring weights, is used as the covariate. The parameter estimate $\varepsilon_{n,K}$ is computed by minimizing the log-loss, i.e., by maximum likelihood estimation.

$$\varepsilon_{n,K} = \underset{\varepsilon_K}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}\left(\bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K-1)}(\varepsilon_K)\right)(O_i)$$

Targeted update: The targeted update is then given by

$$\bar{Q}^{\bar{a},*}_{n,Y|\bar{L}(K-1)} = \bar{Q}^{\bar{a},0}_{n,Y|\bar{L}(K-1)}(\varepsilon_{n,K})$$

This solves the $(K)^{th}$ component of the efficient influence curve estimating equation.

Steps $K-1,\ldots,1$: The process for Step K is repeated for every time point, $k = K-1,\ldots,0$.

The Final Step:

The final step is to take the empirical mean of the final conditional expectation over all observations in the data set. Note that this empirical distribution of the pre-intervention variables is already targeted towards the estimation of our parameter, and therefore does not require any update. More specifically, given the empirical distribution as initial estimator the maximum likelihood estimator of the parameter $\varepsilon_{K=0}$ in the parametric submodel is exactly equal to 0. We therefore arrive at the TMLE:

$$\psi^*_n = E_n\left[\bar{Q}^{\bar{a},*}_{n,Y|L(0)}\right] = \frac{1}{n}\sum_{i=1}^{n} \bar{Q}^{\bar{a},*}_{n,Y|L(0)}$$

4.3.2 Statistical properties

The TMLE estimator is a substitution estimator based on targeted fits of functions of Q-factors of the likelihood, $\psi^*_n = \Psi\left(\bar{Q}^{\bar{a},*}_{n,Y|L(0)}, \bar{Q}^{\bar{a},*}_{n,Y|L(1)}, \ldots, \bar{Q}^{\bar{a},*}_{n,Y|L(K)}\right)$. Each k-specific targeted Q-factor solves the score equation for the k-specific parametric sumbodel. The submodel is a generalized linear model, and its score is therefore the "clever" covariate times the residual. Thus, the TMLE, which iteratively solves the score equation of each k-specific submodel, also iteratively solves each k-specific component of the efficient influence curve estimating equation. Upon completion of the algorithm, the TMLE therefore solves the entire influence curve estimating equation, which is the sum over all $(K+1)$ components.

$$\sum_{i=1}^{n} D^*_{\psi^*_n}(O_i) = \sum_{i=1}^{n}\sum_{k=0}^{K} D^*_{\bar{Q}^*_{n,Y|L(k)},g_n}(O_i) = 0$$

This implies that the TMLE is doubly robust in the sense that it will be consistent if either the intervention distribution g_n or the initial estimators of the relevant Q-factors,

$\left(\bar{Q}^0_{n,Y|L(0),\bar{a}}, \bar{Q}^0_{n,Y|L(1),\bar{a}}, \ldots, \bar{Q}^0_{n,Y|L(K),\bar{a}}\right)$, are estimated consistently. If both are estimated consistently then the TMLE, ψ^*_n, is asymptotically linear, which means it has a normal distribution centered at the truth with variance that can be conservatively estimated with

$$\frac{1}{n}\sum_{i=1}^{n}\left(D^*_{\psi^*_n}(O_i)\right)^2$$

When asymptotic linearity cannot be assumed bootstrapping can be used for inference.

4.4 Simulation Results

This section investigates the performance of the TMLE a simulated dataset with a known distribution. Its performance is compared with that of the IPCW estimating equation estimator, and that of the well-known Kaplan-Meier survival curve. The data consisted of fixed covariates W_1, W_2, W_3, W_4 and fixed treatment A_1 drawn from the following distributions:

$$W_1, W_2 \sim \mathcal{N}(0,1), W_3, W_4 \sim Ber(0.5)$$

$$A_1 \sim Ber(expit(0.001W_1 + 0.01W_2 - 0.5W_3 + 0.5W_4 - 0.2W_3W_1))$$

We also included a time-dependent covariate $L_2(t)$ that was allowed to jump from value 0 to 1 once and then stay fixed over the course of follow-up. This covariate along with the censoring indicator $A_2(t)$, and event indicator $L_1(t)$ at each time-point of follow-up were drawn from the distributions implied by the following hazard/intensity functions. Note that we use out of convenience the abuse of notation, $R(t)$, to denote that the subject is in the "risk set" for each counting process at time t.

$$E[L_2(t)|W,A_1,R(t)] = expit(-3 + 0.01W_1 - 0.5A_1 + 0.01t)$$

$$E[A_2(t)|W,A_1,L_2(t),R(t)] = expit(-7 + 0.001W_1 - 0.002W_2 + 0.5W_3 - 3W_4 - 2A_1 + 4L_2 - 0.01t)$$

$$E[L_1(t)|W,A_1,L_2(t-1),R(t)] = expit(-5 + 0.01sin(W_1W_2) + 2W_3 - 3W_4 - A_1 + A_1W_3W_4 + 2L_2 + 0.01t))$$

The data can thus be summarized $O = \big(O(t) : t = 1, \ldots, \tau\big) = \big((W, A_1, L_2(t), A_2(t), L_1(t)) : t = 1, \ldots, \tau\big) \sim P_0$. The parameter of interest is the marginal cumulative event probability at time $t = 20$, $E[L_1(20)]^{\bar{a}}$ under specific interventions, $\bar{a}^0 = (A_1 = 0, A_2(0) = A_2(1) = \ldots = A_2(19) = 0)$ and $\bar{a}^1 = (A_1 = 0, A_2(0) = A_2(1) = \ldots = A_2(19) = 0)$, corresponding respectively to no treatment without censoring and treatment without censoring. The true parameter value was computed using an empirical data set consisting of independent 1,000,000 draws from $P^{\bar{a}}$ under each intervention. A sample of 1,000 observations was then drawn from P_0 and the KM, IPCW, and TMLE estimators were applied. Data simulations and estimators were implemented in SAS software, v9.2.

4.4.1 Performance under positivity violations

Positivity is a requirement for the identification of both statistical and causal effect parameters. As a heuristic example, if a treatment or intervention of interest is absolutely never given to a particular subgroup of patients, perhaps because it is against medical guidelines or the law, then the average causal effect (of treatment) parameter for such patients is nonsensical. Mathematically, a violation of positivity corresponds with a parameter whose efficient influence curve is unbounded. For the parameter considered here it is easy to see that a positivity violation means that we get a D^* comprised of at least one fraction with a 0 in the denominator, which means $D^* = \inf$.

In practice, we often encounter situations data that exhibits near-positivity violations for some data subgroups. For example, in the ATRIA-1 study, persons with a history of bleeding events and falls are sometimes prescribed warfarin a treatment intervention, though this is usually uncommon. While this is not a violation of positivity in the strictest sense, it can have serious consequences for estimation. This is generally known as a practical positivity violation. For estimators whose influence curves have factors that involve the inverse probability of treatment, practical positivity violations are a major concern, because such estimators behave statistically like the empirical mean of their respective influence curves. Thus a very small probability of treatment in the denominator can lead to an outlying value, whose effect on the empirical mean of the influence curve may be likened to the effect of an outlier in the typical estimation of an empirical mean. Both TMLE and IPCW are all estimators of this type, and a careful investigation of their respective influence curves can give insight into their behaviors under practical positivity violations.

Recall that the TMLE influence curve has the same form as the efficient influence curve.

$$
\begin{aligned}
D^{*,TMLE}_{Q^*_n,g_n}(O) = {} & \frac{I(\bar{A}(K) = \bar{a}(K))}{g_{K,n}} \left(L_1(K+1) - \bar{Q}^*_{Y|\bar{L}(K),n}\right) \\
& + \sum_{k=0}^{K-1} \frac{I(\bar{A}(k) = \bar{a}(k))}{g_{k,n}} \left(\bar{Q}^*_{Y|\bar{L}(k+1),n} - \bar{Q}^*_{Y|\bar{L}(k),n}\right) \\
& + \bar{Q}^*_{Y|L(0),n} - \psi_0
\end{aligned}
$$

where g_n is our estimator of the g-factors of the likelihood, and Q^*_n is our updated estimator of the Q-factors.

The IPCW estimator influence curve is

$$
D^{IPCW}_{g_n}(O) = \frac{I(\bar{A}(\tilde{k}) = \bar{a}(\tilde{k}))}{g_{n,\tilde{k}}} \left(L_1(\tilde{k}+1)\right) - \psi_0
$$

where $\tilde{k} = min(K, \underset{k}{\operatorname{argmax}}\, R(k) = 0)$, with the same g_n.

Although the TMLE is efficient when D^* is bounded, its performance relative to the IPCW under positivity violations is not obvious. The reason is that the TMLE influence curve involves a sum of fractions with g_n in the denominator, whereas the IPCW influence curve only has one such fraction. In the TMLE, the numerator of the fraction is always a number in bounded by [-1,1] while the IPCW is exactly equal to 0 or 1.

Table 4.1: Simulation: TMLE performance under positivity violations

Truth	$\psi_0^1 = 0.104$	$\psi_0^0 = 0.217$	$\psi_0^1 - \psi_0^0 = -0.113$
Estimator		MSE (Bias)	
		No positivity violation	
KM	0.0014 (-0.0317)	0.0048 (0.0529)	0.0087 (-0.0846)
IPCW	0.0007 (-0.0011)	0.0002 (-0.0023)	0.0010 (0.0012)
TMLE	0.0006 (-0.0012)	0.0002 (-0.0036)	0.0008 (0.0024)
		Substantial positivity violation	
KM	0.0056 (-0.0741)	0.0236 (0.1504)	0.0515 (-0.2245)
IPCW	0.0036 (-0.0437)	0.0005 (-0.0038)	0.0039 (-0.0400)
TMLE	0.0031 (-0.0318)	0.0005 (-0.0047)	0.0034 (-0.0271)
		Extreme positivity violation	
KM	0.0058 (-0.0755)	0.0263 (0.1580)	0.0558 (-0.2335)
IPCW	0.0055 (-0.0697)	0.0003 (-0.0082)	0.0047 (-0.0615)
TMLE	0.0040 (-0.0499)	0.0003 (-0.0052)	0.0038 (-0.0447)

The simulation study presented here investigates the performance of each estimator under positivity violations, introduced by scaling the linear component of the the treatment assignment mechanism by constant factors, 1, 10, and 20 corresponding, respectively, to ”no”, ”substantial”, and ”extreme” positivity violations. The KM, IPCW, and TMLE were computed for 100 samples of size n=1000. Bias was computed as the difference between the mean of the sample estimates and the true values. Mean squared error was computed in the usual way.

For both the TMLE and IPCW the treatment assignment and censoring mechanisms were estimated under consistent parametric logistic regressions. To reflect common practice, the estimated g-factors at every time-point were truncated to the interval $[0.01, 0.99]$. To carry out the TMLE, we first stratified our data set according to each treatment intervention, and then fit initial estimators of the intervention-specific Q-factors at every time-point with Super Learners. These Super Learners used the baseline covariates, W_1, W_2, W_3, W_4, and the most recent value of the time-dependent covariate, $L_2(t)$, as explanatory variables. The library of candidate estimators contained (1) the unconditional mean; (2) a logistic regression for continuous outcomes in the interval [0,1]; (3) ordinary least squares regression; (4) neural network 3-layer perceptron with two hidden units; and (5) a recursive partitioning decision tree. These candidates were combined with a set of convex weights that minimized the 5-fold cross validated negative Bernoulli loglikelihood risk.

As expected the KM is heavily biased due to the fact that both the treatment and censoring probabilities depend on covariate values. The IPCW performs better than the KM due to the consistent modeling of the g-factors. The TMLE performed the best in terms of MSE across positivity violations of different magnitudes. The MSE of the TMLE was modestly better than that of the IPCW when there was no positivity violation. In this case, the bias of both estimators was less than $\frac{1}{\sqrt{n}}$, which means that it is negligible for inference. The largest TMLE gains relative to the IPCW were observed in the most extreme positivity violation example.

4.4.2 Performance under model misspecification

Model misspecification is a major concern for estimation in nonparametric statistical models. As noted previously the IPCW is consistent in the nonparametric model only if the g-factors of the likelihood are estimated consistently. The TMLE is double robust in that it achieves consistency if either the g-factors or the relevant Q-factors are estimated consistently. The TMLE is also locally efficient in that it achieves the nonparametric efficiency bound when both the g and Q-factors are consistent. This section investigates these properties via the simulation. In order to avoid redundancy with the section on positivity violations above, the linear part of the treatment assignment mechanism was scaled by a factor of 3.

Here, we assess the performance of the IPCW and TMLE estimators built upon parametric estimators of correctly specified g-factors, g_{cor}, or misspecified g-factors, g_{mis}, respectively. The gains in efficiency for TMLE are investigated by comparing three different approaches to the estimation of the Q-factors. The first approach used the NULL model in which each Q-factor $\bar{Q}_{Y|\bar{L}(k),\bar{a}}, k = 0, \dots, K$ was assumed to be independent of the covariates. The second approach estimated the Q-factors with logistic regression functions of the baseline covariates and the most recent time-dependent covariate. The third approach constructed Super Learners for each Q-factor, as in the previous section.

Table 4.2: Simulation: Double robustness of the TMLE

Truth	$\psi_0^1 = 0.104$	$\psi_0^0 = 0.217$	$\psi_0^1 - \psi_0^0 = -0.113$
Estimator		MSE (Bias)	
KM	0.0031 (-0.0544)	0.0131 (0.1064)	0.0272 (-0.1608)
IPCW$_{g_{mis}}$	0.0018 (-0.0360)	0.0011 (0.0287)	0.0050 (0.0647)
IPCW$_{g_{cor}}$	0.0010 (-0.0090)	0.0003 (-0.0030)	0.0012 (-0.0060)
TMLE$_{g_{mis},Q_{NULL}}$	0.0017 (-0.0361)	0.0012 (0.0294)	0.0051 (-0.0655)
TMLE$_{g_{mis},Q_{GLM}}$	0.0008 (-0.0005)	0.0003 (-0.0044)	0.0010 (0.0038)
TMLE$_{g_{mis},Q_{SL}}$	0.0009 (-0.0102)	0.0003 (-0.0035)	0.0012 (-0.0067)
TMLE$_{g_{cor},Q_{NULL}}$	0.0009 (-0.0091)	0.0003 (-0.0031)	0.0011 (-0.0061)
TMLE$_{g_{cor},Q_{GLM}}$	0.0007 (-0.0034)	0.0003 (-0.0047)	0.0009 (0.0013)
TMLE$_{g_{cor},Q_{SL}}$	0.0008 (-0.0052)	0.0003 (-0.0046)	0.0010 (-0.0006)

Again the KM is biased because it assumes independence of both the treatment and censoring processes. In this sense, the KM is in fact an IPCW estimator with a misspecified g. The IPCW is largely unbiased under g_{cor}, but quite biased under g_{mis}. There are several points of interest in comparing the TMLEs presented here. First, the TMLE performance is better than or equal to that of the IPCW and KM in terms of both bias and MSE. The TMLE that uses the NULL model of the Q-factors behaves very similarly to the IPCW, but the benefits of the estimation of the Q-factors are apparent in the table. When the g-factors are misspecified, the double robustness of the TMLE is evidenced by the major bias reduction associated with using GLM or SL fits for the Q-factors. This underscores the importance of estimating the Q-factors when the model for g is unknown.

Why the TMLE that used Super Learners for the Q-fits has an apparently lower MSE than the TMLE that used the GLM (logistic regression model) is not entirely clear. One might think that the Super Learner would perform better in terms of risk on the Q-factors and that this might translate into a smaller MSE for the parameter of interest. However, the properties of the TMLE are driven by the influence curve, and in theory the estimator for the Q-factors should be selected to minimize its variance. In practice, one might choose between different TMLEs indexed by different Q-factor estimators on the basis of the cross validated variance of the influence curve, though in the simulations presented here the observed differences in efficiency were modest at best.

4.5 Application: Causal effect of warfarin

Warfarin is a blood thinner medication used in the prevention of thromboembolic stroke. However, it is well known that its physiological mechanism of action also puts users at higher risk for adverse bleeding events, whose health consequences may be just as devastating as thromboembolic stroke. The efficacy of warfarin has been shown in randomized clinical trials, though some practitioners have questioned whether the extent of the benefits is the same in the general atrial fibrillation population. The reason is that typical patients have more comorbidities and may not be as healthy in general as compared with trial participants. Further, the monitoring for adverse events required of clinical trials may not reflect the normal standards of care in a typical clinical setting. One of the major research objectives in the ATRIA-1 cohort study was to estimate causal effects of warfarin on adverse outcomes in the general atrial fibrillation patient population at Kaiser Permanente Northern California.

Like most drugs, warfarin prescription decisions are based in part on the patient's medical history. In particular those who are thought to be at high risk for stroke are more likely to receive the drug. The risk factors for stroke, however, are largely the same as those for falls, bleeds, and death by any cause. The warfarin prescription decision must therefore weigh the potential benefits of stroke prevention against these potential adverse outcomes. Thus treatment is highly dependent on clinical covariates, and estimation of causal effects must take this into account.

Here we estimate intervention-specific marginal probabilities of stroke of death within a 1-year time frame. The first intervention consists of continuous warfarin usage and no censoring, while the second gives no warfarin therapy and no censoring. The causal effect of warfarin is then defined as the difference in these intervention-specific marginal probabilities.

4.5.1 ATRIA-1 cohort data

Using the same counting process framework outlined in section 4.2.1. $L_1(t)$ will denote the counting process for the stroke or death event at some time-point t, and we are interested in its value at the 1-year (365 days) time-point $Y = L_1(K+1 = 365)$; $L_2(t)$ will denote the (fixed and time-dependent) covariate processes; $A_1 = A_1(t) : t = 0, \ldots, \tau$ will denote the fixed baseline warfarin treatment; and $A_2(t)$ will denote the censoring counting process, where censoring events include disenrollment from Kaiser, end of study, or switching warfarin treatment. We regard the 13,599 as an empirical sample drawn from a probability distribution, $O_1, \ldots, O_{13,559} \sim P_0$, and denote the empirical distribution P_n.

At the beginning of the study, 5,289 subjects were taking warfarin and 8,270 were not. After the baseline assessment, 2,267 subjects started using warfarin and 1,228 stopped using warfarin within the first year. These individuals were censored at the day when the treatment was switched. 168 subjects were censored due to disenrollment from the Kaiser system, and none were administratively censored within the 1-year time frame. Among those who were not censored, 171 had a thromboembolic stroke and 485 died within the first year of follow-up. The remaining 9,240 were observed to be alive, stroke-free, and had not switched treatments at the end of the first year of study. Under the data structure described above the working data set contained 13,559 persons $\times$ 365 days = 4,949,035 rows, though after the timing of censoring and terminal events were taken into account the dataset consisted of 3,915,087 person-days at risk.

4.5.2 Causal effect of warfarin parameter

Our outcome here is defined as stroke or death within a 1-year time-frame, and our parameter of interest is the intervention-specific marginal mean of this outcome: $\psi^{\bar{a}} = \Psi(Q^{\bar{a}}) = E_0[Y^{\bar{a}}]$. Equivalently, this is the marginal cumulative probability of stroke of death within one year, or the proportion of the population who would experience stroke or death within one year under the specified intervention. The fixed interventions of interest are (0) no warfarin for 1-year versus (1) continuous warfarin therapy for 1-year. Both interventions also prevent early censoring. Based on this we may define our additive causal effect parameter as the difference between the two intervention-specific marginal means.

$$\psi^{\bar{a}=1} - \psi^{\bar{a}=0}$$

where

$$(\bar{a} = 1) := (A_1(0) = \ldots = A_1(365) = 1, A_2(0) = \ldots = A_2(365) = 0)$$

$$(\bar{a} = 0) := (A_1(0) = \ldots = A_1(365) = 0, A_2(0) = \ldots = A_2(365) = 0)$$

4.5.3 TMLE implementation

The g-factor corresponding to the fixed treatment assignment mechanism was given by the conditional expectation of warfarin treatment given the medical history at baseline, $g_{A_1,0} = E_0[A_1(0)|L(0)]$. This was estimated with a Super Learner for the negative Bernoulli loglikelihood risk. This Super Learner was fitted on a dataset with 13,559 rows - i.e., one per subject. The g-factor corresponding to the time-dependent censoring mechanism was given by the conditional intensity of censoring (as defined previously) given the medical history up to the most recent time-point, $g_{A_2(t),0} = \left(E_0[A_2(t)|\bar{L}(t),\bar{A}(t-1)] : t = 0,\ldots,\tau\right)$. This was estimated with a Super Learner for the risk defined as the expectation of the sum of the negative Bernoulli loglikelihood loss at each time point. To improve precision we used the entire ATRIA-1 dataset (even time-points beyond day 365) to fit this Super Learner.

The initial fit for the terminal Q-factor corresponding to the time-dependent event intensity mechanism at day 365 was estimated with a Super Learner for the risk defined as the expectation of the sum of the negative Bernoulli loglikelihood at each time point. To improve precision we used the entire ATRIA-1 dataset (even time-points beyond day 365) to fit this Super Learner. The initial fits for the remaining Q-factors were fit with Super Learners for the negative Bernoulli loglikelihood risk. The Super Learner for each time-point specific Q-factor was fit using only those observations that had not yet been censored or experienced the event.

All of the Super Learners (both for the g and Q-factors) used 5-fold cross validation and a library with the following candidates:

- MEAN - the unconditional mean, i.e., the NULL model
- LOGIT or LOGIT_CTS01 - logistic regression or logistic regression for continuous outcomes in [0,1]
- LDA - linear discriminant analysis (not used for the Q-fits with continuous outcomes)
- NN2 - neural network 3-layer perceptron with two hidden units
- TREE - recursive partitioning decision tree

The Super Learners and the TMLE algorithm were implemented in SAS v 9.2, using the SAS macros %SUPERLEARNER and %TMLE_RCS. The code for both marcros are provided in the Appendix.

4.5.4 Results

The Super Learner for the fixed treatment assignment mechanism was a weighted average of the unconditional mean (9%), logistic regression (37%), linear discriminant analysis (12%), neural network (39%), and decision tree (3%). The range of the fitted treatment probabilities was [0.05,0.95], which fortunately, does not suggest a positivity violation with regard to treatment assignment. The Super Learner for the time-dependent censoring mechanism only involved the neural network (6%), and decision tree (94%).

The Super Learner for the event intensity was a weighted average of logistic regression (66%), linear discriminant analysis (4%), neural network (20%), and decision tree (%10). The weights for the Super Learner for remaining Q-factors (364 in total) are too numerous to describe in detail, though generally the higher weight was given to the logistic regression candidate estimator. The initial estimator of the Q-factors for every time point, days 0 to 364, were updated with TMLE to solve the efficient score equation.

The Table below presents the KM, IPCW, and TMLE estimators and estimates of their standard errors based on their respective influence curves. All three estimators yielded point estimates that indicate the continuous warfarin intervention decreased the proportion of stroke or death over the no warfarin intervention. The KM estimator suggests that the marginal stroke probability under the continuous warfarin intervention was 3.5% less than that under the no warfarin interverntion. However, because warfarin treatment decisions are made based on the patient's medical covariate history, the KM is not consistent.

The IPCW and TMLE are more interesting. The range of the estimates of the g-factors used in both TMLE and IPCW indicates that there is no practical positivity violations for the estimation of the parameter of interest. Thus, if our estimates of the treatment assignment and the censoring mechanisms are consistent, then both TMLE and IPCW are consistent for the statistical parameter or interest. Under the additional (untestable) assumption that there are no unmeasured confounders, the IPCW and TMLE also are consistent for the causal effect parameter. The IPCW suggests a reduction of 1.9% whereas the TMLE suggests 2.1%. As expected, the influence curves indicate that the TMLE was somewhat more precise than the IPCW.

It is noteworthy that that TMLE and IPCW estimates, which do adjust for the covariate confounders, suggest that the beneficial effect of warfarin is less than that suggested by the KM which does not adjust. Because warfarin is typically prescribed to sicker patients at a higher risk for stroke, one might anticipate the opposite finding. That is, the unadjusted estimator should make the drug look less beneficial. To help explain this, we must first recall that our event is a combined outcome of "stroke or death (from another cause)" and that the majority of the events are the latter. The TMLE estimate of the marginal event probability under the warfarin intervention is much higher than that of the KM. This means that the people who were not on warfarin or those who were on warfarin but were censored would have had a higher "stroke or death" event probability than those who were actually on warfarin and were not censored. One could imagine this to be the case for people who were not taking warfarin due to a fall or a serious bleeding event. Further, the TMLE estimate for the no warfarin intervention was lower than the KM, which suggests that warfarin was withheld from persons who had a higher underlying risk of "stroke or death."

We conducted an informal test of this by regressing our Super Learner fit for the "stroke or death" event intensity on the baseline warfarin treatment. As expected based on the above findings, a higher intensity was associated with a lower probability of receiving warfarin. We also regressed the Super Learner fit for the event intensity on the censoring mechanism separately for those who started out on warfarin and those who did not. Amongst persons whos tarted the study on warfarin,

Table 4.3: Causal effect of warfarin on the probability of stroke or death within 1 year

Estimator	Warfarin	No warfarin	Difference	S.E.	p-value
KM	0.0398	0.0751	-0.0352	0.0046	0.0001
IPCW	0.0490	0.0680	-0.0190	0.0066	0.0020
TMLE	0.0520	0.0731	-0.0210	0.0065	0.0006

higher event intensities were positively associated with censoring, which suggests that when a patient's underlying risk of ”stroke or death” increased, they were taken off warfarin. Conversely, increased risk of ”stroke or death” was negative associated with starting warfarin treatment.

In summary, an intervention defined by giving warfarin to all atrial fibrillation patients significantly reduced the marginal probability of stroke or death by 2% relative to the null intervention (no warfarin at all). This effect estimate was smaller than the estimate that ignored the confounding, seemingly introduced by clinician's apparent reluctance to prescribe warfarin to patients whose medical history implied a higher underlying risk of death.

4.6 Discussion

The estimation of causal effects of medical therapies remains one of the most important practical areas of scientific research. Even the best planned randomized controlled trials can generate data structures that violate assumptions required for the consistency of traditional estimators. Influence curve-based estimators often allow consistent estimation in the face of such violations in RCTs and also extends the analysis of causal effects to the observational data setting. The latter is remarkable in that it allows the estimation of causal effects directly from the patient populations in which therapies are typically prescribed, as opposed to trial populations that tend to comprised of healthier subjects.

Increasing awareness of causal effect estimators, e.g., TMLE or IPCW, has led to academic debate over which estimators are more useful in theory and practice. The TMLE combines the most attractive theoretical features of the estimating equation approaches, e.g., IPCW, and maximum likelihood estimation, and avoids several of their shortcomings. Like the IPCW, the TMLE is an example of influence curve-based estimation. The key distinction is that the IPCW requires that the parameter of interest to be defined as the solution to an influence curve-based estimating equation. This is not always possible, and even when it is, the estimating equation may have multiple solutions without a clearcut optimality criterion. Further, inconsistent estimation in IPCW can lead to parameter estimates that are outside the statistical model. For example, it is not uncommon to see IPCW confidence intervals for probabilities that lie outside the range $[0,1]$. Because the TMLE is a substitution estimator, parameter estimates never escape the bounds of the statistical model. The parameter is defined as mapping applied to the probability distribution of the data and therefore extends to cases where it cannot be written as the solution to an estimating equation.

The maximum likelihood principle provides a clearcut theoretical optimality criterion for use in practical estimation problems.

In this article we implemented a TMLE algorithm (van der Laan and Gruber, 2011) that can be applied in either the RCT or observational scenarios to estimate the marginal intervention-specific cumulative event probability for right-censored survival data with fixed or time-dependent covariates in a nonparametric statistical model. The TMLE is a substitution estimator defined as a mapping applied to a series of iterated conditional expectations, one for every time-point, of the intervention-specific outcome given the data history. It solves the efficient influence curve estimating equation for the marginal intervention-specific cumulative event probability and is doubly robust in the sense that it is consistent if either the intervention distribution or the relevant components of of the event process distribution are estimated consistently. If both are estimated consistently, it achieves the efficiency bound. This TMLE algorithm represents several improvements over previously published TMLE methodology (Stitleman and van der Laan, 2010). It is in a sense *more targeted* because it requires only estimation of a series of conditional expectations, and not the entire density of the event and covariate processes.

We compared the practical performance of the TMLE with the KM and IPCW estimators on a simulated data structure under both positivity violations and model misspecifications. Finally we compared the performance of the TMLE with that of KM and IPCW in the estimation of the causal effect of warfarin therapy on the marginal probability of stroke or death by 1 year, using real-world data on 13,559 individuals with atrial fibrillation. The results of the analyses indicated that a medical policy of treating all patients with warfarin versus no treatment would have reduced 1-year stroke or death probability from 7% to 5%. While this causal effect parameter provides a simple illustration of the TMLE methodology presented in this article. It should be noted, however, that such interventions, i.e., continuous warfarin for all patients or no warfarin for any, are unrealistic in practice. Future work applied work in this area should focus on comparisons of realistic treatment rules, for example, dynamic treatment rules that respond to changes in covariate values in the patient's medical history. The methods presented here are general and may be adapted to estimate the causal effects of such dynamic treatment rules, and this is certainly an area of application where further research is needed.

Bibliography

H Bang and J M Robins. Doubly robust estimation in missing data and causal inference models. *Biometrics*, 61:962–972, 2005. doi: 10.1111/j.1541-0420.2005.00377.x.

P J Bickel, C A J Klaassen, Y Ritov, and J Wellner. *Efficient and Adaptive Estimation for Semiparametric Models*. Springer-Verlag, 1993.

Leo Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996.

Paul Chaffee, Alan E Hubbard, and Mark J van der Laan. Permutation-based pathway testing using the super learner algorithm. Technical Report 263, Division of Biostatistics, University of California, March 2010. URL `http://www.bepress.com/ucbbiostat/paper263`.

D R Cox and D Oakes. *Analysis of Survival Data*. Chapman-Hall, 1984.

Ivan Diaz Munoz and Mark J van der Laan. Super learner based conditional density estimation with application to marginal structural models. *The International Journal of Biostatistics*, 7, 2011. doi: 10.2202/1557-4679.1356. URL `http://www.bepress.com/ijb/vol7/iss1/38`.

Susan Gruber and Mark J van der Laan. An application of collaborative targeted maximum likelihood estimation in causal inference and genomics. *International Journal of Biostatistics*, 6, 2010. doi: 10.2202/1557-4679.1182. URL `http://www.bepress.com/ijb/vol6/iss1/18`.

F E Harrell Jr., K L Lee, and D B Mark. Tutorial in biostatistics. multivariable prognostic models: Issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, 15:361–387, 1996.

D W Hosmer, T Hosmer, S Le Cessie, and S Lemeshow. A comparison of goodness-of-fit tests for the logistic regression model. *Statistics in Medicine*, 16:965–980, 1997. doi: 10.1002/(SICI)1097-0258(19970515)16:9⟨965::AID-SIM509⟩3.0.CO;2-O.

E L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481, 1958.

J Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

Eric Polley and Mark van der Laan. *SuperLearner: Super Learner Prediction*, 2011. URL `http://CRAN.R-project.org/package=SuperLearner`. R package version 2.0-4.

Eric C Polley and Mark J van der Laan. Super learner in prediction. Technical Report 266, Division of Biostatistics, University of California, May 2010. URL `http://www.bepress.com/ucbbiostat/paper266`.

Kristin E Porter, Susan Gruber, Mark J van der Laan, and Jasjeet S Sekhon. The relative performance of targeted maximum likelihood estimators. *International Journal of Biostatistics*, 7, 2011. doi: 10.2202/1557-4679.1308. URL `http://www.bepress.com/ijb/vol7/iss1/31`.

Sandra E Sinisi, Eric C Polley, Maya L Petersen, Soo-Yon Rhee, and Mark J van der Laan. Super learning: An application to the prediction of hiv-1 drug resistance. *Statistical Applications in Genetics and Molecular Biology*, 6, 2007. doi: 10.2202/1544-6115.1240. URL `http://www.bepress.com/sagmb/vol6/iss1/art7`.

Ewout W Steyerberg, Gerard J J M Borsboom, Hans C van Houwelingen, Marinus J C Eijkemans, and J Dik F Habbema. Validation and updating of predictive logistic regression models: a study on sample size and shrinkage. *Statistics in Medicine*, 23:2567–2586, 2004. doi: 10.1002/sim.1844. URL `http://onlinelibrary.wiley.com/doi/10.1002/sim.1844/pdf`.

Ori M Stitleman and Mark J van der Laan. Collaborative targeted maximum likelihood for time to event data. *International Journal of Biostatistics*, 6, 2010. doi: 10.2202/1557-4679.1249. URL `http://www.bepress.com/ijb/vol6/iss1/21`.

Ori M Stitleman, C William Wester, Victor De Gruttola, and Mark J van der Laan. Targeted maximum likelihood estimation of effect modification parameters in survival analysis. *International Journal of Biostatistics*, 7, 2011. doi: 10.2202/1557-4679.1307. URL `http://www.bepress.com/ijb/vol7/iss1/19`.

Anastasios A Tsiatis. A note on a goodness-of-fit test for the logistic regression model. *Biometrika*, 67:250–251, 1980. doi: 10.1093/biomet/67.1.250. URL `http://biomet.oxfordjournals.org/content/67/1/250.full.pdf+html`.

Catherine Tuglus and Mark J van der Laan. Repeated measures semiparametric regression using targeted maximum likelihood methodology with application to transcription factor activity discovery. *Statistical Applications in Genetics and Molecular Biology*, 10, 2011. doi: 10.2202/1544-6115.1553. URL `http://www.bepress.com/sagmb/vol10/iss1/art2`.

Mark J van der Laan. Estimation based on case-control designs with known prevalence probability. *International Journal of Biostatistics*, 4, 2008. doi: 10.2202/1557-4679.1114. URL `http://www.bepress.com/ijb/vol4/iss1/17`.

Mark J van der Laan. Targeted maximum likelihood based causal inference part i. *International Journal of Biostatistics*, 6, 2010a. doi: 10.2202/1557-4679.1211. URL `http://www.bepress.com/ijb/vol6/iss2/2`.

Mark J van der Laan. Targeted maximum likelihood based causal inference part ii. *International Journal of Biostatistics*, 6, 2010b. doi: 10.2202/1557-4679.1241. URL `http://www.bepress.com/ijb/vol6/iss2/3`.

Mark J van der Laan and Sandrine Dudoit. Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples. Technical report, Division of Biostatistics, University of California, November 2003.

Mark J van der Laan and Susan Gruber. Collaborative double robust targeted maximum likelihood estimation. *International Journal of Biostatistics*, 6, 2010. doi: 10.2202/1557-4679.1181. URL `http://www.bepress.com/ijb/vol6/iss1/17`.

Mark J van der Laan and Susan Gruber. Targeted minimum loss based estimation of an intervention specific mean outcome. Technical Report 290, Division of Biostatistics, University of California, 2011.

Mark J van der Laan and J M Robins. *Unified Methods for Censored Longitudinal Data and Causality*. Springer-Verlag, 2003.

Mark J van der Laan and Sherri Rose. *Targeted Learning: Causal Inference for Observational and Experimental Data*. Springer, 2011.

Mark J van der Laan and Daniel Rubin. Targeted maximum likelihood learning. *International Journal of Biostatistics*, 2, 2006. doi: 10.2202/1557-4679.1043. URL `http://www.bepress.com/ijb/vol2/iss1/11`.

Mark J van der Laan, Sandrine Dudoit, and Sunduz Keles. Asymptotic optimality of likelihood-based cross-validation. *Statistical Applications in Genetics and Molecular Biology*, 3, 2004. doi: 10.2202/1544-6115.1036. URL `http://www.bepress.com/sagmb/vol3/iss1/art4`.

Mark J van der Laan, Eric C Polley, and Alan E Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6, 2007. doi: 10.2202/1544-6115.1309. URL `http://www.bepress.com/sagmb/vol6/iss1/art25`.

S Vinterbo and L Ohno-Machado. A recalibration method for predictive models with dichotomous outcomes. In *Predictive Models in Medicine: Some Methods for Construction and Adaptation*. Norwegian University of Science and Technology, 1999. URL `pubs/self/Vinterbo1999-PhD.pdf`. ISBN 82-7984-011-7, ISSN 0802-6394.

D H Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

Appendix A

%SUPERLEARNER: A SAS macro for data-adaptive machine learning

A.1 Introduction

The Super Learner is a general approach to data-adaptive machine learning estimation of a parameter, $Q_0 = Q(P_0) = \underset{Q}{\operatorname{argmin}}\, R_{P_0}(Q)$, defined as a risk minimizer in a nonparametric statistical model. In brief, the Super Learner is an ensemble estimator that combines the outputs from a collection of K user-supplied candidate estimators (the library), $\hat{Q}_{n,1},\ldots,\hat{Q}_{n,K}$, such that the resulting ensemble minimizes the cross validated risk. The library may include fully parametric estimators such as generalized linear regressions and/or data-adaptive machine learning algorithms such as decision trees, artificial neural networks, gradient boosting machines, etc. In the context of linear risk functions, defined as the expectation of a loss function on the unit data structure, van der Laan et al. (2007) proposed extending the library of candidate estimators to an entire class consisting of convex combinations of the original candidates

$$\hat{Q}_{n,\alpha} = m\big(\hat{Q}_{n,1},\ldots,\hat{Q}_{n,K}|\alpha\big)$$

where $\alpha = (\alpha_1,\ldots,\alpha_K), \sum_{k=1}^{K}\alpha_k = 1, \alpha_k \geq 0 : k = 1,\ldots,K$. They defined the convex Super Learner as the candidate in this extended library indexed by the convex weights α_n that minimized the V-fold cross validated risk, and proposed an algorithm to estimate these convex weights. This convex Super Learner has been proven under mild regularity conditions to perform asymptotically as well as the "oracle" estimator, which is defined as the convex combination of the estimators in the user-supplied library that minimizes the true risk (van der Laan et al., 2007). Typically, the Super Learner is applied to prediction, i.e., estimation of the conditional expectation function, or density estimation. While prediction and density estimation are of interest in their own right, they also play a fundamental role in the estimation of interesting finite dimensional parameters, e.g., marginal intervention-specific means and additive effects.

A convex Super Learner algorithm has been implemented in the R programming language under the package SuperLearner (Polley and van der Laan, 2011), and has been shown to perform well for estimation of conditional expectation function (Polley and van der Laan, 2010) and more recently for the estimation of the conditional density function (Diaz Munoz and van der Laan, 2011). The R Super Learner implementation has been applied to prediction problems in genetics (Sinisi et al., 2007), has been used as a component in robust hypothesis testing (Chaffee et al., 2010), and has been increasingly used as a component in estimation of causal effects (Stitleman and van der Laan, 2010; van der Laan and Gruber, 2010; Gruber and van der Laan, 2010; Tuglus and van der Laan, 2011; Diaz Munoz and van der Laan, 2011). Procedurally similar algorithms for regression and classification have also been described in the machine learning literature under the names stacking or stacked generalization (Wolpert, 1992; Breiman, 1996). There are several Weka implementations of stacking, which could also be used to implement a Super Learner. Both the R SuperLearner package and the Weka stacking implementations require all data to be loaded into a computer's working (RAM) memory. This can prove to be a major obstacle for the implementation of a Super Learner for large data sets on standard personal computers.

This Appendix describes %SUPERLEARNER, a new Super Learner implementation in SAS (Statistical Analysis System) macro. It is intended as an alternative geared towards SAS users, that allows scalable Super Learners for large data sets to be constructed on computers with relatively modest RAM memory resources. The %SUPERLEARNER macro requires SAS/BASE, SAS/STAT, and SAS/OR modules, and candidate estimators may be constructed under a broad unified programming framework.

A.2 Big data and SAS

As costs for data collection and storage have decreased, data sets have become larger. The increases in data size include increased numbers of covariates (or features) per observation and increased numbers of observations themselves. In time-dependent data, we may increase the size of our dataset by sampling the time-dependent process at finer and finer time intervals to allow identification of very low latency effects. While increasing data size has obvious theoretical benefits, large data sets are relatively useless without appropriate practical tools. Because of this, ”big data” computing and analysis has become an important topic over the last few years. ”Big data” generally refers to any data set that is too large to be analyzed in a given computer's RAM memory, and therefore big data for a particular machine may not be so ”big” for another. The common thread underlying all big data solutions is a sense of scalability that is not limited by fixed RAM resources.

The SAS software platform is unique amongst the major statistical programming languages in that it processes data, organized into a matrix, row-by-row through fast sequential reads and writes to the hard-disk. This framework, which yields efficient and scalable processing of big data, allows SAS to serve as both a database management tool and data analysis engine. This likely contributed to SAS having become a de facto standard data tool in many research organizations over the last several decades. While the SAS framework clearly eliminates RAM memory limitations, the sequential disk IO (input/output), however efficient, can represent a major bottleneck for speed in SAS applications involving big data. Thus, ”good” SAS programming for big data centers on minimizing disk IO wherever possible, and we have tried to incorporate these practices in the %SUPERLEARNER macro presented here.

A.3 %SUPERLEARNER procedural overview

%SUPERLEARNER is a SAS macro that takes as input a SAS dataset containing the observed sample and constructs a convex Super Learner (as described previously) based on a user-supplied library of candidate estimators, a choice of squared error or negative loglikelihood-based risk function, and number of folds for V-fold cross validation. The user-supplied candidate estimators in the library are themselves SAS macros that take as input a SAS dataset with training data and output a .sas program that contains DATA STEP code that maps explanatory variables into candidate estimates. The library here corresponds to a an actual disk directory that contains each candidate estimator macro as a separate .sas program.

The algorithm begins by sequentially constructing V-fold cross validated fits for each candidate estimator. The fits are saved to disk as .sas programs indexed by the candidate estimator name and validation fold number under a user-supplied working directory. A single DATA STEP then calls the fits to generate cross validated estimates, which appended as new variables to the original input data set. The cross validated estimates are then fed to a constrained optimization procedure in PROC NLP to estimate the set of convex weights, α_n, that, when applied to the estimates, minimizes the cross validated risk. For single time-point data structures the cross validated risk is simply the mean of the usual cross validated squared error of negative Bernoulli loglikelihood losses for each row corresponding to an independent subject. For data structures with multiple rows per subject, e.g., time-to-event data, the loss is taken to be the sum loss over all the subject-specific observations. The optimization is initialized at the regular cross validation selector that applies a weight of 1 to the best single candidate and 0 to the rest. The optimization technique is left to the SAS default. The resulting convex weight estimates are saved to a SAS data set.

Each candidate estimator in the library is then fit on the SAS data set containing the entire training data sample, and again the fits are saved to disk under the user-supplied working directory as .sas programs with SAS DATA STEP code to implement the fit mappings. At this point, the convex Super Learner fit is fully defined and is saved to disk under the user-supplied working directory as the file F_SuperLearner.sas. This file contains SAS DATA STEP code (1) to generate estimates from each candidate (induced by the entire training sample) in the library, and then (2) to construct the weighted combination of these indicated by α_n.

Upon completion, the %SUPERLEANER macro returns the original input training SAS data set with new column variables corresponding estimates from each of the candidates in the library. By default, the Super Learner prints to the SAS listing device the estimated convex weights α_n, the cross validated risk of the individual candidate estimators and the risk on the validation dataset, if supplied by the user. The inputs to the %SUPERLEARNER macro are saved as a SAS dataset. The user may also choose whether to print to the SAS listing device all normally printed output during the fitting of candidate estimators and the convex weights.

A.4 Input parameters

```
%SUPERLEARNER(TRAIN=, TEST=, Y=, Y_TYPE=, X=, ID=, T=, WEIGHTS=,
              SL_LIBRARY=, LIBRARY_DIR=, EnterpriseMiner=,
              LOWER_BOUND=, UPPER_BOUND=,
              LOSS=L2, V=10, FOLD=, SEED=510, VERBOSE=F, WD=);
```

- **TRAIN**: A SAS data set with training sample data.
- **TEST**: An optional SAS data set containing validation data. This data set is not used in the fitting the Super Learner.
- **Y**: The target variable. This is univariate and may be either continuous or binary valued. Missing values are not allowed.
- **Y_TYPE**: The target variable type. Takes values CTS or BIN.
- **X**: The covariates or explanatory variables. All variables must be numeric interval or binary/categorical class variables. Missing values treated in accordance with candidate estimators.
- **ID**: Unique identifier for each independent unit of data. Missing values are not allowed.
- **T**: Variable to indicate time in a counting process framework, including time-to-event.
- **WEIGHTS**: Observation weights used in computing the loss function.
- **SL_LIBRARY**: The names of the candidate estimators to include in the Super Learner library.
- **LIBRARY_DIR**: The file path of the directory that contains the SAS macros for each candidate.
- **EnterpriseMiner**: A T/F indicator of whether to construct a SAS datamining catalog.
- **LOWER_BOUND**: All estimates less than LOWER_BOUND are truncated at this value.
- **UPPER_BOUND**: All estimates greater than UPPER_BOUND are truncated at this value.

- **LOSS**: The loss function. Takes value L2 for squared error or LOG for -2 times the Bernoulli loglikelihood losses. If the data has multiple multiple rows per subject, the sum loss over all subject-specific observations.
- **V**: The number of folds for V-fold cross validation on TRAIN data set.
- **FOLD**: Name of SAS variable with fold assignment.
- **SEED**: A numeric seed value for random number generators throughout the %SUPERLEARNER macro. This must be specified for results to be reproducible. Default is 510. Missing value is not allowed.
- **VERBOSE**: A ”T/F” value indicating whether SAS should output all standard output to the listing device.
- **WD**: The working directory. All estimator fits, convex weight estimates, and general Super Learner information will be saved to this directory.

A.5 Candidate estimator macros

Each user-supplied candidate estimator is written as a SAS macro constructed according to a general template. The candidate estimator macro must take as input a SAS dataset containing training data that contains the target variable Y and explanatory variables X. The resulting fit must be a mapping that is written to a .sas program as DATA STEP code.

```
%CANDIDATE(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);
```

- **TRAIN**: Data used for training the candidate fit
- **Y**: The target variable. This is univariate and may either be continuous or binary valued. Missing values are not allowed.
- **Y_TYPE**: The type of target variable. Takes values CTS or BIN.
- **X**: The covariates or explanatory variables. These must be numeric interval or binary/categorical class variables. Missing values may require special attention.
- **ID**: Unique identifier for each independent unit of data. Missing values are not allowed.
- **T**: Variable to indicate time in a counting process framework, including time-to-event.
- **WEIGHTS**: Observation weights used in fitting.
- **WD**: The working directory in which to save the candidate fit.

The values of the inputs should typically be left undefined in the macro definition, as they will be filled in appropriately by the core %SUPERLEARNER macro. For candidate estimators that involve random number generation, the user may wish construct several similar candidate estimators indexed by different fixed SEED values. The explicit use of Y_TYPE, ID, T, WEIGHTS, or SEED, is not required inside the macro definition, though they may be useful for certain candidate estimators. Note that the candidate estimator may include any number of intermediary DATA STEPs or PROC STEPs, but the fitted mapping must be saved as a .sas file (to the working directory WD) with code that can be executed within a single SAS DATA STEP.

Example candidate estimator: Logistic regression

The candidate estimator macro %LOGIT fits an additive logistic regression candidate estimator using PROC LOGISTIC. The fitted estimator mapping is then coded as a series of DATA STEP commands and written to the file F_LOGIT.sas in the specified working directory WD. This may be used to generate predictions for any independent SAS dataset by using the %include functionality in a subsequent DATA STEP.

```
%MACRO LOGIT(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

*******************************;
* DESCRIPTIVE TITLE STATEMENTS *;
*******************************;
title3 "LOGIT";
title4 "FITTING: LOGISTIC REGRESSION";
title5 "VARIABLE SELECTION: ALL";

**************************;
* CONSTRUCT ESTIMATOR FIT *;
**************************;
proc logistic data=&TRAIN descending;
 model &Y = &X;
 %IF &WEIGHTS ne %THEN %DO; weight &WEIGHTS; %END;
 ods output ParameterEstimates=_MyCoef;
run;

*********************;
* SAVE ESTIMATOR FIT *;
*********************;
data _null_;
 set _MyCoef end=eof;
 file "&WD\F_LOGIT.sas";
 if _n_ = 1 then put "p_LOGIT = 1/(1+exp(-1*(";
 if Variable="Intercept" then put Estimate;
 else put "+" Estimate "*" Variable ;
 if eof then put ")));" ;
run;
```

```
********************;
* CLEAN WORK SPACE *;
********************;
proc datasets lib=work; delete _: ; quit;

%MEND LOGIT;
```

While this example is rather simple, further user-supplied candidate estimators may be quite complex and include any number of DATA STEPs or PROCs. The unifying feature of the candidate estimators is that they are all functions that take as input a SAS dataset containing training data and output a .sas program file with instructions to map explanatory variables into a an estimate. This is a general and extensible framework that can include PROCs from several different SAS modules or potentially software outside of SAS itself. As a general rule it is a good idea to include a clean-up step, e.g., PROC DATASETS, to delete any temporary SAS datasets that were constructed during the fitting process.

For users who are not comfortable with SAS macro programming, several examples of candidate estimators, which might best be described as a ”default” library are included in the Appendix of this dissertation. These candidates include logistic regression, linear discriminant analysis, least squares regression, LASSO regression, and stepwise variable selection regression routines. From *SAS/EnterpriseMiner* we include a recursive partitioning decision tree, a boosting algorithm, and an artificial neural network with a multilayer perceptron architecture. Users are encouraged to supply their own candidates to build larger libraries with potentially more interesting candidates.

A.6 Printed output, SAS datasets, and estimator fits

Standard output to the SAS listing device is turned off by the default setting VERBOSE=F. If VERBOSE=T, then all normally produced output from the the candidate estimators will be printed to the listing device. Upon completion, the %SUPERLEARNER macro prints the the convex weights applied to the estimators in the library, the V-fold cross validated risk of each candidate in the library, and the risk computed for the validation data if one is provided. The input data sets TRAIN and TEST, are augmented with estimates from each candidate int he library. The TEST data set is also augmented with the Super Learner estimates. The candidate estimator fit mappings and the Super Learner fit mapping are all saved to the hard disk as .sas program files under the user-supplied working directory. These files can be used to compute estimates for an independent SAS data set by using the %include macro function.

A.7 Simulation: Conditional expectation of a continuous outcome

The simulation illustrates the Super Learners ability to learn a reasonably complex univariate function based on a relatively large training sample. Consider the data structure $O = (Y,W) \sim P_0 \in \mathscr{M}$, where $W \in \mathbb{R}$ and $Y \in \mathbb{R}$ are both univariate. Our parameter of interest is $\bar{Q}_{Y,0} = \bar{Q}_Y(P_0) = E_0[Y|W]$, the conditional expectation function. The squared error loss function, $\mathscr{L}(\bar{Q}_Y)(O) = (\bar{Q}_Y(W) - Y)^2$ is valid for this parameter in that $\bar{Q}_{Y,0} = \underset{\bar{Q}_Y}{\operatorname{argmin}}\, E_0[\mathscr{L}(\bar{Q}_Y)(O)]$.
The distribution of the data was $W \sim Unif[0,1]$ and $Y \sim \mathscr{N}(\bar{Q}_{Y,0}(W),1)$, where

$$
\begin{aligned}
\bar{Q}_{Y,0} = \quad & I(W < 0.05)10sin(\frac{W}{0.01}) \\
& -5I(0.05 <= W < 0.1)(W+3)^{3/2} \\
& +I(0.1 <= W < 0.2)(2W - 2W^2) \\
& +I(0.2 <= W < 0.3)(30W + 60) \\
& -I(0.3 \leq W < 0.5)(15W - 0.45) \\
& -I(0.5 \leq W)cos(5W)(W+2)^3
\end{aligned}
$$

The training data was comprised of 100,000 observations. A Super Learner for the minimizer of the squared error risk was constructed using 10-fold cross validation. The Super Learner considered as candidates the family of convex weighted combinations of the following library of candidates:

MEAN Unconditional mean outcome, i.e., the NULL model

OLS Ordinary least squares additive regression

LASSO LASSO additive regression, regularization selected with cross validation

NN2 Neural network, a 3-layer perceptron architecture with 2 hidden nodes

TREE Decision tree, recursive partitioning based on significance of the F-statistic p < 0.2

BOOST50 Gradient boosting machine (treeboost), 50 iterations with shrinkage factor 0.2

To increase the flexibility of the candidates, the following transformations of W were included as explanatory variables: W_sq = W^2, W_cu = W^3, W_sin = $sin(W)$, W_cos = $cos(W)$, and W_inv = $\frac{1}{W}$. The squared error (validation) risk was evaluated on a large validation data set comprised of 10,000,000 independent observations. The validation risk for the true conditional expectation function was 1.

```
%SUPERLEARNER(TRAIN=train, TEST=valid,
              Y=Y, Y_TYPE=CTS, X=W W_sq W_cu W_sin W_cos W_inv,
              ID=ID, T=, WEIGHTS=,
              SL_LIBRARY=MEAN OLS LASSO NN2 TREE BOOST50,
              LIBRARY_DIR=C:\test\SuperLearner\Library,
              EnterpriseMiner=T, LOWER_BOUND=, UPPER_BOUND=,
              LOSS=L2, V=10, SEED=510, VERBOSE=F,
              WD=C:\test\SuperLearner);
```

Table A.1: Simulation 1: SAS Super Learner for squared error risk

Estimator	10-fold CV risk	α_n	Validation risk
Super Learner	—–	—-	1.32
Unconditional mean (null model)	147.82	0.00	147.93
Ordinary least squares regression	82.27	0.00	109.30
Lasso regression	86.03	0.00	99.08
Artificial neural network	30.69	0.00	30.17
Decision tree	1.42	0.51	1.48
Gradient boosting	1.43	0.49	1.44

A.8 Simulation: Conditional expectation of a binary outcome

This simulation uses a smaller sample size but with more explanatory covariates in the setting of a binary outcome. Consider the data structure $O = (Y,W) \sim P_0 \in \mathscr{M}$, where $Win\mathbb{R}^d$ and $Y \in \{0,1\}$. Our parameter of interest is again the conditional expectation function $\bar{Q}_{Y,0} = \bar{Q}_Y(P_0) = E_0[Y|W] = P_0[Y=1|W]$. The negative Bernoulli loglikelihood loss function, $\mathscr{L}(\bar{Q}_Y)(O) = -2log\big(\bar{Q}_Y(W)^Y(1-\bar{Q}_Y(W))^{1-Y}\big)$ is valid for this parameter in that $\bar{Q}_{Y,0} = \underset{\bar{Q}_Y}{\text{argmin}}\, R_0[\bar{Q}_Y(O)] = \underset{\bar{Q}_Y}{\text{argmin}}\, E_0[\mathscr{L}(\bar{Q}_Y)(O)]$.

The distribution of the data was $W_1,\ldots,W_{20} \sim Ber(0.5)$, and $W_{21},\ldots,W_{40} \sim \mathscr{N}(0,1)$, with $Y \sim Ber(\bar{Q}_{Y,0}(W))$, where $\bar{Q}_{Y,0} = \big(1+e^{-(W_3+5W_{25}+0.2sin(W_{32}W_2)+W_4W_5+W_{40}^2)}\big)^{-1}$. The training set was comprised of 1,000 observations. A Super Learner for the minimizer of the negative Bernoulli loglikelihood risk was constructed using 10-fold cross validation. The Super Learner considered as candidates the family of convex weighted combinations of the logit transform of the following candidates:

MEAN Unconditional mean outcome, i.e., the NULL model

LOGIT Additive logistic regression

LDA Linear discriminant analysis

NN2 Neural network, a 3-layer perceptron architecture with 2 hidden nodes

TREE Decision tree, recursive partitioning based on significance of the F-statistic $p < 0.2$

BOOST50 Gradient boosting machine (treeboost), 50 iterations with shrinkage factor 0.2

The training data was comprised of 1,000 observations. The true risk was evaluated on a large validation data set comprised of 10,000,000 observations. The risk of the true conditional expectation function evaluated on the validation data was 0.447.

```
%SUPERLEARNER(TRAIN=train, TEST=valid,
              Y=Y, Y_TYPE=BIN,
              X=W1  W2  W3  W4  W5  W6  W7  W8  W9 W10
                W11 W12 W13 W14 W15 W16 W17 W18 W19 W20
                W21 W22 W23 W24 W25 W26 W27 W28 W29 W30
                W31 W32 W33 W34 W35 W36 W37 W38 W39 W40,
              ID=ID, T=, WEIGHTS=,
              SL_LIBRARY=MEAN LOGIT LDA NN2 TREE BOOST50,
              LIBRARY_DIR=C:\test\SuperLearner\Library,
              EnterpriseMiner=T, LOWER_BOUND=, UPPER_BOUND=,
              LOSS=LOG, V=10, SEED=510, VERBOSE=F,
              WD=C:\test\SuperLearner);
```

Table A.2: Simulation 2: Super Learner for negative Bernoulli loglikelihood risk

Estimator	10-fold CV risk	α_n	Validation risk
Super Learner	—–	—-	0.570
Unconditional mean (null model)	1.334	0.24	1.326
Linear discriminant analysis	0.614	0.09	0.603
Logistic regression	0.619	0.46	0.602
Artificial neural network	1.086	0.02	1.256
Decision tree	1.008	0.03	0.708
Gradient boosting	0.691	0.16	0.647

A.9 Discussion

In this article we presented a scalable SAS macro implementation of the convex Super Learner algorithm for parameters defined as the minimizers of a squared error or negative Bernoulli log likelihood risk functions. The implementation uses a V-fold cross validation scheme and additive convex weights to combine candidate estimators in a user-supplied library. The %SUPERLEARNER macro was demonstrated and shown to perform well on toy examples involving simulated data in the point-treatment setting and in observational longitudinal data as presented in Chapter 2, where it was used to estimate the stroke intensity function for persons with atrial fibrillation.

We hope this software provides a relatively easy to use tool that allows non-technical SAS users to implement Super Learners for prediction or density estimation. The %SUPERLEARNER macro represents one particular framework for implementing a Super Learner in SAS. The framework, which involves saving the estimator fits (including the SuperLearner.sas) to disk has many benefits. First, it provides a simple unified framework for saving fits and generating estimates on independent datasets. Estimates from multiple candidates and the Super Learner may all be constructed in a single DATA STEP, which significantly reduces disk IO as compared with native PROCs. The fits saved as .sas program files are essentially text files, and therefore tend to require relatively little hard-disk space in comparison with SAS outmodel data sets. Using only DATA STEP code for generating predictions obviates the need for installation of potentially more expensive SAS modules on machines dedicated solely to prediction or scoring for new observations; thus potentially saving organizations both time and money. The DATA STEP code makes the actual Super Learner estimator mapping an explicit transparent set of instructions, which may help to explain results to non-technical stakeholders who may be wary of a black-box machine learning prediction engines. It is also relatively easy to translate DATA STEP instruction into other programming languages if desired.

Although there are several benefits to this approach, others are possible and potentially more computationally efficient under specific specific choices for the candidate estimator library or specific hardware configurations. An example of the former might include so-called stacked regression, where each candidate is a different a priori specified generalized linear regression model. If this were the case, then BY processing might be used to improve computational efficiency. The %SUPERLEARNER macro presented here was intended to incorporate candidate estimators of diverse types and uses a serial implementation of V-fold cross validation for a library of K candidates intended for a single-computer user with a single disk IO channel. Cross validation of the candidate estimator library is in essence a highly parallelized algorithm, and given a system with $K(V+1)$ IO channels and a roughly equal number of processors, the run time of a parallelized Super Learner algorithm would theoretically be only as long as that of the lengthiest candidate estimator plus the run time of the procedure for fitting the convex weights. The code presented here may be easily extended to take advantage of such hardware, if available, via the SAS/CONNECT module.

In any case, the %SUPERLEARNER macro presented here is not an end in itself. Rather, it provides a framework for data-adaptive machine learning in SAS. The performance of any given Super Learner is only (asymptotically) as good as the oracle in the user-supplied library, so construction of good candidate estimators is of utmost importance. The example candidate estimators presented here are somewhat diverse, but each includes several tuning parameters that might be adjusted for particular problems. Users of %SUPERLEARNER are encouraged to construct their own candidate estimator macros within this framework with an aim towards generating a large and diverse library.

A.10 %SUPERLEARNER code

```
*****************************************************************************;
* %SUPERLEARRNER                                                            *;
*                                                                           *;
* SAS MACRO FOR DATA-ADAPTIVE MACHINE LEARNING                              *;
* FITS A SUPER LEARNER FOR CONDITIONAL EXPECTATION OF BINARY OR CONTINUOUS  *;
* OUTCOMES IN SINGLE-TIME POINT OR LONGITUDIINAL DATA STRUCTURES            *;
*                                                                           *;
* JORDAN BROOKS                                                             *;
*                                                                           *;
*****************************************************************************;
* TESTED ON SAS 9.2 WINDOWS 64-BIT                                          *;
*****************************************************************************;
* INPUTS TO THE %SUPERLEARNER MACRO:                                        *;
*                                                                           *;
* TRAIN                 = Input SAS dataset containing training data        *;
*                                                                           *;
* TEST (optional)       = Input SAS dataset containing validation data      *;
*                                                                           *;
* Y                     = Outcome/target variable. Must be NUMERIC.         *;
*                         Binary outcomes must be coded 0/1.                *;
*                         Missing values are not allowed.                   *;
*                                                                           *;
* Y_TYPE                = Type of outcome/target variable:                  *;
*                         CTS for continuous outcome                        *;
*                         BIN for binary outcome                            *;
*                                                                           *;
* X                     = Explanatory variables used to predict Y.          *;
*                         Must be NUMERIC. Missing values are not allowed.  *;
*                                                                           *;
* ID                    = Unique identifier of independent units/subjects   *;
*                                                                           *;
* t (optional)          = Name of the time-stamp variable for longitudinal  *;
*                         data structures. Must be NUMERIC.                 *;
*                                                                           *;
* WEIGHTS (optional)    = Observation WEIGHTS. Must be NUMERIC non-negative. *;
*                                                                           *;
* SL_LIBRARY            = Names of SAS macros for candidate estimators in the *;
*                         Super Learner library. The macros must be saved   *;
*                         under the exact name of candidate estimator as a  *;
*                         .sas program in the filepath given in LIBRARY_DIR. *;
*                                                                           *;
* LIBRARY_DIR           = Filepath for directory containing the candidate   *;
*                         estimator macros, which are saved as .sas programs. *;
*                                                                           *;
* EnterpriseMiner       = Enter T/F to indicate use of SAS/EnterpriseMiner. *;
*                         If set to T, a SAS datamining database catalog is *;
*                         constructed based on the training data sample.    *;
*                                                                           *;
* LOWER_BOUND (optional)= Lower bound for predictions. This is applied to all *;
*                         candidate estimators. If the user does not specify *;
*                         a lower bound, the defaults are:                  *;
*                         0.00000001 for Y_TYPE=BIN, or                     *;
*                         -9999999999999999 for Y_TYPE=CTS                  *;
*                                                                           *;
* UPPER_BOUND (optional)= Upper bound for predictions. This is applied to all *;
*                         candidate estimators. If the user does not specify *;
*                         an upper bound, the defaults are:                 *;
*                         0.99999999 for Y_TYPE=BIN, or                     *;
*                         9999999999999999 for Y_TYPE=CTS                   *;
*                                                                           *;
* LOSS                  = The LOSS function, whose expectation, i.e., RISK,   *;
*                         is minimized:                                     *;
*                         L2 for squared error RISK                         *;
*                         LOG for negative Bernoulli loglikelihood RISK     *;
*                         Note: The LOSS is a function of the unit data     *;
*                         structure. In longtiudinal data structures, the   *;
*                         LOSS function is taken to be the sum of the time- *;
*                         point specific LOSSes for each unit/subject.      *;
*                                                                           *;
* V                     = The number of folds for V-fold cross validation   *;
*                         NOTE: For Y_TYPE=BIN the random sampling for fold *;
*                         assignment is stratified on the outcome.          *;
*                                                                           *;
* FOLD (optional)       = Variable containing the fold number for cross     *;
*                         validation. If none is specified then FOLDs are   *;
*                         are assigned randomly.                            *;
*                                                                           *;
* SEED                  = The SEED for the random number generator.         *;
*                                                                           *;
* WD                    = The working directory. All candidate algorithm fits *;
*                         will be saved to this directory as .sas programs. *;
*                         The inputs to the %SUPERLEARNER SAS macro and the *;
*                         fitted alpha WEIGHTS will be saved here as SAS    *;
*                         datasets. This directory will be assigned a SAS   *;
*                         libname "SL" during the fitting process.          *;
*                                                                           *;
* VERBOSE (optional)    = A T/F that indicates whether all output normally  *;
*                         produced by SAS data and procedure steps should be *;
*                         sent to the listing device. The default is F.     *;
*****************************************************************************;
%MACRO SUPERLEARNER(TRAIN=, TEST=, Y=, Y_TYPE=, X=, ID=, T=, WEIGHTS=,
                    SL_LIBRARY=, LIBRARY_DIR=, EnterpriseMiner=T,
                    LOWER_BOUND=, UPPER_BOUND=, LOSS=L2, V=10, SEED=715, FOLD=,
                    VERBOSE=F, WD=);

options nosyntaxcheck;

**************************************************;
* INDEX THE EXISTING VARIABLES ON THE TRAINING SET *;
**************************************************;
proc contents data = &TRAIN out = _TRAIN_NAMES(keep = varnum name) noprint; run;
proc sql noprint;
 select distinct name into :KEEP_TRAIN separated by ' ' from _TRAIN_NAMES;
quit;
%LET nKEEP_TRAIN = %sysfunc(countw(&KEEP_TRAIN, ' '));
%DO _i=1 %TO &nKEEP_TRAIN;
 %LET KEEP_TRAIN_&_i = %SCAN(&KEEP_TRAIN, &_i, " ");
%END;

***************************************************;
* TIDY UP INPUTS AND SET DEFAULTS BEFORE PROCESSING *;
***************************************************;
%IF &VERBOSE=F %THEN %DO; ods listing close; %END;
%LET X = %sysfunc(COMPBL(&X));
%LET nX = %sysfunc(countw(&X));
%DO _i=1 %TO &nX;
 %LET X_&_i = %SCAN(&X, &_i, " ");
%END;
%IF "&LOWER_BOUND" = %THEN %DO;
 %IF &LOSS=LOG %THEN %DO; %LET LOWER_BOUND=0.00000001; %END;
 %IF &LOSS=L2  %THEN %DO; %LET LOWER_BOUND=-9999999999999999; %END;
```

```
%END;
%IF "&UPPER_BOUND" = %THEN %DO;
 %IF &LOSS=LOG %THEN %DO; %LET UPPER_BOUND=0.99999999; %END;
 %IF &LOSS=L2  %THEN %DO; %LET UPPER_BOUND=9999999999999999; %END;
%END;
%LET  SL_LIBRARY = %sysfunc(COMPBL(&SL_LIBRARY));
%LET K = %sysfunc(countw(&SL_LIBRARY));
%DO _i=1 %TO &K;
 %LET cand_&_i = %SCAN(&SL_LIBRARY, &_i, " ");
 %include "&LIBRARY_DIR\&&cand_&_i...sas";
%END;

*****************************;
* SORT BY ID and time-stamp *;
*****************************;
proc sort data=&TRAIN;
 by &ID &t;
run;

*************************************;
* ASSIGN FOLDS FOR CROSS VALIDATION *;
*************************************;
%IF &FOLD = %THEN %DO;
%LET FOLD = fold;
%IF &Y_TYPE=CTS %THEN %DO;
 data _fold;
  set &TRAIN;
  by &ID &t;
  if last.&ID;
  rand=&V*ranuni(&SEED);
  fold=ceil(rand);
  keep &ID fold;
 run;
 proc sort; by &ID; run;
 data &TRAIN;
  merge &TRAIN _fold;
  by &ID;
 run;
 proc datasets lib=work; delete _: ; run; quit;
%END;
%ELSE %IF &Y_TYPE=BIN %THEN %DO;
 data _last;
  set &TRAIN;
  by &ID &t;
  if last.&ID;
 run;
 data _Y0;
  set _last (where=(&Y=0) keep = &ID &Y);
  rand=&V*ranuni(&SEED);
  fold=ceil(rand);
  keep &ID fold;
 run;
 data _Y1;
  set _last (where=(&Y=1) keep = &ID &Y);
  rand=&V*ranuni(&SEED);
  fold=ceil(rand);
  keep &ID fold;
 run;
 data _fold;
  set _Y0 _Y1;
 run;
 proc sort; by &ID; run;
 data &TRAIN;
  merge &TRAIN _fold;
  by &ID;
 run;
 proc datasets lib=work; delete _: ; run; quit;
%END;
%END;

*****************************************************************;
* SAVE INPUTS TO SUPER LEARNER PROCEDURE TO PERMANENT SAS DATASET *;
*****************************************************************;
libname sl "&WD";
data sl.sl_inputs;
 TRAIN          ="&TRAIN";
 TEST           ="&TEST";
 Y              ="&Y";
 Y_TYPE         ="&Y_TYPE";
 X              ="&X";
 ID             ="&ID";
 T              ="&t";
 WEIGHTS        ="&WEIGHTS";
 LOSS           ="&LOSS";
 SL_LIBRARY     ="&SL_LIBRARY";
 LIBRARY_DIR    ="&LIBRARY_DIR";
 LOWER_BOUND    ="&LOWER_BOUND";
 UPPER_BOUND    ="&UPPER_BOUND";
 EnterpriseMiner="&EnterpriseMiner";
 V              ="&V";
 SEED           ="&SEED";
 FOLD           ="&FOLD";
 WD             ="&WD";
 VERBOSE        ="&VERBOSE";
run;

************************************;
* CONSTRUCT SAS DATAMINING CATALOG *;
************************************;
%IF &EnterpriseMiner = T %THEN %DO;
 %IF &Y_TYPE=CTS %THEN %DO;
  proc dmdb batch data=&TRAIN dmdbcat=sl.dmdbcat; VAR &X &Y; target &Y; run;
 %END;
 %ELSE %IF &Y_TYPE=BIN %THEN %DO;
  proc dmdb batch data=&TRAIN dmdbcat=sl.dmdbcat; VAR &X; class &Y; target &Y; run;
 %END;
%END;

****************************************************************;
* CONSTRUCT CROSS VALIDATED FITS FOR CANDIDATE ESTIMATOR LIBRARY *;
****************************************************************;
title2 'CONSTRUCT AND SAVE CROSS VALIDATED CANDIDATE ESTIMATOR FITS';
%DO b=1 %TO &V;
 %DO _k=1 %TO &K;
  %&&cand_&_k(TRAIN=&TRAIN(where=(&FOLD ne &b)), ID=&ID, T=&t, WEIGHTS=&WEIGHTS,
             Y=&Y, Y_TYPE=&Y_TYPE, X=&X, SEED=&SEED, WD=&WD);
  data _null_;
   fname="fname";
   rc=filename(fname,"&WD\F_&&cand_&_k.._&b..sas");
   if fexist(fname) then rc=fdelete(fname);
   rc=rename("&WD\F_&&cand_&_k...sas","&WD\F_&&cand_&_k.._&b..sas","file");
  run;
  proc datasets lib=work; delete _: ; run; quit;
 %END;
 proc datasets lib=sl; delete _: ; run; quit;
%END;

*******************************************************************;
* COMPUTE CROSS VALIDATED ESTIMATES FOR CANDIDATE ESTIMATOR LIBRARY *;
*******************************************************************;
data &TRAIN;
 set &TRAIN;
 %DO _fold=1 %TO &V;
  if &FOLD=&_fold then do;
   %DO _k = 1 %TO &K; %include "&WD\F_&&cand_&_k.._&_fold..sas"; %END;
  end;
 %END;
 array p {&K} %DO _k = 1 %TO &K; p_&&cand_&_k %END; ;
 do _i=1 to &K;
  p{_i} = max(p{_i},&LOWER_BOUND);
  p{_i} = min(p{_i},&UPPER_BOUND);
 end;
 keep %DO _i=1 %TO &nKEEP_TRAIN; &&KEEP_TRAIN_&_i %END;
 %DO _k=1 %TO &K; p_&&cand_&_k %END; ;
run;

*******************************************;
* COMPUTE USUAL CROSS VALIDATION SELECTOR *;
*******************************************;
%IF &K = 1 %THEN %DO; %LET minCVRISK=1; %END;
%ELSE %DO;
 %RISK(DSN=&TRAIN, Y=&Y, YHAT=%DO _k = 1 %TO &K; p_&&cand_&_k %END;,
       WEIGHTS=&WEIGHTS, LOSS=&LOSS, ID=&ID);
 data _null_;
  set RISK_sum;
  where _STAT_="MEAN";
  array RISK{&K} %DO _k = 1 %TO &K; loss_&_k %END; ;
  minCVRISK = min(of RISK{*});
  do _i=1 to &K;
   if RISK{_i}=minCVRISK then do; call symputx("minCVRISK", _i, 'G'); end;
  end;
  drop _i;
 run;
%END;

*********************************;
* ESTIMATE SUPER LEARNER WEIGHTS *;
*********************************;
title2 'ESTIMATE CONVEX WEIGHTS: OPTIMIZATION STARTS AT CROSS VALIDATION SELECTOR';
%LET alpha=;
%LET init=;
%DO _i=1 %TO &K;
 %LET alpha = &alpha a&_i;
 %IF &_i = &minCVRISK %THEN %DO; %LET init = &init 1; %END;
 %ELSE %DO; %LET init = &init 0; %END;
%END;

%IF &LOSS=L2 %THEN %DO;
 proc nlp data=&TRAIN;
  lsq L2;
  parms %DO _i=1 %TO &K; a&_i =%SCAN(&init, &_i, " "), %END; ;
  bounds 0 <= %DO _i=1 %TO &K; a&_i %END; <=1;
  lincon  0 %DO _i=1 %TO &K; + a&_i %END; = 1;
  %IF &WEIGHTS ne %THEN %DO;
   L2 = &WEIGHTS**0.5*(&Y - (%DO _i=1 %TO &K; + a&_i * p_&&cand_&_i %END;));
  %END;
  %ELSE %DO;
   L2 = (&Y - (%DO _i=1 %TO &K; + a&_i * p_&&cand_&_i %END;));
  %END; ;
  ods output ParameterEstimates=sl.alpha;
 quit;
%END;

%ELSE %IF &LOSS=LOG %THEN %DO;
 proc nlp data=&TRAIN;
  min logL;
  parms %DO _i=1 %TO &K; a&_i =%SCAN(&init, &_i, " "), %END; ;
  bounds 0 <= %DO _i=1 %TO &K; a&_i %END; <=1;
  lincon  0 %DO _i=1 %TO &K; + a&_i %END; = 1;
  %LET P=; %DO _i=1 %TO &K; %LET P = &P p_&&cand_&_i ; %END;
  array P{&K} %DO _k = 1 %TO &K; %SCAN(&P, &_k, ' ') %END; ;
  array logit{&K} %DO _k = 1 %TO &K; logit_%SCAN(&P, &_k, ' ') %END; ;
  do _i = 1 to &K;
   P{_i} = max(P{_i},&LOWER_BOUND);
   P{_i} = min(P{_i},&UPPER_BOUND);
   logit{_i} = log(P{_i}/(1-P{_i})) ;
  end;
  drop _:;
  %LET linpart = ;
  %DO _i=1 %TO &K; %LET linpart = &linpart + a&_i * logit_p_&&cand_&_i; %END;
  %IF &WEIGHTS ne %THEN %DO;
   logL = 2*&WEIGHTS*(log(1+exp(-1*(&linpart)))-(&Y-1)*(&linpart));
  %END;
  %ELSE %DO;
   logL = 2*(log(1+exp(-1*(&linpart)))-(&Y-1)*(&linpart));
  %END;
  ods output ParameterEstimates=sl.alpha;
 quit;
%END;

****************************************;
* SAVE AND INDEX SUPER LEARNER WEIGHTS *;
****************************************;
data sl.alpha; set sl.alpha nobs=nobs; if _n_ ge (nobs-&K+1); run;
data sl.alpha;
 set sl.alpha (keep = Parameter Estimate);
 Candidate = resolve(catt('&cand_'||trim(left(_n_))));
 call symputx(catt("a",_n_), Estimate, 'G');
run;
proc datasets lib=work; delete _: ; run; quit;

**************************************************;
* CONSTRUCT FITS FOR CANDIDATE ESTIMATOR LIBRARY *;
**************************************************;
title2 'FIT CANDIDATE ESTIMATORS ON ENTIRE TRAINING DATASET';
%DO _k=1 %TO &K;
 %&&cand_&_k(TRAIN=&TRAIN, ID=&ID, T=&t, WEIGHTS=&WEIGHTS,
            Y=&Y, Y_TYPE=&Y_TYPE, X=&X, SEED=&SEED, WD=&WD);
 proc datasets lib=work; delete _: ; run; quit;
%END;
proc datasets lib=sl; delete _: ; quit;

********************************;
* CONSTRUCT SUPER LEARNER FIT *;
********************************;
data _null_;
 file "&WD\F_SuperLearner.sas";
 put "***************************************;";
 put "*             SUPER LEARNER           *;";
 put "**************************************;";
 %DO _k = 1 %TO &K;
  put "* &&cand_&_k *;";
 %END;
run;
%DO _k = 1 %TO &K;
 data _null_;
  infile "&WD\F_&&cand_&_k...sas";
  input;
  file "&WD\F_SuperLearner.sas" MOD;
  put _infile_;
 run;
%END;
data _null_;
 file "&WD\F_SuperLearner.sas" MOD;
 put "*********************;" ;
 put "* BOUND PREDICTIONS *;" ;
 put "*********************;" ;
 put "array P{&K} ";
 %DO _k = 1 %TO &K;
  put "p_&&cand_&_k" ;
 %END;
 put ";";
 put "do _i=1 to &K;" ;
 put " p{_i} = max(p{_i},&LOWER_BOUND);" ;
 put " p{_i} = min(p{_i},&UPPER_BOUND);" ;
 put "end;" ;
 put "***************************;" ;
 put "* SUPER LEARNER WEIGHTING *;" ;
 put "***************************;" ;
 %IF &LOSS=L2 %THEN %DO;
  put " p_SL = 0 " ;
  %DO _k = 1 %TO &K;
   put " + &&a&_k * p_&&cand_&_k" ;
  %END;
  put ';' ;
 %END;
 %ELSE %IF &LOSS=LOG %THEN %DO;
  put "array logit{&K}" %DO _k = 1 %TO &K; " logit_p_&&cand_&_k" %END; ";";
  put "do _i = 1 to &K;";
  put " logit{_i} = log(P{_i}/(1-P{_i}));";
  put "end;";
  put " p_SL = 1/(1+exp(-1*(0+ " ;
  %DO _k = 1 %TO &K;
   put " + &&a&_k * logit_p_&&cand_&_k" ;
  %END;
  put ')));' ;
 %END;
 put "keep ";
 %DO _i=1 %TO &nKEEP_TRAIN;
  put "&&KEEP_TRAIN_&_i ";
 %END;
 %DO _k=1 %TO &K;
  put "p_&&cand_&_k";
 %END;
 put "p_SL ;" ;
run;

**************************************************************************;
* COMPUTE SUPER LEARNER ESTIMATES FOR VALIDATION SET (TEST) IF PROVIDED *;
**************************************************************************;
```

```
%IF &TEST ne %THEN %DO;
 title2 'COMPUTE PREDICTIONS FOR VALIDATION/TEST DATA SET';
 data &TEST; set &TEST; %include "&WD\F_SuperLearner.sas"; run;
%END;

******************************;
* PRINT SUPER LEARNER WEIGHTS *;
******************************;
ods listing;
title2 "ALPHA";
proc print data=sl.alpha; run;

************************************************;
* PRINT CROSS VALIDATED RISK OF CANDIDATE LIBRARY *;
************************************************;
title2 "CROSS VALIDATED RISK ON TRAINING DATA";
%RISK(DSN=&TRAIN, Y=&Y, YHAT=%DO _k = 1 %TO &K; p_&&cand_&_k %END;,
      WEIGHTS=&WEIGHTS, LOSS=&LOSS, ID=&ID);
proc datasets lib=work; delete LOSS; run; quit;

*************************************************************************;
* PRINT RISK OF CANDIDATE LIBRARY AND SUPER LEARNER ON VALIDATION DATA (TEST) *;
*************************************************************************;
%IF &TEST ne %THEN %DO;
 title2 "RISK ON VALIDATION/TEST DATA";
 %RISK(DSN=&TEST, Y=&Y, YHAT=%DO _k = 1 %TO &K; p_&&cand_&_k %END; p_SL,
       WEIGHTS=&WEIGHTS, LOSS=&LOSS, ID=&ID);
 proc datasets lib=work; delete LOSS; run; quit;
%END;

%MEND SUPERLEARNER;
```

A.11 Utility macro: %RISK

```
***********************************************************************;
* RISK - COMPUTE SQUARED ERROR (L2) or BERNOULLI LOGLIKELIHOOD (LOG) RISK   *;
***********************************************************************;
%MACRO RISK(DSN=, Y=, YHAT=, WEIGHTS=, LOSS=, ID=, T=);

%LET P = %sysfunc(countw(&YHAT));
proc sort data=&DSN; by &ID &t; run;

%IF &LOSS=L2 %THEN %DO;
 data loss;
  set &DSN;
  by &ID &t;
  retain %DO _k = 1 %TO &P; loss_&_k %END; ;
  array loss{&P} %DO _k = 1 %TO &P; loss_&_k %END; ;
  array yhat{&P} %DO _k = 1 %TO &P; %SCAN(&YHAT, &_k, ' ') %END; ;
  if first.&ID then do; do i=1 to &P; loss{i} = 0; end; end;
  do i=1 to &P;
   loss{i} = loss{i} +
   %IF &WEIGHTS ne %THEN &WEIGHTS*((&Y-yhat{i})**2);
   %ELSE (&Y-yhat{i})**2; ;
  end;
  if last.&ID;
  %DO _i=1 %TO &P; label loss_&_i = %BQUOTE(%SCAN(&YHAT, &_i, " ")); %END;
 run;
 title3 "L2-loss: &DSN";
 proc means;
  VAR %DO _k = 1 %TO &P; loss_&_k %END; ;
  output out=RISK_sum;
 run;
%END;

%ELSE %IF &LOSS=LOG %THEN %DO;
 data loss;
  set &DSN;
  by &ID &t;
  retain %DO _k = 1 %TO &P; loss_&_k %END; ;
  array loss{&P} %DO _k = 1 %TO &P; loss_&_k %END; ;
  array yhat{&P} %DO _k = 1 %TO &P; %SCAN(&YHAT, &_k, ' ') %END; ;
  if first.&ID then do; do i=1 to &P; loss{i} = 0; end; end;
  do i=1 to &P;
   loss{i} = loss{i} +
   %IF &WEIGHTS ne %THEN -2*&WEIGHTS*(&Y*log(yhat{i}) + (1-&Y)*log(1-yhat{i}));
   %ELSE -2*(&Y*log(yhat{i}) + (1-&Y)*log(1-yhat{i})); ;
  end;
  if last.&ID;
  %DO _i=1 %TO &P; label loss_&_i = %BQUOTE(%SCAN(&YHAT, &_i, " ")); %END;
 run;
 title3 "Log-loss: &DSN";
 proc means;
  VAR %DO _k = 1 %TO &P; loss_&_k %END; ;
  output out=RISK_sum;
 run;
%END;

%MEND RISK;
```

A.12 Candidate estimator macros

A.12.1 Unconditional mean (NULL model)

```
%MACRO MEAN(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

********************************;
* DESCRIPTIVE TITLE STATEMENTS *;
********************************;
title3 "MEAN or NULL MODEL";
title4 "FITTING: UNCONDITIONAL MEAN";
title5 "VARIABLE SELECTION: ALL";

***************************;
* CONSTRUCT ESTIMATOR FIT *;
***************************;
proc means data=&TRAIN mean;
 var &Y;
 %IF &WEIGHTS ne %THEN %DO; weight &WEIGHTS; %END;
 output out=_mean mean=mean;
run;

**********************;
* SAVE ESTIMATOR FIT *;
**********************;
data _null_;
 set _mean ;
 file "&WD\F_MEAN.sas";
 put "p_MEAN =" mean ";" ;
run;

*******************;
* CLEAN WORK SPACE *;
*******************;
proc datasets lib=work; delete _: ; quit;

%MEND MEAN;
```

A.12.2 Ordinary Least Squares Regression

```
%MACRO OLS(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

*******************************;
* DESCRIPTIVE TITLE STATEMENTS *;
*******************************;
title3 "OLS";
title4 "FITTING: ORDINARY LEAST SQUARES REGRESSION";
title5 "VARIABLE SELECTION: ALL";

**************************;
* CONSTRUCT ESTIMATOR FIT *;
**************************;
proc reg data=&TRAIN;
 model &Y = &X;
 %IF &WEIGHTS ne %THEN %DO; weight &WEIGHTS; %END;
 ods output ParameterEstimates=_MyCoef;
run;

*********************;
* SAVE ESTIMATOR FIT *;
*********************;
data _null_;
 set _MyCoef end=eof;
 file "&WD\F_OLS.sas";
 if _n_ = 1 then put "p_OLS = ";
 if Variable="Intercept" then put Estimate;
 else put "+" Estimate "*" Variable ;
 if eof then put ";" ;
run;

*******************;
* CLEAN WORK SPACE *;
*******************;
proc datasets lib=work; delete _: ; quit;

%MEND OLS;
```

A.12.3 LASSO Squared Error Regression

```
%MACRO LASSO(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

*******************************;
* DESCRIPTIVE TITLE STATEMENTS *;
*******************************;
title3 "LASSO";
```

```
title4 "FITTING: LASSO SQUARED ERROR REGRESSION";
title5 "VARIABLE SELECTION: ALL MAIN TERMS";

***************************;
* CONSTRUCT ESTIMATOR FIT *;
***************************;
proc glmselect data=&TRAIN SEED=&SEED;
 model &Y = &X / selection=lasso(stop=cv) cvdetails=all cvmethod=random;
 %IF &WEIGHTS ne %THEN %DO; weight &WEIGHTS; %END;
 ods output ParameterEstimates=_MyCoef;
run;

**********************;
* SAVE ESTIMATOR FIT *;
**********************;
data _null_;
 set _MyCoef end=eof;
 file "&WD\F_LASSO.sas";
 if _n_ = 1 then put "p_LASSO = ";
 if Parameter="Intercept" then put Estimate;
 else put "+" Estimate "*" Parameter ;
 if eof then put ";" ;
run;

********************;
* CLEAN WORK SPACE *;
********************;
proc datasets lib=work; delete _: ; quit;

%MEND LASSO;
```

A.12.4 Logistic Regression

```
%MACRO LOGIT(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

********************************;
* DESCRIPTIVE TITLE STATEMENTS *;
********************************;
title3 "LOGIT";
title4 "FITTING: LOGISTIC REGRESSION";
title5 "VARIABLE SELECTION: ALL";

***************************;
* CONSTRUCT ESTIMATOR FIT *;
***************************;
proc logistic data=&TRAIN descending;
 model &Y = &X;
 %IF &WEIGHTS ne %THEN %DO; weight &WEIGHTS; %END;
 ods output ParameterEstimates=_MyCoef;
run;

**********************;
* SAVE ESTIMATOR FIT *;
**********************;
data _null_;
 set _MyCoef end=eof;
 file "&WD\F_LOGIT.sas";
 if _n_ = 1 then put "p_LOGIT = 1/(1+exp(-1*(";
 if Variable="Intercept" then put Estimate;
 else put "+" Estimate "*" Variable ;
 if eof then put ")));" ;
run;

*******************;
* CLEAN WORK SPACE *;
*******************;
proc datasets lib=work; delete _: ; quit;

%MEND LOGIT;
```

A.12.5 Logistic regression for continuous outcomes in [0,1]

```
%MACRO LOGIT_CTS01(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

********************************;
* DESCRIPTIVE TITLE STATEMENTS *;
********************************;
title3 "LOGIT_CTS01";
title4 "FITTING: LOGISTIC REGRESSION FOR CONTINUOUS OUTCOMES in (0,1)";
title5 "VARIABLE SELECTION: ALL";

***************************;
* CONSTRUCT ESTIMATOR FIT *;
***************************;
%LET nX = %sysfunc(countw(&X));
proc nlmixed data=&TRAIN;
 parms b0=0 %DO _i=1 %TO &nX;, b&_i=0 %END; ;
 p = 1/(1+exp(-1*(b0 %DO _i=1 %TO &nX; + b&_i * %SCAN(&X, &_i, ' ') %END;)));
 %IF &WEIGHTS ne %THEN %DO; logL = 2*&WEIGHTS*(&Y*log(p)+(1-&Y)*log(1-p)); %END;
 %ELSE %DO; logL = 2*(&Y*log(p)+(1-&Y)*log(1-p)); %END;
 model &Y ~ general(logL);
 ods output ParameterEstimates= _MyCoef;
run;

**********************;
* SAVE ESTIMATOR FIT *;
**********************;
data _null_;
 set _MyCoef end=eof;
 file "&WD\F_LOGIT_CTS01.sas";
 _n_1 = _n_ - 1;
 if _n_ = 1 then do;
  put "p_LOGIT_CTS01 = 1/(1+exp(-1*(" Estimate;
 end;
 else do;
 %DO _i=1 %TO &nX;
  if _n_ = &_i+1 then do;
   put "+" Estimate "* %SCAN(&X, &_i, ' ')";
  end;
 %END;
 end;
 if eof then put ")));" ;
run;

*******************;
* CLEAN WORK SPACE *;
*******************;
proc datasets lib=work; delete _: ; quit;

%MEND LOGIT_CTS01;
```

A.12.6 Linear Discriminant Analysis

```
%MACRO LDA(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

********************************;
* DESCRIPTIVE TITLE STATEMENTS *;
********************************;
title3 "LDA";
title4 "FITTING: LINEAR DISCRIMINANT ANALYSIS";
title5 "VARIABLE SELECTION: ALL ";

***************************;
* CONSTRUCT ESTIMATOR FIT *;
***************************;
proc discrim data=&TRAIN
 method=NORMAL pool=YES;
 class &Y;
 var &X;
 %IF &WEIGHTS ne %THEN %DO; weight &WEIGHTS; %END;
 priors prop;
 ods output LinearDiscFunc=_LDF;
run;

**********************;
* SAVE ESTIMATOR FIT *;
**********************;
data _null_;
 set _LDF end=eof;
 file "&WD\F_LDA.sas";
 if _n_ = 1 then put "_ldf0 =" _0 ;
 else put "+" _0 "*" Variable;
 if eof then put ";" ;
run;
data _null_;
 set _LDF end=eof;
 file "&WD\F_LDA.sas" MOD;
 if _n_ = 1 then put "_ldf1 =" _1 ;
 else put "+" _1 "*" Variable;
 if eof then put ";" ;
run;
data _null_;
 file "&WD\F_LDA.sas" MOD;
 put "p_LDA = exp(_ldf1)/(exp(_ldf0) + exp(_ldf1));" ;
run;

********************;
* CLEAN WORK SPACE *;
********************;
proc datasets lib=work; delete _: ; quit;

%MEND LDA;
```

A.12.7 Multilayer Perceptron Artificial Neural Network

```
%MACRO NN2(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

********************************;
* DESCRIPTIVE TITLE STATEMENTS *;
********************************;
title3 "NN2";
```

```
title4 "FITTING: NEURAL NET (MLP WITH 2 HIDDEN UNITS)";
title5 "VARIABLE SELECTION: ALL";

***************************;
* CONSTRUCT ESTIMATOR FIT *;
***************************;
proc neural data=&TRAIN dmdbcat=sl.dmdbcat random=&SEED;
 %IF &Y_TYPE=BIN %THEN %DO;
  netoptions object=LIKE;
  target &Y  / id=o level=ordinal;
 %END;
 %ELSE %IF &Y_TYPE=CTS %THEN %DO;
  netoptions object=DEV;
  target &Y  / id=o level=interval;
 %END;
 input &X / id=i level=interval;
 hidden 2 / id=h;
 prelim 5;
 code file="&WD\F_NN2.sas";
run;

**********************;
* SAVE ESTIMATOR FIT *;
**********************;
data _null_;
 file "&WD\F_NN2.sas" MOD;
 %IF &Y_TYPE=BIN %THEN %DO; put "p_NN2 = p_&Y.1 ;" ; %END;
 %ELSE %IF &Y_TYPE=CTS %THEN %DO; put "p_NN2 = p_&Y ;" ; %END;
run;

%MEND NN2;
```

A.12.8 Recursive Partitioning Decision Tree

```
%MACRO TREE(TRAIN, Y, Y_TYPE, X, ID, T, WEIGHTS, SEED, WD);

********************************;
* DESCRIPTIVE TITLE STATEMENTS *;
********************************;
title3 "TREE";
title4 "FITTING: DECISION TREE WITH COMPLEXITY PRUNING";
title5 "VARIABLE SELECTION: ALL";

***************************;
* CONSTRUCT ESTIMATOR FIT *;
***************************;
proc arboretum data=&TRAIN;
 performance multipass;
 %IF &Y_TYPE=BIN %THEN %DO;
  target &Y / level=BINARY;
  input &X;
  assess measure=ASE prunedata=train;
 %END;
 %ELSE %IF &Y_TYPE=CTS %THEN %DO;
  target &Y / level=INTERVAL;
   input &X;
   assess measure=ASE prunedata=train;
 %END;
 code file="&WD\F_TREE.sas";
run;
```

```
quit;

**********************;
* SAVE ESTIMATOR FIT *;
**********************;
data _null_;
 file "&WD\F_TREE.sas" MOD;
 %IF &Y_TYPE=BIN %THEN %DO; put "p_TREE = p_&Y.1 ;"; %END;
 %ELSE %IF &Y_TYPE=CTS %THEN %DO; put "p_TREE = p_&Y ;"; %END;
run;

%MEND TREE;
```

Appendix B

Supplement for TMLE for calibration

B.1 Efficient Influence Curves for calibration

In this Appendix we provide the efficient influence curve for all the parameters discussed in the main text. In brief, the efficient influence curve, D^*, is a fundamental property of a parameter, characterized as a mapping on the distribution of the data. It is unique and is given by the pathwise derivative of this mapping evaluated at the true distribution of the data (van der Laan and Rose, 2011). Asymptotically linear estimators are are defined as estimators that can be written as the empirical mean of D^* plus sum typically second order term. D^* is a function of the true probability distribution, P_0, and the data O, and is therefore itself a random variable. Its variance defines the efficiency bound for unbiased estimator in a (possibly nonparametric) statistical model. Often, the efficient influence curve involves the parameter itself, and can be used to derive an estimating equation.

Estimators are mappings from an empirical distribution to the parameter space. The influence curve of an estimator is the pathwise derivative of the (estimator) mapping. Theory teaches us that an estimator is efficient in a statistical model if and only if its influence curve is equal to the efficient influence curve. We use this fact to derive the efficient influence curves for our parameters here. Finally we show that all the TMLEs described in this article solve the efficient influence curve estimating equations for their respective parameters.

B.1.1 EIC for calibration to a scalar proportion ψ_0

Recall the data structure $O = (W,Y) \sim P_0 \in \mathscr{M}$, and let $S = S(W)$ be some summary measure of the covariates. The calibration parameter was defined

$$\psi_0 = E_{Q_{W,0}}[Y|I_{(\mathscr{A})}\{S\} = 1] = E_{Q_{W,0}}[\bar{Q}_{Y,0}|I_{(\mathscr{A})}\{S\} = 1]$$

Consider the nonparametric maximum likelihood empirical estimator

$$\psi_n = \frac{\frac{1}{n}\sum_{i=1}^{n} Y_i I_{\mathscr{A}}\{S_i\}}{\frac{1}{n}\sum_{i=1}^{n} I_{\mathscr{A}}\{S_i\}}$$

This estimator is efficient (and unbiased) in the nonparametric model, which implies that its influence curve is equal to the efficient influence curve, except that we replace the empirical distribution P_n with the true distribution P_0.

$$\begin{aligned} D^*(P_0)(O) \quad = \quad & \frac{1}{E_0[I_{(\mathscr{A})}\{S\}]}\left\{ YI_{(\mathscr{A})}\{S\} - E_0[YI_{(\mathscr{A})}\{S\}]\right\} \\ & - \frac{E_0[YI_{(\mathscr{A})}\{S\}]}{E_0[I_{(\mathscr{A})}\{S\}]^2}\left\{ I_{(\mathscr{A})}\{S\} - E_0[I_{(\mathscr{A})}\{S\}]\right\} \end{aligned}$$

To construct a TMLE for ψ_0, we must write D^* as the sum of the score of a function of (Y,W) with conditional mean 0 given W and the score of a mean 0 function of W. The first term can be decomposed into a function of (Y,W) with conditional mean 0 given W and a function of W. For the second term, we can replace Y inside the expectation operator with its true conditional expectation given W, $\bar{Q}_{Y,0}(W)$.

$$D^*(P_0)(O) = D^*_Y(\bar{Q}_{Y,0}, Q_{W,0})(O) + D^*_{W,1}(\bar{Q}_{Y,0}, Q_{W,0})(O) + D^*_{W,2}(\bar{Q}_{Y,0}, Q_{W,0})(O)$$

where

$$D^*_Y(\bar{Q}_{Y,0}, Q_{W,0})(Y,W) = \frac{I_{(\mathscr{A})}\{S\}}{E_{Q_{W,0}}[I_{(\mathscr{A})}\{S\}]}\left\{ Y - \bar{Q}_{Y,0}(W)\right\}$$

$$D^*_{W,1}(\bar{Q}_{Y,0}, Q_{W,0})(W) = \frac{1}{E_{Q_{W,0}}[I_{(\mathscr{A})}\{S\}]}\left\{ \bar{Q}_{Y,0}(W) I_{(\mathscr{A})}\{S\} - E_{Q_{W,0}}[\bar{Q}_{Y,0}(W) I_{(\mathscr{A})}\{S\}]\right\}$$

$$D^*_{W,2}(\bar{Q}_{Y,0}, Q_{W,0})(W) = -\frac{E_{Q_{W,0}}[\bar{Q}_{Y,0}(W) I_{(\mathscr{A})}\{S\}]}{E_{Q_{W,0}}[I_{(\mathscr{A})}\{S\}]^2}\left\{ I_{(\mathscr{A})}\{S\} - E_{\bar{Q}_{Y,0}}[I_{(\mathscr{A})}\{S\}]\right\}$$

This decomposition makes it clear that $E_0[D^*(\bar{Q}_{Y,0}, Q_{W,0})] = 0$, and that D^* is spanned by the scores of mean 0 functions of W and the score of a function of (Y,W) with conditional mean 0, given W.

Note that the empirical distribution $Q_{W,n}$, which places probability mass $\frac{1}{n}$ on every observation, solves both

$$\frac{1}{n}\sum_{i=1}^{n} D^*_{W,1}(\bar{Q}_Y, Q_{W,n})(O_i) = 0$$

and
$$\frac{1}{n}\sum_{i=1}^{n} D^*_{W,2}(\bar{Q}_Y, Q_{W,n})(O_i) = 0$$
for any $\bar{Q}_Y$. This is implies that the empirical distribution $Q_{W,n}$ is already targeted towards our parameter ψ_0 and will not require any updating in the TMLE procedure.

The form of D^*_Y is the "clever covariate" multiplied by the residual. Our TMLE updated $\bar{Q}^*_{Y,n}$, used $\frac{I_{(\mathscr{A})}\{S\}}{E_{Q_{W,0}}[I_{(\mathscr{A})}\{S\}]}$ as a covariate which means that we directly solved
$$\frac{1}{n}\sum_{i=1}^{n} D^*_Y(Q_{W,n}, \bar{Q}^*_{Y,n})(O_i) = 0$$
Some simple re-arrangement shows that
$$\frac{\frac{1}{n}\sum_{i=1}^{n} I_{(\mathscr{A})}\{S_i\}Y_i}{\frac{1}{n}\sum_{i=1}^{n} I_{(\mathscr{A})}\{S_i\}} = \frac{\frac{1}{n}\sum_{i=1}^{n} I_{(\mathscr{A})}\{S_i\}\bar{Q}^*_{Y,n}(W_i)}{\frac{1}{n}\sum_{i=1}^{n} I_{(\mathscr{A})}\{S_i\}}$$
or simply
$$E_n[Y|I_{(\mathscr{A})}\{S_i\}] = E_n[\bar{Q}^*_{Y,n}(W_i)|I_{(\mathscr{A})}\{S_i\}]$$
That is, the empirical mean of our TMLE updated prediction function estimator is equal to the empirical mean of Y given $I_{(\mathscr{A})}\{S_i\}$. This makes complete sense because the both the empirical mean and our TMLE are both unbiased and efficient estimators that solve that efficient influence curve for ψ_0.

B.1.2 EIC for calibration to a vector of proportions

Recall the vector parameter
$$\Psi(P_0) = \begin{bmatrix}\psi_{0,1}\\ \vdots \\ \psi_{0,J}\end{bmatrix} = \begin{bmatrix}E_0[Y|I_{(\mathscr{A}_1)}\{S\}=1]\\ \vdots \\ E_0[Y|I_{(\mathscr{A}_J)}\{S\}=1]\end{bmatrix}$$
Because the parameter is now a vector with J components, its efficient influence curve is also a vector with J components. The form each component of this vector efficient influence curve is similar to that for the scalar parameter described above.
$$D^*(P_0)(O) = \begin{bmatrix}D^*_1(P_0)(O)\\ \vdots \\ D^*_J(P_0)(O)\end{bmatrix}$$
Each component can be decomposed into score functions that are mean 0 function of W and mean 0 functions of (Y,W) with conditional mean 0, given W, as before. The same properties hold.

Again, the empirical distribution of W solves those components that only depend on W, thus obviating the need for TMLE updates to QWn. Our TMLE updated $\bar{Q}^*_{Y,n}$ used a J-dimensional parametric submodel with a clever covariate corresponding to each of the J-components of the efficient influence curve. And again, this directly solves D^*_Y. Again, some simple re-arrangement shows that the empirical mean of our estimator within each of the J partitions is equal to the empirical mean of Y within each of the J partitions.

B.1.3 EIC for calibration to the intensity at a specific time point $\phi_0(t)$

Under the counting process framework, we defined a time-specific data structure $O(t) = \big(R(t), R(t)\mathscr{F}_t, R(t)Y(t)\big)$ and our full data as $O = \big(O(t) : t = 0, \ldots, \tau\big)$. $S_t = S(\mathscr{F}_t)$ was a summary of the history. The t-specific intensity calibration parameter was

$$\phi_0(t) = E_0[Y(t)|I_{(\mathscr{A})}\{S_t\} = 1, R(t) = 1]$$

Let $D^*_t(P_0)(O(t))$ be the efficient influence curve for $\phi_0(t)$. The form of this is quite similar to that for a single time-point binary outcome discussed above, except that it depends on the history, $\mathscr{F}_t$, and also includes the "at risk" indicator $R(t)$.

$$D^*_t(P_0)(O(t)) = \frac{R(t)}{E_0[R(t)]}\left\{\frac{Y(t)I_{(\mathscr{A})}\{S_t\}}{E_0[I_{(\mathscr{A})}\{S_t\}|R(t)=1]} - \phi_0(t)\right\}$$
$$-\frac{R(t)}{E_0[R(t)]}\frac{E_0[Y(t)I_{(\mathscr{A})}\{S_t\}|R(t)=1]}{E_0[I_{(\mathscr{A})}\{S_t\}|R(t)=1]^2}\left\{I_{(\mathscr{A})}\{S_t\} - E_0[I_{(\mathscr{A})}\{S_t\}|R(t)=1]\right\}$$

This can be decomposed into

$$D^*_{Y,t} = \frac{R(t)I_{(\mathscr{A})}\{S_t\}}{E_0[I_{(\mathscr{A})}\{S_t\}, R(t)]}\left\{Y(t) - E_0[Y(t)|\mathscr{F}_t, R(t)=1]\right\}$$
$$D^*_{\mathscr{F}_t,R(t),1,t} = \frac{R(t)}{E_0[I_{(\mathscr{A})}\{S_t\}, R(t)]}\left\{I_{(\mathscr{A})}\{S_t\}E_0[Y(t)|\mathscr{F}_t, R(t)=1] - \phi_0(t)\right\}$$

$$D^*_{\mathscr{F}_t,R(t),2,t} = -\frac{R(t)}{E_0[R(t)]}\frac{E_0[Y(t)I_{(\mathscr{A})}\{S_t\}|R(t)=1]}{E_0[I_{(\mathscr{A})}\{S_t\}|R(t)=1]^2}\left\{I_{(\mathscr{A})}\{S_t\} - E_0[I_{(\mathscr{A})}\{S_t\}|R(t)=1]\right\}$$

The first term is a score of conditional distribution of $Y(t)$ given $\mathscr{F}_t$ and $R(t) = 1$, while other terms are scores of distribution of the history $\mathscr{F}_t$, given $R(t) = 1$. Note that

$$D^*_t(P_0) = D^*_t(Q_{\mathscr{F}_t,R(t),0}(t), \bar{Q}_{Y(t),0}(t))$$

Note that the t-specific empirical distribution $Q_{\mathscr{F}_t,R(t),n}(t)$ solves the efficient influence curve estimating equations for both $D^*_{\mathscr{F}_t,R(t),1,t}$ and $D^*_{\mathscr{F}_t,R(t),2,t}$ at every t. This implies that these empirical distributions are already targeted towards the estimation of our parameter of interest and no TMLE updating is necessary.

Our TMLE updated $\bar{Q}^*_{Y(t),n}$ directly solves $\frac{1}{n}\sum_{i=1}^n D^*_{Y,t}(Q_{\mathscr{F}_t,R(t),n}(t),\bar{Q}^*_{Y(t),n})(O_i)=0$. And a simple rearrangement shows that the empirical mean of our TMLE estimator is equal to the empirical nonparametric maximum likelihood estimator at time t.

B.1.4 EIC for a function calibration $(\phi_0(t): t=1,\ldots,\tau)$

The efficient influence curve for $(\phi_0(t): t=1,\ldots,\tau)$ can be thought of as a τ-dimensional vector function.

$$D^*(P_0)(O)=\begin{bmatrix} D^*(P_0)(O)(t=1) \\ \vdots \\ D^*(P_0)(O)(t=\tau) \end{bmatrix}$$

Again the same properties discussed above for a specific t also hold for every and all t.

B.1.5 EIC for calibaration to a weighted average (or crude rate) $\bar{\phi}_0$

Recall the weighted average parameter

$$\bar{\phi}_0=\frac{1}{\sum_t E_0[I_{(\mathscr{A})}\{S_t\},R(t)]}\sum_t E_0[I_{(\mathscr{A})}\{S_t\},R(t)]\phi_0(t)$$

Because $\bar{\phi}_0$ is a weighted average of the t specific influence curves for $(\phi_0(t): t=1,\ldots,\tau)$, its influence curve is also a weighted average of the t-specific influence curves for $(\phi_0(t): t=1,\ldots,\tau)$. This follows from the functional delta method.

$$\bar{D^*}=\sum_t\frac{E_0[I_{(\mathscr{A})}\{S_t\},R(t)]}{\sum_t E_0[I_{(\mathscr{A})}\{S_t\},R(t)]}D^*_t$$

And this can be decomposed as

$$\bar{D}^*_Y(P_0)(O)=\sum_t\frac{I_{(\mathscr{A})}\{S_t\}R(t)}{\sum_t E_0[R(t),I_{(\mathscr{A})}\{S_t\}]}\left\{Y(t)-E_0[Y(t)|\mathscr{F}_t,R(t)=1]\right\}$$

$$\bar{D}^*_{\mathscr{F},R,1}=\sum_t\frac{I_{(\mathscr{A})}\{S_t\}R(t)}{\sum_t E_0[R(t),I_{(\mathscr{A})}\{S_t\}]}\left\{I_{(\mathscr{A})}\{S_t\}E_0[Y(t)|\mathscr{F}_t,R(t)=1]-\phi_0(t)\right\}$$

$$\bar{D}^*_{\mathscr{F},R,2}=-\sum_t\frac{I_{(\mathscr{A})}\{S_t\}R(t)}{\sum_t E_0[R(t),I_{(\mathscr{A})}\{S_t\}]}\frac{E_0[Y(t)I_{(\mathscr{A})}\{S_t\}|R(t)=1]}{E_0[I_{(\mathscr{A})}\{S_t\}|R(t)=1]}\left\{I_{(\mathscr{A})}\{S_t\}-E_0[I_{(\mathscr{A})}\{S_t\}|R(t)=1]\right\}$$

Here the t-specific empirical distributions of $(\mathscr{F}_t, R(t))$ solve the estimating equations for $\bar{D^*_{\mathscr{F},R,1}}$ and $\bar{D^*_{\mathscr{F},R,2}}$. These are therefore already targeted towards our $\bar{\phi}_0$ and do not require TMLE updates. Our TMLE update $\bar{Q}^*_{Y,n}$ uses pools observations over all t, so including the simple "clever covariate" $R(t)I_{(\mathscr{A})}\{S_t\}$ solves the estimating equation $\frac{1}{n}\sum_{i=1}^n \bar{D^*_Y}(O_i) = 0$.

A simple rearrangement of this estimating equation yields

$$\frac{\sum_t \frac{1}{n}\sum_{i=1}^n R_i(t)I_{(\mathscr{A})}\{S_{t,i}\}Y_i}{\sum_t \frac{1}{n}\sum_{i=1}^n R_i(t)I_{(\mathscr{A})}\{S_{t,i}\}} = \frac{\sum_t \frac{1}{n}\sum_{i=1}^n R_i(t)I_{(\mathscr{A})}\{S_{t,i}\}\bar{Q}^*_{Y,n}(\mathscr{F}_{t,i}R_i(t))}{\sum_t \frac{1}{n}\sum_{i=1}^n R_i(t)I_{(\mathscr{A})}\{S_{t,i}\}}$$

Or more simply

$$\frac{\sum_t \sum_{i=1}^n R_i(t)I_{(\mathscr{A})}\{S_{t,i}\}Y_i}{\sum_t \sum_{i=1}^n R_i(t)I_{(\mathscr{A})}\{S_{t,i}\}} = \frac{\sum_t \sum_{i=1}^n R_i(t)I_{(\mathscr{A})}\{S_{t,i}\}\bar{Q}^*_{Y,n}(\mathscr{F}_{t,i}R_i(t))}{\sum_t \sum_{i=1}^n R_i(t)I_{(\mathscr{A})}\{S_{t,i}\}}$$

So the weighted average of our TMLE updated conditional intensity estimator will be equal to the "crude rate", which sums over all subjects and time points the total number of observed events and divides by the total amount of observed exposure time. These results also hold for a vector of weighted average parameters, except that now the efficient influence curve is also a vector.

B.2 %TMLE_CALIBRATION

```
*****************************************************************************;
* %TMLE_CALIBRATION                                                         *;
*                                                                           *;
* SAS MACRO FOR FOR TARGETED MAXIMUM LIKELIHOOD ESTIMATION (TMLE) PROCEDURE *;
* TO CALIBRATE A PREDICTION FUNCTION ESTIMATE. THIS CORRESPONDS TO ENFORCING*;
* AN IMPLICIT CONSTRAINT ON THE PREDICTION FUNCTION ESTIMATE ITSELF.        *;
*                                                                           *;
* JORDAN BROOKS                                                             *;
*****************************************************************************;
* TESTED ON SAS 9.2 WINDOWS 64-BIT                                          *;
*****************************************************************************;
* INPUTS TO THE %TMLE_CALIBRATION MACRO:                                    *;
*                                                                           *;
* DSN           = Input SAS dataset. For calibration of the conditional     *;
*                 intensity of a time-dependent counting process, the data  *;
*                 must be in long format, i.e., with one row per time-point *;
*                                                                           *;
* Y             = Event/target variable. Must be NUMERIC coded 0/1.         *;
*                 Missing values are not allowed.                           *;
*                                                                           *;
* Qn_0          = A .sas program file containing the initial prediction     *;
*                 function estimator mapping.                               *;
*                                                                           *;
* Yhat_0        = The variable name for the initial prediction estimate, i.e., *;
*                 the output of Qn_0.                                       *;
*                                                                           *;
* CUTS          = Cut points that define intervals on the range of Q_Yn. The *;
*                 resulting continguous intervals define the data partitioning *;
*                 subgroups for the calibration parameter.                  *;
*                                                                           *;
* Qn_star      = A .sas program file containing the (TMLE updated) calibrated  *;
*                prediction function estimator mapping. The name of the output *;
*                variable for the calibrated prediction estimates is the value *;
*                of "Yhat_0"_star.                                             *;
*                                                                              *;
* MINEPS       = The minimum value of the L1-norm of the epsilon parameter in  *;
*                the TMLE updates to imply convergence. That is, the TMLE will *;
*                iterate until the epsilon estimate is sufficiently close to 0.*;
*                                                                              *;
* LOWER_BOUND  = Lower bound for predictions/estimates. Default is 0.00000001. *;
*                                                                              *;
* UPPER_BOUND  = Upper bound for predictions/estimates. Default is 0.99999999. *;
********************************************************************************;

%MACRO TMLE_CALIBRATION(DSN, Y, Qn_0, Yhat_0, Qn_star, Yhat_star,
CUTS, MINEPS,
LOWER_BOUND=0.00000001, UPPER_BOUND=0.99999999);

**********************************************************;
* COUNT AND INDEX CUT-POINTS TO FORM CALIBRATION SUBGROUPS *;
**********************************************************;
%LET ncut = %sysfunc(countw("&CUTS", " "));
%LET ngrp = %EVAL(&ncut + 1);
%DO i = 1 %TO &ncut;
 %LET CUTS_&i = %SCAN("&CUTS", &i, " ");
%END;

****************************************;
* COMPUTE INITIAL ESTIMATOR PREDICTIONS *;
****************************************;
data _train;
 set &dsn;
 %include "&Qn_0";
 Yhat_0 = &Yhat_0;
 keep &Y Yhat_0;
run;

**************************;
* ITERATIVE TMLE UPDATES *;
**************************;
%LET sumabseps=100000;
%LET k=0;
%DO %WHILE (&sumabseps > &MINEPS);
%LET k = %EVAL(&k+1);

*****************************************************************;
* COMPUTE _H (CLEVER COVARIATE) - CALIBRATION SUBGROUP INDICATORS *;
*****************************************************************;
data _train;
 set _train;
 Yhat_%EVAL(&k-1) = max(Yhat_%EVAL(&k-1), &LOWER_BOUND);
 Yhat_%EVAL(&k-1) = min(Yhat_%EVAL(&k-1), &UPPER_BOUND);
 array CUTS[&ncut] _TEMPORARY_ (&CUTS);
 if Yhat_%EVAL(&k-1) ne . then _H = 1;
 do i = 1 to &ncut;
  if Yhat_%EVAL(&k-1) > CUTS{i} then _H =i+1;
 end;
 array _Harray{%EVAL(&ncut+1)} _H1 - _H%EVAL(&ncut+1);
 do i = 1 to %EVAL(&ncut+1);
  _Harray{i}=0;
  if _H = i then _Harray{i} = 1;
 end;
 logitYhat = log(Yhat_%EVAL(&k-1)/(1-Yhat_%EVAL(&k-1)));
 drop i;
run;

***************;
* FIT EPSILON *;
***************;
proc logistic descending data=_train outest=_eps;
 model &Y = _H1 - _H%EVAL(&ncut+1) / noint offset=logitYhat;
run;

*********************************************;
* SAVE FITTED EPSILON TO MACRO VARIABLE, AND *;
* COMPUTE MAGNITUDE OF FITTED EPSILON        *;
*********************************************;
data _null_;
 set _eps;
 array _Harray{%EVAL(&ncut+1)} _H1 - _H%EVAL(&ncut+1);
 do i = 1 to %EVAL(&ncut+1);
  _Harray{i} = round(_Harray{i},0.0001);
 end;
 %DO i = 1 %TO %EVAL(&ncut+1);
  call symputx("eps&k._&i", _H&i);
 %END;
 do i = 1 to %EVAL(&ncut+1);
  _Harray{i} = abs(_Harray{i});
 end;
 sumabseps = round(sum(OF _H1 - _H%EVAL(&ncut+1)),0.00001);
 call symputx("sumabseps", sumabseps);
run;

***************************;
* PREDICT ON TRAINING DATA *;
***************************;
data _train;
 set _train;
 Yhat_&k = 1/(1+exp(-1*(logitYhat
   %DO hhh=1 %TO %EVAL(&ncut+1); + _H&hhh * &&eps&k._&hhh  %END;
          )));
run;

%END;

************************************************;
* SAVE CALIBRATED ESTIMATOR TO NEW SAS PRORGAM *;
************************************************;
data _null_;
 infile "&Qn_0";
 input;
 file "&Qn_star";
 put _infile_;
run;
data _null_;
 file "&Qn_star" MOD;
 put "Yhat_0 = &Yhat_0;";
 %DO j = 1 %TO &k;
  put "Yhat_%EVAL(&j-1) = max(Yhat_%EVAL(&j-1), &LOWER_BOUND);";
  put "Yhat_%EVAL(&j-1) = min(Yhat_%EVAL(&j-1), &UPPER_BOUND);";
  put "logitYhat = log(Yhat_%EVAL(&j-1)/(1-Yhat_%EVAL(&j-1)));";
  put "if Yhat_%EVAL(&j-1) ne . then _H = 1;";
  %DO i = 1 %TO &ncut;
   put " if Yhat_%EVAL(&j-1) > &&CUTS_&i then _H=&i+1;";
  %END;
  %DO i = 1 %TO &ngrp;
   put " logitYhat = logitYhat + (_H=&i)*&&eps&j._&i;";
  %END;
  put "Yhat_&j = 1/(1+exp(-1*logitYhat));";
  if &j = &k then put "&Yhat_star = 1/(1+exp(-1*logitYhat));";
 %END;
 put "keep &Yhat_star;";
run;
***************************;
* CLEAN UP WORK DIRECTORY *;
***************************;
proc datasets lib=work; delete _: ; quit;

%MEND TMLE_CALIBRATION;
```

Appendix C

Supplement for TMLE for intervention-specific survival parameters

C.1 Efficient Influence Curve for intervention-specific mean

The efficient influence curve for the intervention-specific marginal mean in a longitudinal data structure is

$$D^*_{Q_0,g_0}(O) = \frac{I(\bar{A}(K)=\bar{a}(K))}{g_{0,K}}\left(L_1(K+1) - \bar{Q}_{Y|\bar{L}(K)}\right) + \sum_{k=0}^{K-1} \frac{I(\bar{A}(k)=\bar{a}(k))}{g_{0,k}}\left(\bar{Q}_{Y|\bar{L}(k+1)} - \bar{Q}_{Y|\bar{L}(k)}\right) + \bar{Q}_{Y|L(0)} - \psi_0$$

The derivation is given in (van der Laan and Gruber, 2011), and reproduced here for convenience. Denote an influence curve $D^{IPCW}_{Q,g} = \frac{I(\bar{A}(K)=\bar{a}(K))}{g_{0,K}}Y - \psi_0$. This is a gradient of the pathwise derivative of Ψ in the model in which g_0 is known. The efficient influence curve is $D^* = \sum_{k=0}^{K} D^*_k$, where $D^*_k = \Pi(D|T_k)$ is the projection of D onto the tangent space $T_k = \{h(L(k)|Pa(L(k)) : E_Q[h|Pa(L(k))] = 0\}$ of $\bar{Q}^{\bar{a}}_{Y|\bar{L}(k)}$ in the Hilbert space $L^2_0(P)$ with inner-product $< h_1, h_2 >_P = E_0[h_1 h_2]$. Recall that $\bar{Q}_{Y|\bar{L}(k)} = E[Y^{\bar{a}}|\bar{L}(k)]$, and the recursive relationship $Q_{Y|\bar{L}(k-1)} = E[\bar{Q}_{Y|\bar{L}(k)}|\bar{Q}_{Y|\bar{L}(k-1)}$. At the terminal time point K the projection is

$$D^*_{K+1} = \frac{I(\bar{A}(K)=\bar{a}(K))}{g_{0,K}}\left(L_1(K+1) - \bar{Q}^{\bar{a}}_{Y|\bar{L}(K)}\right)$$

At the penultimate time-point the projection is

$$\begin{aligned} D^*_K &= \frac{I(\bar{A}(K-1)=\bar{a}(K-1))}{g_{0,K-1}}\left(\bar{Q}^{\bar{a}}_{Y|\bar{L}(K)} - E[Q^{\bar{a}}_{Y|\bar{L}(K)}|Q^{\bar{a}}_{Y|\bar{L}(K-1)}\right) \\ &= \frac{I(\bar{A}(K-1)=\bar{a}(K-1))}{g_{0,K-1}}\left(\bar{Q}^{\bar{a}}_{Y|\bar{L}(K)} - \bar{Q}^{\bar{a}}_{Y|\bar{L}(K-1)}\right) \end{aligned}$$

Finally at the zeroeth time-point we get

$$\bar{Q}^{\bar{a}}_{Y|\bar{L}(0)} - E[Q^{\bar{a}}_{Y|\bar{L}(0)}] = \bar{Q}^{\bar{a}}_{Y|\bar{L}(0)} - \Psi(Q^{\bar{a}})$$

For the formal proof, see van der Laan and Gruber (2011).

C.2 %TMLE_RCS

```
******************************************************************************;
* %TMLE_RCS                                                                  *;
*                                                                            *;
* SAS MACRO FOR FOR TARGETED MARXIMUM LIKELIHOOD ESTIMATION OF THE           *;
* FIXED INTERVENTION-SPECIFIC MARGINAL CUMULATIVE EVENT PROBABILITY IN       *;
* SURVIVAL DATA WITH TIME-DEPENDENT COVARIATES AND WITH INFORMATIVE RIGHT-   *;
* CENSORING.                                                                 *;
*                                                                            *;
* REQUIRES %SUPERLEARNER SAS MACRO FOR THE INITIAL Q-FACTOR ESTIMATORS       *;
*                                                                            *;
* JORDAN BROOKS                                                              *;
*                                                                            *;
******************************************************************************;
* TESTED ON SAS 9.2 WINDOWS 64-BIT                                           *;
******************************************************************************;
* INPUTS TO THE %TMLE_RCS MACRO:                                             *;
*                                                                            *;
* DSN                = Input SAS dataset containing observations in long     *;
*                      format, i.e., one row per subject per time-point.     *;
*                                                                            *;
* ID                 = Name of the variable that uniquely identifies         *;
*                      independent observations/subjects.                    *;
*                                                                            *;
* t                  = Name of the time-stamp variable. The first row for each *;
*                      subject has t=0.                                      *;
*                                                                            *;
* L1                 = The event counting process. This is really L1(t+1), a  *;
*                      random variable that takes value 0 until the time-point *;
*                      just before the event occurs. The first row for each  *;
*                      subject is then L1(0+1), the indicator that the event *;
*                      process jumps at time t=1. Once the event process jumps *;
*                      It stays at 1 for all remaining time-points.          *;
*                      Must be NUMERIC coded 0/1.                            *;
*                      Missing values are not allowed.                       *;
*                                                                            *;
* L2                 = The time-dependent covariate process history. This is *;
*                      is really L2(t), a vector of random variables that    *;
*                      represent the covariate history up to and including  *;
*                      time-point t. All variables here must be NUMERIC.     *;
*                      Missing values are not allowed.                       *;
*                                                                            *;
* A                  = The fixed treatment. Must be a NUMERIC coded 0/1.     *;
*                      Missing values are not allowed.                       *;
*                                                                            *;
* C                  = The censoring process. Really this is C(t), a random  *;
*                      variable that takes value 0 at every time-point before *;
*                      censoring occurs. When censoring occurs, the process  *;
*                      jumps to C(t)=1, and stays at 1 for all reamining time- *;
*                      points. Must be NUMERIC coded 0/1.                    *;
*                      Missing values are not allowed.                       *;
*                                                                            *;
* K                  = The pen-ultimate time-point of interest. That is, the   *;
*                      parameter of interest is the marginal intervention-     *;
*                      specific cumulative event probability by time t=K+1.    *;
*                                                                            *;
* setA               = The fixed treatment intervention of interest.           *;
*                                                                            *;
* g_An               = A .sas program with DATA STEP code to compute the fixed *;
*                      treatment probabilities. This is a function of L2(0).   *;
*                                                                            *;
* g_dCn              = A .sas program file with DATA STEP code to compute the  *;
*                      censoring intensity esimtates.                          *;
*                      This is a function of L2, A, and t.                     *;
*                                                                            *;
* g_LOWER_BOUND      = Lower bound for g estimates. The default is 0.001       *;
*                                                                            *;
* g_LOWER_BOUND      = Lower bound for g estimates. The default is 0.999       *;
*                                                                            *;
* Q_dL1n             = A .sas program file containing the estimator mapping    *;
*                      for the event intensity/hazard. This is a function of   *;
*                      as a function L2, A, and t. This is used as the initial *;
*                      estimator of the ultimate Q-factor,                     *;
*                      Qbar_(K) = E[L1(K+1)|setA, L2(t=K)], in the TMLE.       *;
*                                                                            *;
* Q_LOWER_BOUND      = Lower bound for Qbar_(k) = E[L1(K+1)|setA, L2(t)]       *;
*                      The default is 0.00000001                               *;
*                                                                            *;
* Q_UPPER_BOUND      = Upper bound for Qbar_(k) = E[L1(K+1)|setA, L2(t)]       *;
*                      The default is 0.99999999                               *;
*                                                                            *;
* SL_LIBRARY         = Names of SAS macros for candidate estimators in the     *;
*                      Super Learner library. The macros must be saved under   *;
*                      the exact name of candidate estimator as a .sas program *;
*                      in the filepath given in LIBRARY_DIR.                   *;
*                                                                            *;
* LIBRARY_DIR        = Filepath for directory containing the candidate         *;
*                      estimator macros, which are saved as .sas programs.     *;
*                                                                            *;
* EnterpriseMiner    = Enter T/F to indicate use of SAS/EnterpriseMiner. If    *;
*                      set to T, a SAS datamining database catalog is          *;
*                      constructed.                                            *;
*                                                                            *;
* V                  = The number of folds for V-fold crossvalidation.         *;
*                                                                            *;
* SEED               = The SEED for the random number generator.               *;
*                                                                            *;
* FOLD (optional)    = Variable containing the fold number for cross           *;
*                      validation in the Super Learner.                        *;
*                                                                            *;
* VERBOSE (optional) = A T/F that indicates whether all output normally        *;
*                      produced by SAS data and procedure steps should be      *;
*                      sent to the listing device. The default is F.           *;
*                                                                            *;
* WD                 = The working directory. Initial estimator and TMLE       *;
*                      updated fits will be saved to this directory as .sas    *;
*                      programs. This directory will be assigned a SAS libname *;
*                      "SL" during each fitting process.                       *;
*******************************************************************************;

%MACRO TMLE_RCS(DSN, ID, t, L1, L2, A, C, K, setA,
                g_An, g_dCn, g_LOWER_BOUND, g_UPPER_BOUND,
                Q_dL1n, Q_LOWER_BOUND, Q_UPPER_BOUND,
                SL_LIBRARY, LIBRARY_DIR, EnterpriseMiner, V, SEED, FOLD, VERBOSE,
                WD);

ods listing close;

***************************************************************************;
* AUGMENT DATASET TO INCLUDE ONE ROW PER SUBJECT PER TIME-POINT UNTIL t=K *;
***************************************************************************;
data tmle;
 set &DSN (where=(&t <= &K) keep = &ID &t &L1 &L2 &A &C);
 by &ID &t;
 dead=0;
 output;
 if last.&ID and &t lt &K then do until (&t=&K);
  &t = &t+1;
  dead = &L1;
  output;
 end;
 drop _:;
run;

******************************************************************************;
* COMPUTE g (treatment and censoring) AND Qbar_K (event intensity) ESTIMATES *;
******************************************************************************;
data tmle;
 set tmle;
 by &ID &t;
 retain g_An;
 if first.&ID then do;
  %include "&g_An";
 end;
 keep &ID &t &L1 &L2 &A &C g_An dead;
run;
data tmle;
 set tmle;
 by &ID &t;
 %include "&g_dCn";
 keep &ID &t &L1 &L2 &A &C g_An g_dCn dead;
run;
data tmle;
 set tmle;
 by &ID &t;
 %include "&Q_dL1n";
 retain g;
 if first.&ID then do;
  if &A=1 then g=g_An;
  if &A=0 then g=(1-g_An);
 end;
 g=g*(1-g_dCn);
 g = max(g, &g_LOWER_BOUND);
 g = min(g, &g_UPPER_BOUND);
 H = (&A=&setA and &C=0)/g;
 Q_dL1n = Q_dL1n;
 keep &ID &t &L1 &L2 &A &C g H Q_dL1n dead;
run;

******************************************;
* SORT DESCENDING BY t FOR TMLE ALGORITHM *;
******************************************;
proc sort data=tmle;
 by &ID descending &t;
run;
```

```
************************************************************************;
* DEFINE ACTIVE SET OF UNITS THAT HAD OBSERVED INTERVENTION OF INTEREST *;
************************************************************************;
%LET step = &K;
%LET active = ((&t=&K) and (&A=&setA) and (&C=0)) ;
title2 "TMLE ALGORITHM STEP &K";

*************************************************************;
* COMPUTE ESTIMATE OF Qbar_(K) = E[Y=L(K+1)|setA, L2(t=K)] *;
* IF EVENT HAS OCCURRED PRIOR CURRENT DAY THEN Qbar_(K)=1  *;
*************************************************************;
data tmle;
 set tmle;
 by &ID descending &t;
 if ((&t=&K) and (&A=&setA)) then do;
  Qbar=(Q_dL1n);
  if (dead) then Qbar=1;
 end;
run;

*******************;
* FIT epsilon_(K) *;
*******************;
proc nlmixed data=tmle(where=((&active) and (dead=0)));
 parms eps=0;
 Qbar_star = 1/(1+exp(-1*(log(Qbar/(1-Qbar)) + eps)));
 logL = 2*H*(&L1*log(Qbar_star)+(1-&L1)*log(1-Qbar_star));
 model &L1 ~ general(logL);
 ods output ParameterEstimates=_eps;
run;
data _null_;
 set _eps;
 call symputx("eps", Estimate);
run;
proc datasets lib=work; delete _: ; quit;
data eps;
 t = %EVAL(&K+1);
 &A = &setA;
 eps = &eps;
 output;
run;

*******************;
* UPDATE Qbar_(K) *;
*******************;
data tmle;
 set tmle;
 by &ID descending &t;
 if ((&t=&K) and (&A=&setA)) then do;
  if (dead=1) then Qbar_Star=1;
  else Qbar_star = 1/(1+exp(-1*(log(Qbar/(1-Qbar)) + &eps)));
 end;
run;

******************************************;
* COMPUTE INFLUENCE CURVE COMPONENT D_(K) *;
******************************************;
data tmle;
 set tmle;
 if &t=&K then do;
  D=0;
  if (H ne 0) then do;
   D = H*(&L1-Qbar_star);
  end;
 end;
run;

***************;
* NOW ITERATE *;
***************;
%DO i = 1 %to %sysevalf(&K);

*********************************************************************;
* DEFINE ACTIVE SET OF UNITS THAT HAD OBSERVED INTERVENTION OF INTEREST *;
*********************************************************************;
%LET step = %EVAL(&K-&i);
%LET active = ((&t=&K-&i) and (&A=&setA) and (&C=0)) ;
title2 "TMLE ALGORITHM STEP &step" ;

***************************************************;
* DEFINE PREVIOUS CONDITIONAL EXPECTATION AS OUTCOME *;
***************************************************;
data tmle;
 set tmle;
 by &ID descending &t;
 lagQbar_star=lag(Qbar_star);
run;

*****************************************************************************;
* SUPER LEARNER FOR INITIAL ESTIMATOR Qbar_(K-&i) = E[Qbar_(K-&i+1)|setA, L2(K-&i)] *;
*****************************************************************************;
data temp; set tmle(where=((&active) and (dead=0))); run;
%SUPERLEARNER(TRAIN=temp, TEST=,
              Y=lagQbar_star, Y_TYPE=CTS, X=&L2, ID=&ID, T=, WEIGHTS=,
              SL_LIBRARY=&SL_LIBRARY,
              LIBRARY_DIR=&LIBRARY_DIR,
              EnterpriseMiner=&EnterpriseMiner,
              LOWER_BOUND=&Q_LOWER_BOUND, UPPER_BOUND=&Q_UPPER_BOUND,
              LOSS=LOG, V=&V, SEED=&SEED, FOLD=&FOLD,
              WD=&WD, VERBOSE=F);
proc datasets kill lib=sl; run; quit;
data _null_;
  fname="fname";
  rc=filename(fname,"&WD\F_SuperLearner_&setA._&step..sas");
  if fexist(fname) then rc=fdelete(fname);
  rc=rename("&WD\F_SuperLearner.sas","&WD\F_SuperLearner_&setA._&step..sas","file");
run;

**********************************************************************;
* COMPUTE INITIAL ESTIMATE Qbar_(K-&i) = E[Qbar_(K-&i+1)|setA, L2(K-&i)] *;
**********************************************************************;
data tmle;
 set tmle;
 by &ID descending &t;
 %include "&WD\F_SuperLearner_&setA._&step..sas";
 if ((&t=&K-&i) and (&A=&setA)) then do;
  if (dead) then Qbar=1;
  else Qbar = p_SL;
 end;
run;

*********************;
* FIT epsilon_(K-&i) *;
```

```
*********************;
proc nlmixed data=tmle(where=((&active) and (dead=0)));
 parms eps=0;
 Qbar_star = 1/(1+exp(-1*(log(Qbar/(1-Qbar)) + eps)));
 logL = 2*H*(lagQbar_star*log(Qbar_star)+(1-lagQbar_star)*log(1-Qbar_star));
 model lagQbar_star ~ general(logL);
 ods output ParameterEstimates=_eps;
run;
data _null_;
 set _eps;
 call symputx("eps_&step", Estimate);
run;
data _eps;
 t = &step;
 &A = &setA;
 eps = &&eps_&step;
 output;
run;
proc append base=eps data=_eps; run;
proc datasets lib=work; delete _:; quit;

*********************;
* UPDATE Qbar_(K-&i) *;
*********************;
data tmle;
 set tmle;
 if ((&t=&K-&i) and (&A=&setA)) then do;
  if (dead) then Qbar_Star=1;
  else Qbar_star = 1/(1+exp(-1*(log(Qbar/(1-Qbar)) + &&eps_&step)));
 end;
run;

********************************************;
* COMPUTE INFLUENCE CURVE COMPONENT D_(K-&i) *;
********************************************;
data tmle;
 set tmle;
 lagD = lag(D);
 if &t=&K-&i then do;
  D=lagD;
  if (H ne 0) and (lagQbar_star ne .) then do;
   D = D + H*(lagQbar_star-Qbar_star);
  end;
 end;
run;

%END;

************************************************************************;
* SET A and C FOR ALL SUBJECTS AT t=0 AND COMPUTE COUNTERFACTUAL ESTIMATES *;
************************************************************************;
data cf;
 set tmle (keep = &ID &t &L1 &L2 D where=(&t=0));
 &A=&setA;
 &C=0;
 %include "&g_dCn";
 %IF &setA = 1 %THEN %DO; H = 1/ ((g_An)*(1-g_dCn)); %END;
 %IF &setA = 0 %THEN %DO; H = 1/ ((1-g_An)*(1-g_dCn)); %END;
run;
data cf;
 set cf;
 %include "&WD\f_SuperLearner_&setA._0.sas";
 Qbar = p_SL;
 linpart = log(Qbar/(1-Qbar)) + &eps;
 if linpart < -700 then Qbar_star = 0;
 else if linpart > 700 then Qbar_star = 1;
 else Qbar_star = 1/(1+exp(-1*(linpart)));
run;

*********************************************************************;
* COMPUTE TMLE AS THE EMPIRCAL MEAN OF THE COUNTERFACTUAL ESTIMATES *;
*********************************************************************;
proc means data=cf noprint;
 var Qbar_star;
 output out=cf_stats;
run;
data _null_;
 set cf_stats;
 if _STAT_="MEAN" then do;
  call symputx("n", _FREQ_);
  call symputx("tmle", Qbar_star);
 end;
run;

*****************************************************;
* COMPUTE STANDARD ERROR BASED ON THE INFLUENCE CURVE *;
*****************************************************;
data Dtmle;
 set tmle (keep = &ID D Qbar_star);
 Dtmle = D + Qbar_star - &tmle;
run;
proc means data=Dtmle noprint;
 var Dtmle;
 output out=Dtmle_stats;
run;
data _null_;
 set Dtmle_stats;
 if _STAT_="STD" then do;
  call symputx("se_tmle", Dtmle/sqrt(&n));
 end;
run;

*******************************************;
* PRINT TMLE AND STANDARD ERROR ESTIMATES *;
*******************************************;
data _null_;
 file print;
 put "Marginal cumulative probability of &L1=1 by time &K";
 put " under fixed treatment: &A = &setA";
 put "Sample size = &n";
 put "g-factors bounded = [&g_LOWER_BOUND, &g_UPPER_BOUND]";
 put "TMLE (SE)  = &tmle (&se_tmle)";
run;

%MEND TMLE_RCS;
```