Targeted Maximum Likelihood Estimation and Ensemble Learning for Community-Level Data and Healthcare Claims Data

by

Chi Zhang

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

Committee in charge:

Professor Mark J. van der Laan, Chair
Professor Alan E. Hubbard
Associate Professor Jennifer Ahern

Spring 2019

Targeted Maximum Likelihood Estimation and Ensemble Learning for Community-Level Data and Healthcare Claims Data

Abstract

Targeted Maximum Likelihood Estimation and Ensemble Learning for Community-Level Data and Healthcare Claims Data

by

Chi Zhang

Doctor of Philosophy in Biostatistics

University of California, Berkeley

Professor Mark J. van der Laan, Chair

This dissertation discusses the Targeted maximum Likelihood Estimation (TMLE) and ensemble learning for community-level data and healthcare claims data, along with the conduct of simulation studies and practical examples for causal inference research in medical data. Specifically, we resolve two common questions: how to estimate the community-based causal effect of community-level stochastic interventions, and how to take advantage of data-adaptive ensemble learning to problems of estimation in public health data.

Chapter 1 begins by reviewing the targeted maximum likelihood estimation (TMLE). We also provide a more detailed summary to each of the rest of the chapters.

Chapter 2 studies the framework for target maximum likelihood estimation and statistical inference for the causal effects of community-level treatments on individual-level outcomes where the outcomes could be correlated because of the interactions among individuals from the same communities and the shared community-level covariates. This chapter presents a new solution that considers the case in which the treatment mechanism may cause stochastically assigned exposures and the corresponding causal parameter may require a more easily achievable positivity assumption. Given two different structural equation models, we develop two semi-parametric efficient TMLEs for the estimation of such a community-based causal effect. The proposed TMLEs have several crucial advantages. First, both TMLEs can make use of individual level data in the hierarchical setting, and potentially reduce finite sample bias and improve estimator efficiency. Second, the stochastic intervention framework provides a natural way for defining and estimating casual effects where the exposure variables are continuous or discrete with multiple levels, or even cannot be directly intervened on. Also, the positivity assumption needed for our proposed causal parameters can be weaker than the version of positivity required for other casual parameters.

Chapter 3 builds on the work described in Chapter 2 and presents an open-source software tool for implementing TMLE of the average causal effect of community-level intervention(s) at a single time point. This software supports a wide variety of TMLE implementations. For example, the package supports univariate or multivariate arbitrary (i.e., static, dynamic

or stochastic) interventions with a binary or continuous outcome. It also allows users to use either weighted intercept-based TMLE or unweighted covariate-based TMLE.

In Chapter 4, we propose a new ensemble approach to gain a better understanding of the natural history of nonalcoholic steatohepatitis (NASH). Super Learner (SL) is an ensemble method that uses V-folds cross-validation to build the optimal weighted combination of the predicted values from a library of user-specified prediction algorithms. Because data-adaptive methods are allowed in a SL library, SL can be used to avoid unrealistic parametric assumptions without overfitting the data in practice. This proposed AUC-maximizing ensemble approach couples each prediction model with a comprehensive feature selection algorithm, including Bayesian risk ratio method, column sparsity based regularization, and L1 regularization.

To my beloved parents Mao and Daining,

my lovely cousin Liyuan, my wise grandmother Junhu,

my other family members and friends, for their love and support throughout my life

Contents

List of Figures

List of Tables

Acknowledgments

During my doctoral research, I have benefited from the advice, help, and support of numerous people. I would like first to express my deepest gratitude and appreciation to my research advisor Mark van der Laan for his help and support during my time at Berkeley. Throughout my coursework and during the writing of my dissertation, Mark has patiently and generously trained me as a scholar and equipped me with research skills. Thank you, Mark, for giving me the opportunity to join your research group, and the insightful comments and critiques that inspired this dissertation and pushed me to develop and refine my thinking. I consider myself incredibly lucky to be one of your students and to work with many amazing individuals in our group.

I am extremely grateful to my dissertation committee, which spent time reading my prospectus and provided valuable insights. I want to thank Alan Hubbard for his support and encouragement throughout my years at Berkeley. Without you, I would not have the opportunity to work for Mark in the beginning and intern at Gilead Sciences. I also would like to thank Jennifer Ahern who involved me in her research project and asked priceless questions regarding my research. Jennifer not only spent time discussing about research questions but also offered me emotional supports when I needed. I will always remember the times when we sit together in your office to talk about family. I am also thankful to Maya Peterson who served on my qualifying exam committee, for her thoughtful criticisms and for her enthusiastic support.

Aside from my committee members, I would also like to thank other faculty I interacted with during my time at Berkeley. I have learned so much from them in lectures, seminars, and in personal discussions. My special thanks go to Ani Adhikari who made this opportunity to do a biostatistics PhD possible in the first place by supporting me in transferring to the program.

I am greatly indebted to the extraordinarily dedicated staff of the biostatistics, statistics and economics department, especially La Shana Porlaris, Sharon Norris and Heather Iwata. I would not have survived graduate school were it not for your administrative and professional support.

Oleg A Sofrygin, Ivan Diaz, Laura B. Balzer, Anand Chokkalingam, Ellie Matthay, Dana Goin, Laura Telep, and Robertino Mera have also played important roles in my research on different projects. I am thankful to have had the opportunity to work with Jim Pitman, Haiyan Huang, Hank Ibser, Helmut Pitters, Shobhana Stoyanov in many pleasing experience I have had teaching both undergraduate and graduate statistics.

My Journey at Berkeley would not have been so happy without being surrounded by my lovely fellow students and friends, including Yang Hu, Yujia Zhang, Su, Liu, Shengqin Su, Yizhou Guo, Su Liu, Yuan He, Chaoran Guo, Yuting Ye, Fengshi Niu, Yanqiao Wang, Lei Kang, Jin Rou New, Mary Combs, Ivana Malenica, Simon Walter, Jonathan Levy, and many others. I will always treasure the time that we spent together and the memories that we made.

Berkeley offered me great opportunities to explore my interests beyond statistics and academia. I have been so fortunate to have met many people along the way who are passionate about what they are doing and driven to make the world a better place. Words are powerless to express my gratitude to everyone who has contributed to my professional and personal development. I fondly acknowledge all those people who helped me at different phases of my five years at Berkeley. I cannot imagine where I would be now without your support and company. I really want to thank my landlord Xan Joi who offered me a great place to stay in the last four years and taught me all the valuable life lessons.

Finally, I thank my family. All of you have given me so much unconditional love, patience and support. Thank you for everything you have ever done for me and ever taught me. I am truly blessed with the most amazing family in the world. I love you all.

Chapter 1

Introduction

1.1 Targeted maximum likelihood estimation

The objective of this section is to provide an intuitive explanation of certain elements of the TMLE and a succinct summary of how it works. We are not going to present a comprehensive or rigorous review. For more details we refer to [78] and [79].

van der Laan and Rubin [79] first proposed the theory of the targeted maximum likelihood estimator (TMLE), which combines the favorable characteristics of estimating equation methods with those of plugin substitution estimators. For example, plugin substitution estimators, such as maximum-likelihood- based substitution estimators of the g-formula (MLE), would respect the global constraints of the statistical model, while the estimating equation methods, such as inverse probability of treatment-weighted estimators (IPTW), are regular and asymptotically linear.

The TMLE is an asymptotically efficient substitution estimator of a target parameter of interest $\Psi(P_0)$ of a true data distribution $P_0 \in \mathcal{M}$, where $\mathcal{M}$ is the statistical model that contains possible probability distributions. We assume that Ψ is finite dimensional pathwise differentiable at each $P \in \mathcal{M}$ with the canonical gradient $D^*(P)$, also known as the efficient influence curve (EIC). That is, for each of the parametric working submodels $\mathcal{M}_\epsilon = \{P_\epsilon : \epsilon\} \in \mathcal{M}$ that covers $\mathcal{M}$ and satisfies $P_{\epsilon=0} = P_0$, and its score $S = \frac{d}{d\epsilon} \log P(\epsilon)|_{\epsilon=0}$ at $\epsilon = 0$, we have

$$\frac{d}{d\epsilon}\Psi(P_\epsilon)|_{\epsilon=0} = P_0 v S$$

where v is called a gradient of the pathwise derivative, and $Pf = \int f((u)dP(u)$. $D^*(P)$ is the only gradient of the pathwise derivative that is an element of the tangent space $T(P)$, which is defined as the closure of the linear span of all scores of submodels through P in the Hilbert space $L_0^2(P)$. Also, $D^*(P)$ has a mean of zero, and its variance is a generalized Cramer-Rao lower bound for the variance of locally unbiased estimator of $\Psi(P_0)$, which is, at a minimum, the inverse of the Fisher information.

Without loss of generality, assume that there is no hierarchical structure in data, and so the estimation is based on n observed independent and identically distributed copies of a

random variable $O = (W, A, Y) \sim P_0$. Then the true joint density p_0 of O can be factorized as

$$p_0(O) = p_{W,0}(W)p_{A|W,0}(A|W)p_{Y|A,W,0}(Y|A,W) \equiv Q_{W,0}(W)g_0(A|W)\bar{Q}_0(A,W)$$

This target parameter only depends on P_0 through a relevant Q-factor, i.e., $\Psi(P_0) = \Psi(Q_0)$, where $Q_0 = Q(P_0) \equiv (\bar{Q}_0, Q_{W,0})$, and the remaining factor, $g_0 = g(P_0)$ is a nuisance parameter. Recall that, an estimator $\hat{\Psi}(P_n)$ of $\Psi(P_0)$, that maps the empirical probability distribution P_n of $O_1, \ldots, O_n$ into a value for the parameter it targets, is asymptotically linear with influence curve $IC(O)$ if it satisfies

$$\sqrt{n}(\hat{\Psi}(P_n) - \Psi(P_0)) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} IC(O_i) + o_p(1)$$

and an estimator is asymptotically efficient if and only if its influence curve is equivalent to the efficient influence curve, i.e., $IC(O) = D^*(O)$.

In gerenral, a TMLE estimator can be constructed by the following five steps (assume that the outcome is either binary or bounded continuous):

1. **Estimating** the outcome mechanism $\bar{Q}_0$.

   As an initial estimator $\bar{Q}_n$ of $\bar{Q}_0$, we can simply regress the outcome Y onto the exposure and baseline covariates (A, W), using the negative log-likelihood loss function:

   $$-\mathcal{L}(\bar{Q})(O) = Y \log \bar{Q}(A,W) + (1-Y)\log(1 - \bar{Q}(A,W))$$

   so that $\bar{Q}_0 = \arg\min_{\bar{Q}} E_0 \mathcal{L}(\bar{Q})(O)$.

2. **Estimating** the treatment mechanism g_0.

   To derive an estimator g_n of g_0 that will be used in the targeting step, we can simply regress the exposure A onto the covariates W. We could, for example, use the negative log-likelihood loss function again if the exposure is binary.

3. **Constructing** a "clever covariate" $H_n(A, W)$.

   Here, the clever covariate is used to define a parametric working submodel that fluctuates the initial estimator $\bar{Q}_n$ to remove bias for $\Psi(P_0)$ in the targeting step. It's called the "clever covariate" since it defines the fluctuation direction. For example, for the average treatment effect (ATE) parameter, the clever covariate can be defined as

   $$H_n(A,W) = \frac{A}{g_n(1|W)} - \frac{1-A}{g_n(0|W)} = \frac{2A-1}{g_n(A|W)}$$

4. **Updating** the initial estimator $\bar{Q}_n$.

Fluctuate the initial estimator $\bar{Q}_n(A,W)$ with a parametric submodel, indexed by the univariate parameter ϵ:

$$logit(\bar{Q}_n^*(A,W)) = logit(\bar{Q}_n(A,W)) + \epsilon_n H_n(A,W)$$

where $logit(x) = \log(\frac{x}{1-x})$, and $logit(\bar{Q}_n(A,W))$ is served as a fixed offset and ϵ_n is the resulting coefficient in front of the clever covariate. The amount of fluctuation is determnined by minimizing the empirical loss function:

$$\epsilon_n = \arg\min_{\epsilon} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\bar{Q}_n(\epsilon))(O_i)$$

where $\mathcal{L}(\bar{Q}_n(\epsilon)(O_i)$ could be user-specified, for example, a negative Bernoulli log-likelihood loss function:

$$-\mathcal{L}(\bar{Q}_n(\epsilon))(O_i) = Y_i \log[\bar{Q}_n(\epsilon)(A_i,W_i)] + (1-Y_i)\log[(1-\bar{Q}_n(\epsilon)(A_i,W_i))]$$

5. **Computing** the TMLE estimator.

   The plug-in TMLE estimator is computed by using the updated estimate $\bar{Q}_n^*$ and the empirical distribution of W. For example, the TMLE estimator of the ATE is

$$\psi_n^{TMLE} = \Psi(Q_n^*) = \frac{1}{n}\sum_{i=1}^{n}\left[\bar{Q}_n^*(1,W_i) - \bar{Q}_n^*(0,W_i)\right]$$

   where $Q_n^* = (\bar{Q}_n^*, Q_{W,n})$

In step 1 and 2, we highly recommend that a minimum loss-based learning, such as Super Learning, is used to obtain the estimators. It can avoid making misspecified paraemetric assumptions. In step 4, the parametric working submodel $\{\bar{Q}_n(\epsilon) : \epsilon\}$ and the loss function $\mathcal{L}(\bar{Q}_n)$ are chosen so that a linear combination of the components of the derivative of $\mathcal{L}(\bar{Q}_n(\epsilon))$ at $\epsilon = 0$ equals to the efficient influence curve $D^*(\bar{Q}_n, g_n)$:

$$\frac{d}{d\epsilon}\mathcal{L}(\bar{Q}_n(\epsilon)(O)|_{\epsilon=0} = D^*(\bar{Q}_n, g_n)(O)$$

As mentioned in the beginning of this section, the EIC is an element of the tangent space $T(P)$, and therefore the TMLE solves the EIC estimating equation $0 = \sum_{i=1}^{n} D^*(Q_n^*, g_n)(O_i)$. This is the key for establishing the asymptotic efficiency and double robustness of the TMLE under regularity conditions. In addition, the statistical inference such as standard errors and confidence intervals can be constructed based on the influence curve of the TMLE estimator.

1.2 Chapters summaries

The more detailed summaries of the three works which make up this dissertation are provided below:

Chapter 2: This chapter describes the robust semi-parametric approach to estimate the community-based causal effect of single time-point community-level stochastic interventions. Unlike the commonly used parametric regression models such as mixed models, that can easily violate the required statistical assumptions and result in invalid statistical inference, target maximum likelihood estimation allows more realistic data-generative models and provides double-robust, semi-parametric and efficient estimators. In addition, current applications in causal inference, especially in the context of hierarchical data structures, have focused on deterministic interventions in which each unit in population receives a fixed value. However, positivity violations can easily occur in many cases when certain subgroups in a sample have a (nearly) zero probability of receiving some interventions of interests. Target maximum likelihood estimators (TMLEs) for the causal effect of a community-level static exposure were previously proposed by Balzer *et al* [2]. In this chapter, we build on this work and present identifiability results and develop two semi-parametric efficient TMLEs for the estimation of the causal effect of the single time-point community-level stochastic intervention whose assignment mechanism can depend on measured and unmeasured environmental factors and its individual-level covariates. The first community-level TMLE is developed under a general hierarchical non-parametric structural equation model, which can incorporate pooled individual-level regressions for estimating the outcome mechanism. The second individual-level TMLE is developed under a restricted hierarchical model in which the additional assumption of "no covariate interference within communities" holds. Then, the target quantity of interest is defined as the mean of counterfactual community-level outcomes if all communities in the target population receive probabilistically assigned treatments based on a known specified mechanism, which is also called a "stochastic intervention". Here the community-level outcome is defined as the aggregate of the outcomes measured among individuals who are members from the same communities. The causal effect of interest may also be a contrast of the mean of the exposure-specific outcomes under two different stochastic interventions.

Chapter 3: This chapter describes the **tmleCommunity** package - an open-source software tool for implementing targeted minimum loss-based estimation (TMLE) of the effect of multivariate arbitrary community-level intervention(s) at a single time point on an individual-based outcome of interest, including the average causal effect. Implementations of the inverse-probability-of-treatment-weighting (IPTW) and the G-computation formula (GCOMP) are also available. The main function calculates the marginal treatment effect among independent community units (or independent and identically distributed individual units if no hierarchical structure exists) using TMLE, IPTW and GCOMP. Besides, it allows user-specified data-adaptive machine learning algorithms through **SuperLearner**, **sl3** and **h2oEnsemble** packages. The usage of the **tmleCommunity** package, along with a few examples, will be provided in this chapter.

Chapter 4: Approaches to high dimensional classification problems, particularly those using large administrative claims databases, are rapidly evolving. Dimensionality reduction and feature selection, along with data-driven machine learning algorithms, play crucial roles in those cases. In this chapter, we use SL to predict diagnoses of nonalcoholic steatohepatitis (NASH) in historical US administrative claims data. NASH is uniquely specified in the most recent version of claims coding (ICD-10-CM, US adoption October 2015) but was grouped with other conditions during the prior ICD-9-CM coding era. We first consider a SL library of 19 base learners that contains both nonparametric and parametric models, and evaluated its predictive performance via area under the curve (AUC) and computation time metrics. Results show that SL outperformed any individual base learner included in the library and additional claim codes in the claim database could be used to supplement researcher-specified covariates to improve prediction of NASH diagnoses. In terms of computational complexity, we also consider a SL library of 5 relatively fast algorithms, which showes comparable AUC results with a large gain in computational time. Overall, constructing a Super Learner with a rich library of diverse algorithms coupled with the comprehensive screening algorithm is the most effective and robust prediction strategy with respect to AUC performance, and may be promising for predictive modeling in high-dimensional administrative claims databases.

Chapter 2

Targeted maximum likelihood estimation of community-based causal effect of single time-point community-level stochastic interventions

2.1 Introduction

Motivation

The literature in fields such as epidemiology, econometrics and social science on the causal impact of community-level intervention, is rapidly evolving, both in observational studies and randomized trials. In observation settings, there is a rich literature on assessment of causal effects of families, schools and neighborhoods on child and adolescent development [7, 62]. For instance, the problem addressed by [5] is to estimate the impact of community violence exposure on anxiety among children of African American mothers with depression. Similarly, randomized community trials have increased in recent years. As pointed out by [53] and [69], scientifically speaking, community randomized controlled trials (CRCT) would be a superior strategy estimate the effects of community-level exposures due to self-selection and other difficulties. One example is the MTO study, which estimates the lower-poverty neighborhood effects on crime for female and male youth [37]. Another CRCT example is the ongoing SEARCH study, which estimates the community level interventions for the elimination of HIV in rural communities in East Africa [73]. Despite recent statistical advances, many of the current applications still rely on estimation techniques such as random effect models (or mixed models) [45] and generalized estimating equations (GEE) approach [50, 21]. However, those methods define the causal effect of interest as a coefficient in a most likely misspecified regression model, often resulting in bias and invalid statistical inference in observational settings, and loss of efficiency in randomized community trials. By

contrast, the targeted maximum likelihood estimators (TMLE) is constructed based on the efficient influence curve D^*, and therefore inherits its double robustness and local efficiency properties. Instead of using D^* directly to construct an efficient estimating equation, TMLE is obtained by constructing a locally least favorable submodel that its score (derivative of the log-likelihood) spans D^* [4, 44].

Deterministic interventions, in which each unit's treatment is set to a fixed value or a value defined by a deterministic function of the covariates, are the main strategy implemented in the current literature for the estimation of causal effects from observational data. One causal assumption needed for parameter identifiability is the positivity assumption. For example, the strong positivity assumption requires that all individuals in the population have a nonzero probability of receiving all levels of the treatment. As argued by [57], this strong assumption could be quite unrealistic in many cases. For example, patients with certain characteristics may never receive a particular treatment. On the other hand, a stochastic intervention is one in which each subject receives a probabilistically assigned treatment based on a known specified mechanism. Because the form of the positivity assumption needed for identifiability is model and parameter-specific, stochastic intervention causal parameters are natural candidates if requiring a weaker version of positivity compared to other causal parameters for continuous exposures. Furthermore, a policy intervention will lead to stochastic rather than deterministic interventions if the exposure of interest can only be manipulated indirectly, such as when studying the benefits of vigorous physical activity on a health outcome of interest in the elderly [3]. Because it is unrealistic to enforce every elderly person to have a certain level of physical activity depending on a deterministic rule. To deal with the previous considerations, stochastic interventions could be a more flexible strategy of defining a question of interest and being better supported by the data than deterministic interventions. Thus, using stochastic intervention causal parameters is a good way of estimating causal effects of realistic policies, which could also be naturally used to define and estimate causal effects of continuous treatments or categorical multilevel treatments [33].

Organization of this chapter

The rest of this chapter is organized as follows. In this chapter, we apply the roadmap for targeted learning of a causal effect [56]. In Section 2.2 we specify the causal model through a non-parametric structural equation model (NPSEM), allowing us to define the community-level causal effect of interest for arbitrary community-level stochastic interventions as a parameter of the NPSEM, define the corresponding observed data structure, and establish the identifiability of the causal parameter from the observed data generating distribution. We allow for general types of single time-point interventions, including static, dynamic and stochastic interventions. In other words, there is no further restrictions on the intervention distributions, which could be either degenerate (for deterministic interventions) or non-degenerate (for stochastic interventions). Next, Section 2.3 and 2.4 introduce two different TMLEs of the counterfactual mean outcome across communities under a community level intervention that are based on community-level and individual-level analysis, respec-

tively. Both TMLEs can make use of individual level data in the hierarchical setting. The first community-level TMLE is developed under a general hierarchical causal model and can incorporate some working models about the dependence structure in a community. In other words, the Super Learner library of candidate estimators for the outcome regression can be expanded to include pooled individual-level regressions based on the working model. The first TMLE also includes the case of observing one individual per community unit as a special case. The second individual-level TMLE is developed under a more restricted hierarchical model in which the additional assumption of dependence holds.

2.2 Definition of statistical estimation problem

General hierarchical casual model

Throughout this chapter, we use the bold font capital letters to denote random vectors and matrices. In studies of community-level interventions, we begin with a simple scenario that involves randomly selecting J independent communities from some target population of communities, sampling individuals from those chosen communities, and measuring baseline covariates and outcomes on each sampled individual at a single time point. Also, the number of chosen individuals within each community is not fixed, so communities are indexed with $j = 1, 2, ..., J$ and individual within the j^{th} community are indexed with $i = 1, ..., N_j$.

After selection of the communities and individuals, pre-intervention covariates and a post-intervention outcome are measured on each sampled unit. Because only some of the pre-intervention covariates have clear individual-level counterpart, the pre-intervention covariates separates into two sets: firstly, let denote $W_{j,i}$ the $(1 \times p)$ vector of p such individual-level baseline characteristics, and so $\boldsymbol{W}_j = (W_{j,i} : i = 1, ..., N_j)$ is an $(N_j \times p)$ matrix of individual-level characteristics; secondly let E_j represent the vector of community-level (environmental) baseline characteristics that have no individual-level counterpart and are shared by all community members, including the number of individuals selected within the community (i.e., $N_j \in E_j$). Last, A_j is the exposure level assigned or naturally occurred in community j and $\boldsymbol{Y}_j = (Y_{j,i} : i = 1, ..., N_j)$ is the vector of individual outcomes of interest.

In order to translate the scientific question of interest into a formal causal quantity, we first specify a NPSEM with endogenous variables $X = (E, \boldsymbol{W}, A, \boldsymbol{Y})$ that encodes our knowledge about the causal relationships among those variables and could be applied in both observational setting and randomized trials [54, 55].

$$\begin{aligned} U &= (U_E, U_{\boldsymbol{W}}, U_A, U_{\boldsymbol{Y}}) \sim P_U \\ E &= f_E(U_E) \\ \boldsymbol{W} &= f_{\boldsymbol{W}}(E, U_{\boldsymbol{W}}) \\ A &= f_A(E, \boldsymbol{W}, U_A) \\ \boldsymbol{Y} &= f_{\boldsymbol{Y}}(E, \boldsymbol{W}, A, U_{\boldsymbol{Y}}). \end{aligned} \tag{2.1}$$

where the U components are exogenous error terms, which are unmeasured and random with an unknown distribution P_U. Given an input U, the function $F = \{f_E, f_{\boldsymbol{W}}, f_A, f_{\boldsymbol{Y}}\}$ deterministically assigns a value to each of the endogenous variables. For example, model (2.1) assumes that each individual's outcome Y is affected by its baseline community-level and individual-level covariates $(E, \boldsymbol{W})$ together with its community-level intervention(s) and unobserved factors $(A, U_{\boldsymbol{Y}})$. First, while we might have specification of f_A, the structural equations $f_E, f_{\boldsymbol{W}}, f_{\boldsymbol{Y}}$ do not necessarily restrict the functional form of the causal relationships, which could be nonparametric (entirely unspecific), semiparametric or parametric that incorporates domain knowledge. Second, as summarized by [2], structural causal model (2.1) covers a wide range of practical scenarios as it allows for the following types of between-individual dependencies within a community: (i) the individual-level covariates (and outcomes) among members of a community may be correlated as a consequence of shared measured and unmeasured community-level covariates (E, U_E), and of possible correlations between unmeasured individual-level error terms $(U_{\boldsymbol{W}}, U_{\boldsymbol{Y}})$, and (ii) an individual i's outcome $Y_{j,i}$ may influence another l's outcome $Y_{j,l}$ within community j, and (iii) an individual's baseline covariates $W_{j,l}$ may influence another outcome $Y_{j,i}$. Actually, we can make an assumption about the third type of between-individual dependence, and so the structural equation $f_{\boldsymbol{Y}}$ will be specified under this assumption. More details will be discussed in section (2.4). Third, an important ingredient of this model is to assume that distinct communities are causally independent and identically distributed. The NPSEM defines a collection of distributions (U, X), representing the full data model, where each distribution is determined by F and P_U (i.e., $P_{U,X,0}$ is the true probability distribution of (U, X)). We denote the model for $P_{U,X,0}$ with $\mathcal{M}^{\mathcal{F}}$.

Counterfactuals and stochastic interventions

$\mathcal{M}^{\mathcal{F}}$ allows us to define counterfactual random variables as functions of (U, X), corresponding with arbitrary interventions. For example, with a static intervention on A, counterfactual $\boldsymbol{Y}_a$ can be defined as $f_{\boldsymbol{Y}}(E, \boldsymbol{W}, a, U_{\boldsymbol{Y}})$, replacing the structural equation f_A with the constant a [80]. Thus, $\boldsymbol{Y}_{j,a} = (Y_{j,i,a} : i = 1, ..., N_j)$ represents the vector of individual-level outcomes that would have been obtained in community j if all individuals in that community had actually been treated according to the exposure level a. More generally, we can replace data generating functions for A that correspond with degenerate choices of distributions for drawing A, given $U = u$ and $(E, \boldsymbol{W})$, by user-specified conditional distributions of A^*. Such non-degenerate choices of intervention distributions are often referred to as stochastic interventions.

First, let g^* denote our selection of a stochastic intervention identified by a set of multivariate conditional distributions of A^*, given the baseline covariates $(E, \boldsymbol{W})$. For convenience, we represent the stochastic intervention with a structural equation, where $A^* = f_{A^*}(E, \boldsymbol{Y}, U_{A^*})$ in terms of random errors U_{A^*}, and so define $\boldsymbol{Y}_{g^*} = f_{\boldsymbol{Y}}(E, \boldsymbol{W}, A^*, U_{\boldsymbol{Y}})$. Then $\boldsymbol{Y}_{j,g^*} = (Y_{j,i,g^*} : i = 1, ..., N_j)$ denotes the corresponding vector of individual-level counterfactual outcome for community j. Second, let Y^c denote a scalar representing a

community-level outcome that is defined as a aggregate of the outcomes measured among individuals who are members within a community, and so $Y^c_{g^*}$ is the corresponding community-level counterfactual of interest. One typical choice of Y^c_{j,g^*} is the weighted average response among the N_j individuals sampled from community j, i.e. $Y^c_{j,g^*} \equiv \sum_{i=1}^{N_j} \alpha_{j,i} Y_{j,i,g^*}$, for some user-specified set of weights α for which $\sum_{i=1}^{N_j} \alpha_{j,i} = 1$. If the underlying community size N_j differs, a natural choice of $\alpha_{j,i}$ is the reciprocal of the community size (i.e., $\alpha_{j,i} = 1/N_j$).

Target parameter on the NPSEM

We focus on community-level causal effects where all communities in the target population receive the intervention g^*, then our causal parameter of interest is given by

$$\Psi^F(P_{U,X,0}) = \mathbb{E}_{U,X}[Y^c_{g^*}] = \mathbb{E}_{U,X}\Big\{ \sum_{i=1}^{N} \alpha_i Y_{i,g^*} \Big\}$$

To simply expression, we use $\alpha_i = 1/N$ in the remainder of article. We also assume (without loss of generality) that the community-level outcome Y^c is bounded in $[0, 1]$. If instead $Y^c \in [a, b]$, the the original outcome will be automatically transformed into $Y^{c'} = \frac{Y^c - a}{b-a}$, and our target parameter is corresponding to $Y^{c'}$. Statistical inference such as the point estimate, limiting distribution and confidence interval for the latter target parameter can be immediately mapped into statistical inference for the original target parameter based on Y^c, by simply multiplying by $(b - a)$ [23].

One type of stochastic interventions could be a shifted version of the current treatment mechanism g_0, i.e., $P_{g^*}(A = a|E, \boldsymbol{W}) = g_0(a - \nu(E, \boldsymbol{W})|E, \boldsymbol{W})$ given a known shift function $\nu(E, \boldsymbol{W})$. A simple example is a constant shift of $\nu(E, \boldsymbol{W}) = 0.5$. Another more complex type could be stochastic dynamic interventions, in which the interventions can be viewed as random assignments among dynamic rules. A simple example corresponding to the previous shift function is $P_{g^*}(A = a|E, \boldsymbol{W}) = g_0(\max\{a - 0.5, \min(a)\}|E, \boldsymbol{W})$, indicating that shifted exposure A^* is always bounded by the minimum of the observed exposure A.

One might also be interested in the contrasts of the expectation of community-level outcome across the target population of communities under different interventions, i.e.,

$$\Psi^F(P_{U,X,0}) = \mathbb{E}_{U,X}(Y^c_{g_1^*}) - \mathbb{E}_{U,X}(Y^c_{g_2^*}) = \mathbb{E}_{U,X}\Big\{ \frac{1}{N} \sum_{i=1}^{N} Y_{i,g_1^*} \Big\} - \mathbb{E}_{U,X}\Big\{ \frac{1}{N} \sum_{i=1}^{N} Y_{i,g_2^*} \Big\}$$

where g_1^* and g_2^* are two different stochastic interventions.

Finally, additive treatment effect is a special case of average causal effect with two static interventions $g_1^*(1|e, \boldsymbol{w}) = 1$ and $g_2^*(0|e, \boldsymbol{w}) = 1$ for any $e \in E, \boldsymbol{W} \in \boldsymbol{W}$, i.e.,

$$\mathbb{E}_{U,X}(Y^c(1)) - \mathbb{E}_{U,X}(Y^c(0)) = \mathbb{E}_{U,X}[Y^c_{g_1^*(1|e,\boldsymbol{w})=1}] - \mathbb{E}_{U,X}[Y^c_{g_2^*(0|e,\boldsymbol{W})=1}]$$

Link to observed data

Consider the study design presented above where for a randomly selected community, the observed data consist of the measured pre-intervention covariates, the intervention assignment, the vector of individual-level outcomes. Formally, one observation on community j, is coded as

$$O_{j,i} = (E_j, W_{j,i}, A_j, Y_{j,i})$$

which follows the typical time ordering for the variables measured on the i^{th} individuals within the j^{th} community.

Assume the observed data consists of J independent and identically distributed copies of $\mathbf{O}_j = (E_j, \boldsymbol{W}_j, A_j, \boldsymbol{Y}_j) \sim P_0$, where P_0 is an unknown underlying probability distribution in a model space $\mathcal{M}^I$. Here $\mathcal{M}^I = \{P(P_{U,X}) : P_{U,X} \in \mathcal{M}^F\}$ denotes the statistical model that is the set of possible distributions for the observed data O and only involves modeling g_0 (i.e., specification of f_A). The true observed data distribution is thus $P_0 = P(P_{U,X,0})$.

Identifiability

By defining the causal quantity of interest in terms of stochastic interventions (and target causal parameter as a parameter of the distribution $P_{U,X,0}$) on the NPSEM and providing an explicit link between this model and the observed data, we lay the groundwork for addressing the identifiability through P_0.

In order to express $\Psi^F(P_{U,X,0})$ as a parameter of the distribution P_0 of the observed data O, we now need to address the identifiability of $\mathbb{E}_{U,X}[Y^c_{g^*}]$ by adding two key assumptions on the NPSEM: the randomization assumption so called "no unmeasured confounders" assumption (Assumption 1) and the positivity assumption (Assumption 2). The identifiability assumptions will be briefly reviewed here, for details on identifiability, we refer to see [64, 76, 75, 33].

Assumption 1.

$$A \perp\!\!\!\perp \boldsymbol{Y}_a | E, \boldsymbol{W}$$

where the counterfactual random variable $\boldsymbol{Y}_a$ represents a collection of outcomes measured on the individuals from a community if its intervention is set to $A = a$ in causal model (2.1), replacing the structural equation f_A with the constant a.

Assumption 2.

$$\sup_{a \in \mathcal{A}} \frac{g^*(a|E, \boldsymbol{W})}{g(a|E, \boldsymbol{W})} < \infty, \text{ almost everywhere}$$

where $g^*(a|E, \boldsymbol{W}) = P_{g^*}(A = a|E, \boldsymbol{W})$, and assume $\inf_{a \in \mathcal{A}} g(a|E, \boldsymbol{W}) > \epsilon$ for some small ϵ.

Informally, **Assumption 1** restricts the allowed distribution for P_U to ensure that A and Y shares no common causes beyond any measured variables in $X = (E, \boldsymbol{W}, A, \boldsymbol{Y})$. For example, assumption 1 holds if U_A is independent of U_Y , given $E, \boldsymbol{W}$. Then, this randomization assumption implies $A^* \perp\!\!\!\perp \boldsymbol{Y}_a | E, \boldsymbol{W}$. In addition, as $P_{g^*}(A = a|E, \boldsymbol{W})$ is specified by

users in **Assumption 2**, a good selection of g^* can be used to estimate the causal parameter of interest, but yet does not generate unstable weighting that causes violations of the positivity assumption. Therefore, this posivitiy assumption is easier to achieve compared to other positivity assumptions that other causal parameters used for continuous interventions.

Under **Assumption 1** and **2**, jointly with the consistency assumption (i.e., $A = a$ implies $\boldsymbol{Y}_a = \boldsymbol{Y}$),

$$\begin{aligned} &P(\boldsymbol{Y}_{g^*} = \boldsymbol{y}|A^* = a, E = e, \boldsymbol{W} = \boldsymbol{w}) = P(\boldsymbol{Y}_a = \boldsymbol{y}|A^* = a, E = e, \boldsymbol{W} = \boldsymbol{w}) \\ &= P(\boldsymbol{Y}_a = \boldsymbol{y}|E = e, \boldsymbol{W} = \boldsymbol{w}) = P(\boldsymbol{Y} = \boldsymbol{y}|A = a, E = e, \boldsymbol{W} = \boldsymbol{w}) \end{aligned}$$

So our counterfactual distribution $P(\boldsymbol{Y}_{g^*} = \boldsymbol{y})$ can be written as:

$$\begin{aligned} P(\boldsymbol{Y}_{g^*} = \boldsymbol{y}) &= \int_{e,\boldsymbol{w}} \int_a P(\boldsymbol{Y}_{g^*} = \boldsymbol{y}|A^* = a, E = e, \boldsymbol{w} = \boldsymbol{w}) g^*(a|e, \boldsymbol{w}) d\mu(a) dP_{E,\boldsymbol{W}}(e, \boldsymbol{w}) \\ &\quad \text{by the law of iterated conditional expectation} \\ &= \int_{e,\boldsymbol{w}} \int_a P(\boldsymbol{Y}_a = \boldsymbol{y}|E = e, \boldsymbol{W} = \boldsymbol{w}) g^*(a|e, \boldsymbol{w}) d\mu_a(a) dP_{E,\boldsymbol{W}}(e, \boldsymbol{w}) \\ &\quad \text{by } \textbf{assumption 1} \text{ and } A^* \perp\!\!\!\perp \boldsymbol{Y}_a | E, \boldsymbol{W} \\ &= \int_{e,\boldsymbol{w}} \int_a P(\boldsymbol{Y} = \boldsymbol{y}|A = a, E = e, \boldsymbol{W} = \boldsymbol{w}) g^*(a|e, \boldsymbol{w}) d\mu_a(a) dP_{E,\boldsymbol{W}}(e, \boldsymbol{w}) \\ &\quad \text{by consistency assumption} \end{aligned}$$

with respect to some dominating measure $\mu_a(a)$.

Then, $\mathbb{E}_{U,X}[\boldsymbol{Y}_{g^*}]$ is identified by the G-computational formula [64]:

$$\begin{aligned} \mathbb{E}_{U,X}[\boldsymbol{Y}_{g^*}] &= \mathbb{E}_{E,\boldsymbol{W}}[\mathbb{E}_{g^*}[\boldsymbol{Y}|A^* = a, E, \boldsymbol{W}]] \\ &= \int_{e,\boldsymbol{w}} \int_a \mathbb{E}_{g^*}(\boldsymbol{Y}|a, e, \boldsymbol{w}) g^*(a|e, \boldsymbol{w}) d\mu_a(a) dP_{E,\boldsymbol{W}}(e, \boldsymbol{w}) \end{aligned}$$

This provides us with a general identifiability result for $\mathbb{E}_{U,X}[Y^c_{g^*}]$, the causal effect of the community-level stochastic intervention on any community-level outcome Y^c that is some real valued function of the individual-level outcome $\boldsymbol{Y}$:

$$\mathbb{E}_{U,X}[Y^c_{g^*}] = \mathbb{E}_{U,X}[\sum_{i=1}^{N} \alpha_i Y_{g^*,i}] = \sum_{i=1}^{N} \alpha_i \mathbb{E}_{E,\boldsymbol{W}}[\mathbb{E}_{g^*}[Y_i|A^*, E, \boldsymbol{W}]] \equiv \Psi^I(P_0) = \psi_0^I$$

The statistical parameter and model for observed data

If we only assume the randomization assumption in the previous section, then the statistical model $\mathcal{M}^I$ is nonparametric. Based on the result of identifiability, we note that $\Psi^I : \mathcal{M}^I \to \mathbb{R}$ represents a mapping from a probability distribution of $\mathbf{O}$ into a real number, and $\Psi^I(P_0)$ denotes the target estimand corresponding to the target causal quantity $\mathbb{E}_{U,X}[\boldsymbol{Y}_{g^*}]$.

Before defining the statistical parameter, we introduce some additional notation. First, we denote the marginal distribution of the baseline covariates $(E, \boldsymbol{W})$ by $Q_{E,\boldsymbol{W}}$, with a well-defined density $q_{E,\boldsymbol{W}}$, with respect to some dominating measure $\mu_y(y)$. There is no additional assumption of independence for $Q_{E,\boldsymbol{W}}$. Second, let G denote the observed exposure conditional distribution for A that has a conditional density $g(A|E, \boldsymbol{W})$. Third, we assume that all Y within a community are sampled from the distribution $Q_{\boldsymbol{Y}}$ with density given by $q_{\boldsymbol{Y}}(\boldsymbol{Y}|A, E, \boldsymbol{W})$, conditional on the exposure and the baseline covariates $A, E, \boldsymbol{W}$. Now we introduce the notation $P = P_{\tilde{Q},G}$ for $\tilde{Q} = (Q_{\boldsymbol{Y}}, Q_{E,\boldsymbol{W}})$, and the statistical model becomes $\mathcal{M}^I = \{P_{\tilde{Q},G} : \tilde{Q} \in \tilde{\mathcal{Q}}, G \in \mathcal{G}\}$, where $\tilde{\mathcal{Q}}$ and $\mathcal{G}$ denote the parameter space for $\tilde{Q}$ and G, respectively, and $\tilde{\mathcal{Q}}$ here is nonparametric.

Next, we define G^* as the user-supplied intervention with a new density g^*, which will replace the observed conditional distribution G. So G^* is a conditional distribution that describes how each intervened treatment is produced conditional on the baseline covariate $(E, \boldsymbol{W})$. Given $\tilde{Q}$ and G^*, we use $\mathbf{O}^* = (O^*_{j,i} = (E_j, W_{j,i}, A^*_j, Y^*_{j,i}) : i = 1, ..., N_j, j = 1, , , ., J)$ to denote a random variable generated under the post-intervention distribution $P_{\tilde{Q},G^*}$. Namely, $P_{\tilde{Q},G^*}$ is the G-computation formula for the post-intervention distribution of observed data $\mathbf{O}$ under stochastic intervention G^* [64], and the likelihood for $P_{\tilde{Q},G^*}$ can be factorized as:

$$p_{\tilde{Q},G^*}(\mathbf{O}^*) = [\prod_{j=1}^{J} q_{\boldsymbol{Y}}(\boldsymbol{Y}^*_j|A^*_j, \boldsymbol{W}_j, E_j)][\prod_{j=1}^{J} g^*(A^*_j|E_j, \boldsymbol{W}_j)]q_{E,\boldsymbol{W}}(E, \boldsymbol{W}) \tag{2.2}$$

Thus our target statistical quantity is now defined as $\psi^I_0 = \Psi^I(P_0) = \mathbb{E}_{\tilde{q}_0,g^*}[Y^c_{g^*}]$, where $\Psi^I(P_0)$ is the target estimand of the true distribution of the observed data $P_0 \in \mathcal{M}^I$ (i.e., a mapping from the statistical model $\mathcal{M}^I$ to $\mathbb{R}$). We then define $\bar{Q}(A_j, E_j, \boldsymbol{W}_j) = \int_y \boldsymbol{y} q_{\boldsymbol{Y}}(\boldsymbol{y}|A_j, \boldsymbol{W}_j, E_j) d\mu_y(y)$ as the conditional mean evaluated under common-in-j distribution $Q_{\boldsymbol{Y}}$, and so $\bar{Q}^c(A, E, \boldsymbol{W}) \equiv E(Y^c|A, E, \boldsymbol{W})$ as the conditional mean of the community-level outcome. Now we can refer to $Q_0 = (\bar{Q}^c_0, Q_{E,\boldsymbol{W},0})$ as the part of the observed data distribution that our target parameter is a function of (i.e., with a slight abuse of notation $\Psi^I(P_0) = \Psi^I(Q_0)$), the parameter ψ^I_0 can be written as:

$$\psi^I_0 = \int_{e\in\mathcal{E},\boldsymbol{w}\in\mathcal{W}} \int_{a\in\mathcal{A}} \bar{Q}^c_0(a, e, \boldsymbol{w}) g^*(a|e, \boldsymbol{w}) d\mu_a(a) q_{E,\boldsymbol{W},0}(e, \boldsymbol{w}) d\mu_{e,w}(e, \boldsymbol{w}) \tag{2.3}$$

with respect to some dominating measures $\mu_a(a)$ and $\mu_{e,w}(e, \boldsymbol{w})$, where $(\mathcal{A}, \mathcal{E}, \mathcal{W})$ is the common support of $(A, E, \boldsymbol{W})$.

Sometimes researchers might be interested in target quantities defined as the difference or ratio of two stochastic interventions. For example, one might define two target estimands $\mathbb{E}_{\tilde{q}_0,g^*_1}[Y^c_{g^*_1}]$ and $\mathbb{E}_{\tilde{q}_0,g^*_2}[Y^c_{g^*_2}]$ evaluated under two different interventions g^*_1 and g^*_2, then defining the target quantity as $\mathbb{E}_{\tilde{q}_0,g^*_2}[Y^c_{g^*_2}] - \mathbb{E}_{\tilde{q}_0,g^*_1}[Y^c_{g^*_1}]$. Actually a generalization of target quantities can be expressed as Euclidean-value functions of a collection $\{\mathbb{E}_{\tilde{q}_0,g^*}[Y^c_{g^*}] : g^* \in \mathscr{J}\}$, where $\mathscr{J}$ denotes a finite set of possible stochastic interventions.

2.3 Estimation and inference under the general hierarchical causal model

In the previous section, we have defined a statistical model $\mathcal{M}^I$ for the distribution of $\mathbf{O}$, and a statistical target parameter mapping Ψ^I) for which $\Psi^I(P_{Q_0,G^*})$ only depends on Q_0 through a relevant part $Q_0 = Q(P_0)$ of P_0. Now we want to estimate $\Psi^I(Q_0)$ via a target maximum likelihood estimator (TMLE) and construct an asymptotically valid confidence interval through the efficient influence curve (EIC). Furthermore, we present a novel method for the estimation of the outcome regression in which incorporates additional knowledge about the data generating mechanism that might be known by design.

As a two-stage procedure, TMLE needs to estimate both the outcome regressions $\bar{Q}_0$ and treatment mechanism g_0. Since TMLE solves the EIC estimating equation, its estimator inherits the double robustness property of this EIC and is guaranteed to be consistent (i.e., asymptotically unbiased) if either $\bar{Q}_0$ or g_0 is consistently estimated. For example, in a community randomized controlled trial g_0 is known to be 0.5 and can be consistently estimated, thus its TMLE will always be consistent. Besides, TMLE is efficient when both are consistently estimated. In other words, when g_0 is consistent, a choice of the initial estimator for $\bar{Q}_0$ that is better able to approximate the true value $\bar{Q}_0$ may improve the asymptotic efficiency along with finite sample bias and variance of the TMLE [79].

The efficient influence curve D^*

Before constructing a community-level TMLE of $\Psi^I(P_0)$, we must understand its efficient influence curve. The EIC, evaluated at the true distribution $P_0 \in \mathcal{M}$, is given by:

$$\begin{aligned} D^I(P_0)(\mathbf{O}) = &\frac{g^*}{g_0}(A|E,\boldsymbol{W})(Y^c - \bar{Q}_0^c(A,E,\boldsymbol{W})) \\ &+ \mathbb{E}_{g^*}[\bar{Q}_0^c(A,E,\boldsymbol{W})|E,\boldsymbol{W}] - \Psi^I(\mathbb{P}_{Q,g^*}) \end{aligned}$$

where

$$\begin{aligned} \mathbb{E}_{g^*}[\bar{Q}_0^c(A,E,\boldsymbol{W})|E,\boldsymbol{W}] &= \int_a \bar{Q}_0^{c*}(a,E,\boldsymbol{W})g^*(a|E,\boldsymbol{W})d\mu_a(a) \\ D_Y^I(P_0)(\mathbf{O}) &= \frac{g^*}{g_0}(A|E,\boldsymbol{W})(Y^c - \bar{Q}_0^c(A,E,\boldsymbol{W})) \\ D_{E,\boldsymbol{W}}^I(P_0)(\mathbf{O}) &= \mathbb{E}_{g^*}[\bar{Q}_0^c(A,E,\boldsymbol{W})|E,\boldsymbol{W}] - \Psi^I(\mathbb{P}_{Q,g^*}) \end{aligned}$$

Here $D_Y^I(P)$ and $D_{E,\boldsymbol{W}}^I(P)$ are defined as the projection of the EIC $D^*(P)$ onto the tangent space of $P_{Y|A,E,\boldsymbol{W}}$ at $P \in \mathcal{M}^I$ and $P_{E,\boldsymbol{W}}$ at $P \in \mathcal{M}^I$, given $P = P_{E,\boldsymbol{W}}P_{A|E,\boldsymbol{W}}P_{Y|A,E,\boldsymbol{W}}$, respectively. Note that the projection of the EIC onto the tangent space of $P_{A|E,\boldsymbol{W}}$ (i.e., the exposure mechanism) is zero.

The community-level TMLE

The community-level TMLE first obtains an initial estimate $\hat{\bar{Q}}^c(A, E, \boldsymbol{W})$ for the conditional mean of the community-level outcome $\bar{Q}_0^c(A, E, \boldsymbol{W})$, and also an estimate $\hat{g}(A|E, \boldsymbol{W})$ of the community-level density of the conditional treatment distribution $g(A|E, \boldsymbol{W})$. The second targeting step is to create a targeted estimator $\hat{\bar{Q}}^{c*}$ of $\bar{Q}_0^c$ by updating the initial fit $\hat{\bar{Q}}^c(A, E, \boldsymbol{W})$ through a parametric fluctuation that exploits the information in the estimated density for the conditional treatment distribution $\hat{g}(A|E, \boldsymbol{W})$. The plug-in community-level TMLE is then computed by the updated estimate $\hat{\bar{Q}}^{c*}(A, E, \boldsymbol{W})$ and the empirical distribution of $(E, \boldsymbol{W})$. In this subsection, we describe the community-level TMLE algorithm for estimating the community-based effect under community-level stochastic interventions. For further discussion, please see [33, 68, 2].

Estimation of exposure mechanisms g_0 and g_0^*

A data-adaptive estimator of a conditional density that can be used to estimate the exposure mechanism is proposed by Dáaz and van der Laan [34]. Here, we build on this work and present how to use the histogram-like estimator to estimate the community-level multivariate exposure mechanism $g_0(A|E, \boldsymbol{W})$. First let's define $g_0(a|E, \boldsymbol{W}) \equiv P_0(A = a|E, \boldsymbol{W})$, where the exposures and baseline covariates $(A, E, \boldsymbol{W}) = ((A_j, E_j, \boldsymbol{W}_j) : j = 1, \ldots, J)$ denote the random variables drawn jointly from the distribution $S_0(A, E, \boldsymbol{W})$ with the density $s_0(a, e, \boldsymbol{w}) \equiv g_0(a|e, \boldsymbol{w})q_{E,\boldsymbol{W},0}(e, \boldsymbol{w})$. Here $q_{E,\boldsymbol{W},0}(e, \boldsymbol{w})$ denotes the marginal density of the baseline covariates $(E, \boldsymbol{W})$, and communities are indexed with $j = 1, \ldots, J$. Then, let's denote $g_0^*(a^*|E, \boldsymbol{W}) \equiv P_{g_0^*}(A = a|E, \boldsymbol{W})$. The fitting algorithm for the non-parametric estimator $g_0^*(A^*|E, \boldsymbol{W})$ is equivalent, except that now the exposures and baseline covariates $(A^*, E, \boldsymbol{W}) = ((A_j^*, E_j, \boldsymbol{W}_j) : j = 1, .., J)$ are randomly drawn from $S_0^*(A, E, \boldsymbol{W})$ with the density $s_0^*(a, e, \boldsymbol{w})$ defined as $g_0^*(a|e, \boldsymbol{w})q_{E,\boldsymbol{W},0}(e, \boldsymbol{w})$, where A^* is determined by the user-supplied (stochastic) intervention.

Note that A can be multivariate (i.e., $A = (A(m) : m = 1, \ldots, M)$) where M represents the number of treatment variables, and any of its components $A(m)$ can be either binary, categorical or continuous. The joint probability model for $P(A|E, \boldsymbol{W}) \equiv P(A(1), \ldots, A(M)|E, \boldsymbol{W})$ can be factorized as a sequence:

$$P(A(1)|E, \boldsymbol{W}) \times P(A(2)|A(1), E, \boldsymbol{W}) \times \ldots \times P(A(M)|A(1), \ldots, A(M-1), E, \boldsymbol{W})$$

where each of these conditional probability models $P(A(m)|A(1), \ldots, A(m-1), E, \boldsymbol{W})$ is fitted separately, depending on the type of the m-specific outcome variable $A(m)$. For binary $A(m)$, the conditional probability $P(A(m)|A(1), \ldots, A(m-1), E, \boldsymbol{W})$ will be esimtated by a user-specific library of candidate algorithms, including both parametric estimators and data-adaptive estimators. For continuous (or categorical) $A(m)$, consider a sequence of values $\delta_1, \delta_2, \ldots, \delta_{K+1}$ that span the range of $A(m)$ and define K bins and the corresponding K bin indicators, in which case each bin indicator $B_k \equiv [\delta_k, \delta_{k+1})$ is used as an binary

outcome in a seperate user-specific library of candidate algorithms, with predictors given by $(A(1), \ldots, A(m-1), E, \boldsymbol{W})$. That is how the joint probability $P(A|E, \boldsymbol{W})$ is factorized into such an entire tree of binary regression models.

For simplicity (and without loss of generality), we now suppose A is univariate (i.e., $M = 1$) and continuous and a general template of an fitting algorithm for $P(A|E, \boldsymbol{W})$ is summarized below:

1. Initialization. Consider the usual setting in which we observe J independently and identically distributed copies $\mathbf{o}_j = (e_j, \boldsymbol{w}_j, a_j, \boldsymbol{y}_j : j = 1, \ldots, J)$ of the random variable $\mathbf{O} = (E, \boldsymbol{W}, A, \boldsymbol{Y})$, where the observed exposure $(a_j : j = 1, \ldots, J)$ are continuous.

2. Estimation of $P(A = a|E = e, \boldsymbol{W} = \boldsymbol{w})$.

   a) As described above, consider a sequence of $K + 1$ values that span the support of A values into K bin intervals $\Delta = (\delta_1, ..., \delta_K, \delta_{K+1})$ for a continuous variable A. Then any observed data point a_i belongs to one of the K intervals, in other words, for each possible value $a \in A$ (even if this a is not in the observed $(a_j : j = 1, \ldots, J)$, there always exists a $k \in 1, ..., K$ such that $a \in [\delta_k, \delta_{k+1})$), and the length (bandwidth) of the interval can be defined as $b_{wk} = \delta_{k+1} - \delta_k$.

   b) Then let the mapping $S(a) \in \{1, 2, \ldots, K\}$ denote a unique index of the indicator in λ that a falls in, where $S(a) = k$ if $a \in [\delta_k, \delta_{k+1})$, namely $\delta_{S(a)} \leq a < \delta_{S(a)+1}$. Moreover, we use b_k to denote a binary indicator of whether the observed a belongs to bin k (i.e., $b_k \equiv I(S(a) = k)$ for all $k \leq S(a)$).

      - This is similar to methods for censored longitudinal data, which treats exposures as censored or missing once the indicator b_k jumps from 0 to 1.
      - Since a is a realization of the random variable A for one community, the corresponding random binary indicator of whether A belongs to bin k can be denoted as:

$$B_k = \begin{cases} I(S(A) = k), & \forall k \leq S(A) \\ NA, & \forall k > S(A) \end{cases}$$

   c) Then for each $k = 1, \ldots, K$, a binary nonparametric regression is used to estimate the conditional probability $P(B_k = 1|B_{k-1} = 0, E, \boldsymbol{W})$, which corresponds to the probability of B_k jumping from 0 to 1, given $B_{k-1} = 0$ and the baseline covariates $(E, \boldsymbol{W})$. Here for each k, the corresponding nonparametric regression model is fitted only among observations that are uncensored (i.e., still at risk of getting $B_k = 1$ with $B_{k-1} = 0$). Note the above conditional probability

$$P(B_k = 1|B_{k-1} = 0, E, \boldsymbol{W}) \equiv P(A \in [\delta_k, \delta_{k+1})|A \geq \delta_k, E, \boldsymbol{W})$$

   which is the probability of A belongs to the interval $[\delta_k, \delta_{k+1})$, conditional on A does not belong to any intervals before $[\delta_k, \delta_{k+1})$, and $(E, \boldsymbol{W})$.

d) Then the discrete conditional hazard function for each k is defined as a normalization of the conditional probability using the corresponding interval bandwidth $bw_k (\equiv \delta_{k+1} - \delta_k)$:

$$\lambda_k(A, E, \boldsymbol{W}) = \frac{P(B_k = 1 | B_{k-1} = 0, E, \boldsymbol{W})}{bw_k} = \frac{P(A \in [\delta_k, \delta_{k+1}) | A \geq \delta_k, E, \boldsymbol{W})}{bw_k}$$

e) Finally, for any given observation $(a, e, \boldsymbol{w})$, we first find out the interval index k to which a belongs (i.e., $k = S(a) \in 1, \ldots, K$). Then the discretized conditional density of $P(A = a | E = e, \boldsymbol{W} = \boldsymbol{w})$ can be factorized by:

$$\lambda_k(A, E, \boldsymbol{W}) \times \Big\{ \prod_{t=1}^{k-1} (1 - \lambda_t(A, E, \boldsymbol{W})) \Big\}$$

which corresponds to the conditional probability of a belongs to the interval $[\delta_k, \delta_{k+1})$ and does not belong to any intervals before, given $(E, \boldsymbol{W})$.

3. The conditional density estimators of $g(A|E, \boldsymbol{W})$ is now proportional to:

$$\prod_{j=1}^{J} P(A_j \in [\delta_k, \delta_{k+1}) | E, \boldsymbol{W})$$

$$= \prod_{j=1}^{J} \Big[P(A_j \in [\delta_k, \delta_{k+1}) | A_j \geq \delta_k, E_j, \boldsymbol{W}_j) \times \prod_{t=1}^{k-1} (1 - P(A_j \in [\delta_t, \delta_{t+1}) | A_j \geq \delta_t, E_j, \boldsymbol{W}_j) \Big]$$

where $P(A \in [\delta_k, \delta_{k+1}) | A \geq \delta_k, E, \boldsymbol{W})$ can be estimated by either parametric or data-adaptive algorithms, or the combination of them (i.e., Super Learner). For example, using a main-term only logistic regression:

$$logit\{P(A \in [\delta_k, \delta_{k+1}) | A \geq \delta_k, E, \boldsymbol{W})\}$$
$$= \sum_{t=1}^{k} \alpha_t I(A \in [\delta_{t-1}, \infty)) + \sum_{s=1}^{S} \beta_s E_s + \sum_{l=1}^{p} \gamma_l \boldsymbol{W}_l$$

where we assume that the dimension of E is S and the dimension of $\boldsymbol{W}$ is p, and $I(A \in [\delta_{t-1}, \infty))$ indicates if A falls within the interval $[\delta_{t-1}, \infty)$. Alternatively, we can use Super Learner to build a convex combination of the candidate algorithms in the SL library to minimize the cross-validated risk, given a user-specified loss function.

Note that we need a clever way to determine the bin (interval) cutoffs for a continuous exposure. As proposed by Denby and Mallows [11], we can use a histogram-based method that is a compromise between the equal-bin-width histogram and equal-area histogram methods, and the corresponding parameters can be selected by cross validation. For detailed on constructing a histogram-like cross-validated density estimator, we refer to [34].

Loss function and initial (non-targeted) estimator of $\bar{Q}_0^c$

As an initial estimator of $\bar{Q}_0^c$, we can simply regress the community-level outcome Y^c onto the exposure and baseline covariates $(A, E, \boldsymbol{W})$. The estimation of $\hat{\bar{Q}}^c$ could be processed by either the usual parametric MLE or loss-based machine learning algorithms based on cross validation, such as loss-based super learning. Given that Y_j^c is bounded continuous or discrete for some known range $Y_j^c \in [a, b], \forall j = 1, \ldots, J$, the estimation of $\hat{\bar{Q}}^c$ can be based on the following negative Bernoulli log-likelihood loss function:

$$-\mathcal{L}^c(\bar{Q}^c)(O) = \sum_{j=1}^{J} \Big[Y_j^c \log[\bar{Q}^c(A_j, E_j, \boldsymbol{W}_j)] + (1 - Y_j^c) \log[1 - \bar{Q}^c(A_j, E_j, \boldsymbol{W}_j)] \Big],$$

or the squared error loss

$$\mathcal{L}^c(\bar{Q}^c)(O) = \sum_{j=1}^{J} [Y_j^c - \bar{Q}^c(A_j, E_j, \boldsymbol{W}_j)]^2$$

For example, for continuous Y_j^c, the fitted parameter in a least squares regression can be defined as:

$$\hat{\beta}_{LS}^c = \arg\min_{\beta} \sum_{j=1}^{J} [Y_j^c - \bar{Q}_\beta^c(A_j, E_j, \boldsymbol{W}_j)]^2$$

Loss function and the least favorable fluctuation submodel that spans the efficient influence curve

Recall that the targeting step in the TMLE algorithm needs to define a fluctuation parametric submodel for $\hat{\bar{Q}}^c$ and a corresponding user-specified loss function. Given the initial estimator of outcome mechanism $\hat{\bar{Q}}^c(A_j, E_j, \boldsymbol{W}_j)$, and the initial estimator of treatment mechanisms $\hat{g}(A_j = a|E_j, \boldsymbol{W}_j)$ and $\hat{g}^*(A_j = a|E_j, \boldsymbol{W}_j)$ for each community $j = 1, ..., J$, the TMLE algorithm updates the initial estimator $\hat{\bar{Q}}^c$ into $\hat{\bar{Q}}^{c*}$ by

1. define a submodel $\hat{\bar{Q}}^c(\epsilon)$ with parameter ϵ as

$$logit(\hat{\bar{Q}}^c(\epsilon)(a, E_j, \boldsymbol{W}_j)) = logit(\hat{\bar{Q}}^c(a, E_j, \boldsymbol{W}_j) + \epsilon \hat{H}_j(a, E_j, \boldsymbol{W}_j)), \forall j = 1, ..., J$$

   where $logit(x) = \log(\frac{x}{1-x})$, and $\hat{H}_j(a, E_j, \boldsymbol{W}_j) = \frac{\hat{g}^*(A_j=a|E_j, \boldsymbol{W}_j)}{\hat{g}(A_j=a|E_j, \boldsymbol{W}_j)}$ displays the community-level clever covariate, and the fluctuation parameter ϵ is obtained by a logistic regression of Y^c on $\hat{H}$ with offset $\text{logit}(\hat{\bar{Q}}^c)$. Note that $\hat{\bar{Q}}^c(\epsilon = 0) = \hat{\bar{Q}}^c$ at zero fluctuation.

2. define a community-level loss function such as binary log-likelihood loss function:

$$-\mathcal{L}(\hat{\bar{Q}}^c(\epsilon))(O) = Y^c \log[\hat{\bar{Q}}^c(\epsilon)(A, E, \boldsymbol{W})] + (1 - Y^c) \log[(1 - \hat{\bar{Q}}^c(\epsilon)(A, E, \boldsymbol{W}))],$$

and the derivative of the loss function at zero fluctuation has:

$$\begin{aligned}\frac{d}{d\epsilon}\mathcal{L}(\hat{\bar{Q}}^c(\epsilon)(O))|_{\epsilon=0} &= \hat{H}(A,E,\boldsymbol{W})(Y^c - \hat{\bar{Q}}^c(A,E,\boldsymbol{W})) \\ &= \frac{\hat{g}^*}{\hat{g}}(A|E,\boldsymbol{W})(Y^c - \hat{\bar{Q}}^c(A,E,\boldsymbol{W})) \\ &= D_Y^I(\hat{Q},\hat{g})(O)\end{aligned}$$

where $D_Y^I(\hat{Q},\hat{g})$ is a component of the EIC D^I of Ψ^I at $(\hat{Q},\hat{g})$.

3. the updated fit $\hat{\bar{Q}}^{c*}$ is defined as $\hat{\bar{Q}}^c(\hat{\epsilon}) = logit^{-1}(logit(\hat{\bar{Q}}^c) + \hat{\epsilon}\hat{H})$, where $\hat{\epsilon}$ minimizes the empirical loss function above:

$$\hat{\epsilon} = \arg\min_{\epsilon} \sum_{j=1}^{J} \mathcal{L}(\hat{\bar{Q}}^c(\epsilon))(O_j)$$

Another way to achieve the targeting step is to use weighted regression intercept-based TMLE, where ϵ is obtained by a intercept-only weighted logistic regression of Y^c with offset $logit(\hat{\bar{Q}}^c)(A_j,E_j,\boldsymbol{W}_j)$, predicted weights $\hat{H}_j(A_j,E_j,\boldsymbol{W}_j)$ and no covariates. In summary, this alternative targeting can be implemented by

1. define a submodel $\hat{\bar{Q}}^c(\epsilon)$ with parameter ϵ as

$$logit(\hat{\bar{Q}}^c(\epsilon)(a,E_j,\boldsymbol{W}_j)) = logit(\hat{\bar{Q}}^c(a,E_j,\boldsymbol{W}_j) + \epsilon, \forall j = 1,...,J$$

2. define a weighted (binary log-likelihood) loss function:

$$-\mathcal{L}(\hat{\bar{Q}}^c(\epsilon))(O) = \Big\{ \log[\hat{\bar{Q}}^c(\epsilon)(A,E,\boldsymbol{W})^{Y^c}(1-\hat{\bar{Q}}^c(\epsilon)(A,E,\boldsymbol{W}))^{1-Y^c}] \Big\} \hat{H}_j(a,E_j,\boldsymbol{W}_j)$$

3. the updated fit $\hat{\bar{Q}}^{c*} = \hat{\bar{Q}}^c(\hat{\epsilon}) = logit^{-1}(logit(\hat{\bar{Q}}^c) + \hat{\epsilon})$, where $\hat{\epsilon}$ minimizes the above loss function.

It is worth mentioning that both of the fluctuation methods solves the same empirical EIC estimating equation and thus generate TMLEs with equivalent asymptotic efficiency. However, the latter one, the intercept-based weighted TMLE, is less sensitive to practical positivity violations in finite samples, while obtaining the similar bias reduction in the target parameter [67].

A similar targeting algorithm can be applied to the marginal distribution of $(E,\boldsymbol{W})$. We select a loss function of $Q_{E,\boldsymbol{W}}$ and a parametric working submodel $\hat{Q}_{E,\boldsymbol{W}}(\epsilon)$, so that the derivative of the loss function of $\hat{Q}_{E,\boldsymbol{W}}(\epsilon)$ at zero fluctuation has:

$$\begin{aligned}\frac{d}{d\epsilon}\mathcal{L}(\hat{Q}_{E,\boldsymbol{W}}(\epsilon))|_{\epsilon=0} &= \mathbb{E}_{g^*}[\hat{\bar{Q}}^c(A,E,\boldsymbol{W})|E,\boldsymbol{W}] - \Psi^I(\mathbb{P}_{\hat{Q},\hat{g}^*}) \\ &= D^I_{E,\boldsymbol{W}}(\hat{Q},\hat{g})(O)\end{aligned}$$

However, this targeting step doesn't generate any update because the empirical distribution $Q_{E,\boldsymbol{W}}$ is non-parametric MLE estimator and has no contribution to the bias for our target parameter [80].

The community-level TMLE estimator

Thus our targeted substitution estimator is computed as the weighted mean of the targeted predictions across the J communities, given the updated estimate $\hat{\bar{Q}}^{c*}$, the estimate of the user-specified stochastic intervention, and the empirical distribution of $(E, \boldsymbol{W})$. One natural choice is the empirical mean defined as follows:

$$\hat{\Psi}^I(P_{\hat{Q}^*,\hat{g}^*}) = \frac{1}{J}\sum_{j=1}^{J}\int_{e_j,\boldsymbol{w}_j}\int_a \hat{\bar{Q}}^{c*}(a, e_j, \boldsymbol{w}_j)\hat{g}^*(a|e_j, \boldsymbol{w}_j)d\mu_a(a)q_{E,\boldsymbol{W}}(e_j, \boldsymbol{w}_j)d\mu_{e,w}(e_j, \boldsymbol{w}_j)$$

Statistical inference for the community-level TMLE

By construction, the community-level TMLE estimator $\hat{Q}^*$ will solve the EIC equation:

$$0 = \sum_{j=1}^{J} D^I(\hat{Q}^*, \hat{g}^*)(O_j)$$

which results in its doubly robust locally efficient property.

In practice, community-level TMLE variance is asymptotically estimated as $\mathrm{Var}(\hat{\Psi}^I(\hat{Q}^*)) \approx \frac{(\hat{\sigma}_J^I)^2}{J}$, where $(\hat{\sigma}_J^I)^2$ is the sample variance of the estimated influence curve obtained by

$$(\hat{\sigma}_J^I)^2 = \frac{1}{J}\sum_{j=1}^{J}\{D^I(\hat{Q}^*, \hat{g})(O_j)\}^2$$

where $D^I(\hat{Q}^*, \hat{g})$ is the plug-in estimator of the efficient influence curve of Ψ^I at P_0.

This quantity $(\hat{\sigma}_J^I)^2$ can be used to calculate p values and 95% confidence intervals for different parameters, e.g., $\hat{\Psi}^I(\mathbb{P}_{\hat{Q}^*,\hat{g}^*}) \pm 1.96\frac{\hat{\sigma}_J^I}{\sqrt{J}}$ for the target parameter.

Incorporating hierarchical structure for estimating the outcome mechanism

Based on the previously defined community-level TMLE for the mean of the exposure-specific counterfactual community level outcome, we can still incorporate individual level data rather than simply community wide aggregates of that data. As discussed in section (2.2), one typical choice of the community-level counterfactuals of interest is the weighed average response among all individuals sampled from that community, i.e., $Y^c_{j,g^*} = \sum_{i=1}^{N_j}\alpha_{j,i}Y_{j,i,g^*}$. Hence, the conditional mean of the community-level outcome can be rewritten as a weighted average of

the individual-level outcomes, $\bar{Q}_0^c(A, E, \boldsymbol{W}) = \mathbb{E}(Y^c|A, E, \boldsymbol{W}) = \sum_{i=1}^{N} \alpha_i \mathbb{E}(Y_i|A, E, \boldsymbol{W}) \equiv \sum_{i=1}^{N} \alpha_i \mathbb{E}(Y_i|A, E, \boldsymbol{W}, N)$ where the community-specific sample size N is a random variable that is included in the community-level baseline covariates E.

Without changing the underlying structural causal model (1), estimand and efficient influence curve, we may use an individual-level working model to incorporate pooled individual-level outcome regressions as candidates in the Super Learner library for initial estimation of the expected community-level outcome $\bar{Q}_0^c(A, E, \boldsymbol{W})$ given community and individual level covariates, along with community-level exposures. Specially, we propose a working model that assumes that

$$\mathbb{E}_0(Y_i|A, E, \boldsymbol{W}) = \mathbb{E}_0(Y_i|A, E, W_i) = \bar{Q}_0(A, E, W_i) \tag{2.4}$$

for a common function $\bar{Q}_0$. In practice, this working model suggests that each individual's outcome is drawn from a common distribution that may depend on the individual's baseline covariates, together with the intervention and community-level baseline covariates presented in his or her community, but is not directly influenced by the covariates of others in the same community.

Furthermore, the strength of the working assumptions could be weakened by encoding the knowledge of the dependence relationship among individuals within communities, namely, defining E to progressively contain a larger subset of any individual-level covariates included in $\boldsymbol{W}$ [2]. For weak covariate interference, the baseline individual-level covariates of other individuals who are connected with individual i could be included into W_i. Let F_i denote the subset of individuals whose baseline individual-level covariates affect that individual's outcome Y_i, where $i \in F_i$. Now we have a less restricted and more general version of (2.4) as working model:

$$\mathbb{E}_0(Y_i|A, E, \boldsymbol{W}) = \bar{Q}_0(A, E, (W_l : l \in F_i)) \tag{2.5}$$

for a common function $\bar{Q}_0$.

We note that this TMLE never claims that the individual-level working model holds, instead, it uses the working model as a means to generate an initial estimator of $\bar{Q}_0^c$. The implementation of the community-level TMLE incorporating hierarchical data is similar to the previous community-level TMLE, except that the estimation of the community-level outcome could also be based on a single pooled individual level regression $Y_{j,i}$ on (E_j, A_j, $W_{j,i}$) when assuming the aforementioned working model (2.4). As a consequence, the loss functions for the initial estimation of $\bar{Q}_0^c$, can be specified at the individual-level (instead of at the community-level in the previous subsection). For example, we could use a binary log-likelihood loss function:

$$-\mathcal{L}(\bar{Q}^c)(O) = \sum_{j=1}^{J} \sum_{i=1}^{N_j} \alpha_{j,i} \Big[Y_{j,i} \log[\bar{Q}^c(A_j, E_j, \boldsymbol{W}_j)] + (1 - Y_{j,i}) \log[1 - \bar{Q}^c(A_j, E_j, \boldsymbol{W}_j)] \Big],$$

or a squared error loss:

$$\mathcal{L}(\bar{Q}^c)(O) = \sum_{j=1}^{J}\sum_{i=1}^{N_j} \alpha_{j,i}[Y_{j,i} - \bar{Q}^c(A_j, E_j, \boldsymbol{W}_j)]^2$$

where $\boldsymbol{\alpha} = (\alpha_{j,i} : j = 1, \ldots, J, i = 1, \ldots, N_j)$ is a vector of weights for which $\sum_{i=1}^{N_j} \alpha_{j,i} = 1, \forall j$. A common choice of $\alpha_{j,i}$ is $1/N_j$. If the outcome is continuous and we choose the latter loss function, then the fitted parameter $\hat{\beta}_{LS}$ would minimize the above squared error by solving the following first order condition:

$$\begin{aligned}
\frac{d}{d\beta}\mathcal{L}(\bar{Q}^c_\beta)(O) &= \sum_{j=1}^{J}\sum_{i=1}^{N_j} \alpha_{j,i}\frac{d}{d\beta}[Y_{j,i} - \bar{Q}^c_\beta(A_j, E_j, \boldsymbol{W}_j)]^2 \\
&= -2\sum_{j=1}^{J}\sum_{i=1}^{N_j} \alpha_{j,i}[Y_{j,i} - \bar{Q}^c_\beta(A_j, E_j, \boldsymbol{W}_j)]\frac{d}{d\beta}\bar{Q}^c_\beta(A_j, E_j, \boldsymbol{W}_j) \\
&= -2\sum_{j=1}^{J} \frac{d}{d\beta}\bar{Q}^c_\beta(A_j, E_j, \boldsymbol{W}_j)\Big(\sum_{i=1}^{N_j} \alpha_{j,i}[Y_{j,i} - \bar{Q}^c_\beta(A_j, E_j, \boldsymbol{W}_j)]\Big) = 0
\end{aligned}$$

which can easily show that this fitted parameter is identical to the parameter estimated at the community-level (i.e., $\hat{\beta}^c_{LS}$).

Special case where one observation per community

We will now consider a special case where each community has only one individual (i.e., $N = 1$), and so all individual-level baseline covariates can be treated as environmental factors (i.e., $(E, \boldsymbol{W}) = E$).

Nonparametric structural equation model

Consider a NPSEM with structural equations for endogenous variables $X = (E, A, Y)$,

$$\begin{aligned}
E &= f_E(U_E) \\
A &= f_A(E, U_A) \\
Y &= f_{\boldsymbol{Y}}(E, A, U_Y).
\end{aligned} \tag{2.6}$$

with endogenous unmeasured sources of random variation $U = (U_E, U_A, U_Y)$.

Counterfactuals

Let $Y_a = f_Y(E, a, U_Y)$ denote the counterfactual corresponding with setting the treatment $A = a$, thus the community-level counterfactual outcome is the same as the only observation's outcome in community j (i.e., $Y^c_{j,a} \equiv Y_{j,a}$).

Observed data

Now the observed data become $O = (E, A, Y)$. We observe J i.i.d observations on O.

Target parameter on NPSEM

Consider the following parameter of the distribution of (U, X):

$$\Psi^F(P_{U,X,0}) = \mathbb{E}_{U,X}[Y^c_{g^*}] = \mathbb{E}_{U,X}[Y_{g^*}]$$

Identifiability Result

$$\Psi^F(P_{U,X,0}) = \mathbb{E}_{U,X}[Y^c_{g^*}] = \mathbb{E}_E[\mathbb{E}_{g^*}[Y|A^*, E]] \equiv \Psi^I(P_0)$$

Statistical parameter and efficient influence curve

$$\Psi^I(P_0) = \int_{e\in\mathcal{E}} \int_{a\in\mathcal{A}} \bar{Q}^c_0(a, e) g^*(a|e) d\mu_a(a) q_{E,0}(e) d\mu_e(e)$$

with respect to $\mu_a(a)$ and $\mu_e(e)$, where $(\mathcal{A}, \mathcal{E})$ is the common support of (A, E).

The efficient influence curve of the paramter above is:

$$D^I(P_0)(\mathbf{O}) = \frac{g^*}{g_0}(A|E)(Y^c - \bar{Q}^c_0(A, E)) + \mathbb{E}_{g^*}[\bar{Q}^c_0(A, E)|E] - \Psi^I(\mathbb{P}_{Q,g^*})$$

Estimation and inference

The TMLE estimator has the same procedure as the previously presented TMLE does, except that here $(E, \boldsymbol{W}) = E$.

2.4 Estimation and inference under the restricted hierarchical model with no covariate interference

Restricted hierarchical casual model

What if the third type of dependence in model (2.1) mentioned in section 2.3 is weak or even doesn't exist? This is so called "no covariate interference" [60, 2], which describes that each individual's outcome Y_i is sampled from one distribution only depending on the same individual's own baseline covariate W_i, the baseline community-level covariates E, together with the community-level intervention and that individual's unobserved factors (A, U_{Y_I}). Under this working assumption, we have $\mathbb{E}_0(Y_i|A, E, \boldsymbol{W}) = \bar{Q}_0(A, E, W_i)$. Therefore, when background knowledge about Q_0 is sufficient to ensure an assumption that working model

(2.4) holds, this background changes both the underlying hierarchical causal model and the identifiability results, and so the statistical model, estimand, efficient influence curve, etc. The estimation based on this pooled individual-level regression analysis can leverage the hierarchical data structure and pair the i-specific individual-level outcomes and covariates, which may lead to asymptotically more efficient results than a community-level regression analysis.

In this section, we assume such additional knowledge is available and so consider a new hierarchical causal sub-model which restricts the dependence of individuals in a community. The NPSEM that represents the causal relationships among those endogenous variables is now given by:

$$\begin{aligned} E &= f_E(U_E) \\ \boldsymbol{W} &= f_{\boldsymbol{W}}(E, U_{\boldsymbol{W}}) \\ A &= f_A(E, \boldsymbol{W}, U_A) \\ Y_i &= f_Y(E, W_i, A, U_{Y_i}). \\ U_{Y_i} &\perp\!\!\!\perp U_A | E, W_i \end{aligned} \tag{2.7}$$

Here we assume that the conditional distribution of (W_i, Y_i), given (A, E) are common in i.

Target parameter and the statistical parameter

Let assume that there is a common conditional distribution of A given (E, W_i) across all individuals, i.e., $P(A|E, W_i) \equiv g_I(A|E, W)$, where $g_I(A|E, W)$ denotes the individual-level stochastic intervention. Recall that we may be interested in $Y^c_{j,g^*_I} \equiv \sum_{i=1}^{N_j} \alpha_{j,i} Y_{j,i,g^*_I}$, with respect to some individual-level stochastic intervention g^*_I. We assume that the number of individuals is constant in each community (i.e., $N_j = N, \forall j$). Then our causal parameter of interest is defined by

$$\Psi^F(P_{U,X,0}) = \mathbb{E}_{U,X}(Y^c_{g^*_I}) = \mathbb{E}_{U,X}(\sum_{i=1}^{N} \alpha_i Y_{i,g^*_I})$$

Note that all of the identifiability results in section (2.2) can be naturally applied here. Thus, by identifiability,

$$\begin{aligned} \Psi^F(P_{U,X,0}) &= \sum_{i=1}^{N} \alpha_i \mathbb{E}_{U,X}(Y_{i,g^*_I}) \\ &= \sum_{i=1}^{N} \alpha_i \mathbb{E}_{E,\boldsymbol{W},0}\Big\{ \mathbb{E}_{g^*_I}[\bar{Q}_0(A, E, W_i)] \Big| E, W_i \Big\} \equiv \Psi^{II}(P_{Q,g^*_I}) \end{aligned}$$

where $\Psi^{II} : \mathcal{M}^{II} \to \mathcal{R}$ is the target statistical quantity under the key assumptions of identifiability and working assumption (2.4), and $\mathcal{M}^{II}$ is a sub-model of $\mathcal{M}^I$.

The efficient influence curve D^*

Now, the EIC of Ψ^{II} at $P_0 \in \mathcal{M}^{II}$ is given by:

$$\begin{aligned} D^{II}(P_0)(\mathbf{O}) = \sum_{i=1}^{N} \alpha_i [& \frac{g_I^*}{g_{I,0}}(A|E, W_i)(Y_i - \bar{Q}_0(A, E, W_i)) \\ & + \mathbb{E}_{g_I^*}[\bar{Q}_0(A, E, W_i)|E, W_i] - \Psi^{II}(P_{Q,g_I^*})] \end{aligned}$$

where

$$\begin{aligned} D_Y^{II}(P_0)(O_i) &= \frac{g_I^*}{g_{I,0}}(A|E, W_i)(Y_i - \bar{Q}_0(A, E, W_i)) \\ D_{E,W}^{II}(P_0)(O_i) &= \mathbb{E}_{g_I^*}[\bar{Q}_0(A, E, W_i)|E, W_i] - \Psi^{II}(P_{Q,g_I^*}) \end{aligned}$$

Note that now the EIC is a weighted average of the individual-level EICs.

The individual-level TMLE

Estimation of exposure mechanisms $g_{I,0}$ and $g_{I,0}^*$

Here, the individual-level density of the conditional treatment distribution, adjusting for E and the individual specific covariate W_i, is defined as

$$\begin{aligned} g_I(a|e, w_i) &= E_{\boldsymbol{W}}[g_I(a|e, \boldsymbol{W})|W_i = w_i] = E_{\boldsymbol{W}}[g_I(a|e, \boldsymbol{W}_{-i}, \boldsymbol{W}_i)|W_i = w_i] \\ &= \int_{\boldsymbol{w}_{-i}} g_I(a|e, \boldsymbol{w}_{-i}, w_i) P(\boldsymbol{W}_{-i} = \boldsymbol{w}_{-i}|W_i = w_i) d\mu(\boldsymbol{w}_{-i}) \\ &= \int_{\boldsymbol{w}_{-i}} g_I(a|e, \boldsymbol{w}_{-i}, w_i) P(\boldsymbol{W}_{-i} = \boldsymbol{w}_{-i}) d\mu(\boldsymbol{w}_{-i}) \end{aligned}$$

with respect to some dominating measure $\mu(\boldsymbol{w}_{-i})$, and $\boldsymbol{W}_{-i}$ represents an $((N-1) \times p)$ matrix of individual-level covariates, which includes all individuals in the community except that individual i.

Therefore, the estimate of the individual-level stochastic intervention is given by

$$\hat{g}_I(a|e, w_i) = \frac{1}{J} \sum_{j=1}^{J} \int_{\boldsymbol{w}_{j,-i}} \hat{g}_I(a|e_j, \boldsymbol{w}_{j,-i}, w_{j,i}) P_n(\boldsymbol{W}_{j,-i} = \boldsymbol{w}_{j,-i}) d\mu(\boldsymbol{w}_{j,-i})$$

where $\hat{g}_I(a|e_j, \boldsymbol{w}_{j,-i}, w_{j,i})$ can be obtained by the data adaptive methods based histogram-like estimation presented in section (2.3). Besides, the fitting algorithm for $g_{I,0}^*(A^*|E, \boldsymbol{W}_i)$ is equivalent except that A^* is determined by the user-specific stochastic intervention.

Loss function and initial (non-targeted) estimator of $\bar{Q}_0$

First we assume that the community-level outcome regression is a weighted average of common-in-i individual-level outcome regressions, i,e., $\bar{Q}_0^c = \sum_{i=1}^N \alpha_i \bar{Q}_0$, where $\bar{Q}_0(A, E, W_i) = \mathbb{E}_0(Y_i|A, E, W_i)$. Therefore, to gain an initial estimator of $\bar{Q}_0$, we can simply regress the individual-level outcome Y_i onto the exposure, the community-level covariates, and the i-specific individual-level covariates (A, E, W_i). Without loss of generality, we also assume that Y_i is either bounded continuous or discrete for some known range. Then the estimation can be based on, for example, a squared error loss function:

$$\mathcal{L}(\bar{Q}_\beta)(O) = \sum_{j=1}^{J} \sum_{i=1}^{N_j} \alpha_{j,i}[Y_{j,i} - \bar{Q}_\beta(A_j, E_j, W_{j,i})]^2$$

Here, the fitted parameter $\hat{\beta}_{LS}$ can be solved by minimizing the above squared error function.

Loss function and the least favorable fluctuation submodel that spans the efficient influence curve

Here, the targeting step again needs to define a fluctuation parametric submodel for $\hat{\bar{Q}}$, the initial estimator of the individual-level outcome regression, and a corresponding pre-specified loss function. Given the initial estimators $\hat{\bar{Q}}(A_j, E_j, W_{j,i})$, $\hat{g}_I(A_j = a|E_j, W_{j,i})$ and $\hat{g}_I^*(A_j = a|E_j, W_{j,i})$, the targeting step will update the individual-level regression estimator $\hat{\bar{Q}}$ into $\hat{\bar{Q}}^*$, and so the community-level regression estimator $\hat{\bar{Q}}^c = \sum_{i=1}^N \alpha_i \hat{\bar{Q}}$ into $\hat{\bar{Q}}^{c*} = \sum_{i=1}^N \alpha_i \hat{\bar{Q}}^*$, by

1. define a submodel $\hat{\bar{Q}}(\epsilon)$ with parameter ϵ as

   $$logit(\hat{\bar{Q}}(\epsilon)(a, E_j, W_{j,i}) = logit(\hat{\bar{Q}}(a, E_j, W_{j,i}) + \epsilon \hat{H}_{j,i}(a, E_j, W_{j,i})), \forall j = 1, ..., J$$

   where $\hat{H}_j(a, E_j, W_{j,i}) = \frac{\hat{g}^*(A_j=a|E_j,W_{j,i})}{\hat{g}(A_j=a|E_j,W_{j,i})}$ displays the individual-level clever covariate, and the fluctuation parameter ϵ is obtained by a pooled logistic regression of the individual-level outcome Y_i on the individual-level covariate $\hat{H}_i$ with offset $\text{logit}(\hat{\bar{Q}})$. Note that $\hat{\bar{Q}}(\epsilon = 0) = \hat{\bar{Q}}$ at zero fluctuation.

2. define a loss function for the i-specific individual-level outcome regression, such as negative log-likelihood loss function:

   $$-\mathcal{L}(\hat{\bar{Q}}(\epsilon))(O) = Y_i \log[\hat{\bar{Q}}(\epsilon)(A, E, W_i)] + (1 - Y_i) \log[(1 - \hat{\bar{Q}}(\epsilon)(A, E, W_i))],$$

   Then, we use the average of the individual-level loss functions as the loss function for the community-level outcome regression:

   $$\mathcal{L}^{II}(\hat{\bar{Q}}^c(\epsilon))(O) = \sum_{i=1}^{N} \alpha_i \mathcal{L}(\hat{\bar{Q}}(\epsilon))(O)$$

and the derivative of the loss function at zero fluctuation has:

$$\begin{aligned}\frac{d}{d\epsilon}\mathcal{L}^{II}(\hat{\bar{Q}}^c(\epsilon)(O))|_{\epsilon=0} &= \sum_{i=1}^{N}\alpha_i\Big[\frac{\hat{g}_I^*}{\hat{g}_I}(A|E,W_i)(Y_i-\hat{\bar{Q}}(A,E,W_i))\Big]\\ &= D_Y^{II}(\hat{Q},\hat{g}_I)(O)\end{aligned}$$

where $D_Y^{II}(\hat{Q},\hat{g})$ is a component of the EIC D^{II} of Ψ^{II} at $(\hat{Q},\hat{g}_I)$.

3. the updated fit $\hat{\bar{Q}}^*$ is defined as $\hat{\bar{Q}}(\hat{\epsilon}) = logit^{-1}(logit(\hat{\bar{Q}}) + \hat{\epsilon}\hat{H})$, where $\hat{\epsilon}$ minimizes the empirical loss function above:

$$\hat{\epsilon} = \arg\min_{\epsilon}\sum_{j=1}^{J}\sum_{i=1}^{N_j}\alpha_{j,i}\mathcal{L}(\hat{\bar{Q}}(\epsilon))(O_j)$$

4. the updated community-level regression estimator is $\hat{\bar{Q}}^{c*} = \sum_{i=1}^{N}\alpha_i\hat{\bar{Q}}^*$.

The weighted regression intercept-based TMLE can be implemented in a similar way as in section (2.3), except that the targeting step is now based on the individual-level regressions.

Again, applying a similar targeting step to the marginal distribution of $(E,\boldsymbol{W})$ can easily show that the score of ϵ in the fluctuation model spans the second part of the EIC D^{II}:

$$\begin{aligned}\frac{d}{d\epsilon}\mathcal{L}(\hat{Q}_{E,\boldsymbol{W}}(\epsilon))|_{\epsilon=0} &= \sum_{i=1}^{N}\alpha_i\Big[\mathbb{E}_{g^*}[\hat{\bar{Q}}(A,E,W_i)|E,W_i] - \Psi^I(\mathbb{P}_{\hat{Q},\hat{g}_I^*})\Big]\\ &= D_{E,\boldsymbol{W}}^{II}(\hat{Q},\hat{g}_I)(O)\end{aligned}$$

The individual-level TMLE estimator

The substitution estimator of $\Psi^{II}(P_{Q,g_I^*})$ is defined as follows:

$$\Psi^{II}(P_{\hat{Q}^*,\hat{g}_I^*}) = \frac{1}{J}\sum_{j=1}^{J}\sum_{i=1}^{N_j}\alpha_{j,i}\int_{e_j,w_{j,i}}\int_{a_j}\hat{\bar{Q}}^*(a,e_j,w_{j,i})\hat{g}_I^*(a|e_j,w_{j,i})d\mu_a(a)q_{E,\boldsymbol{W}}(e_j,\boldsymbol{w}_j)d\mu_{e,w}(e_j,\boldsymbol{w}_j)$$

Statistical inference for the individual-level TMLE

Since the individual-level TMLE estimator solves the EIC equation $0 = \sum_{j=1}^{J}D^{II}(\hat{Q}^*,\hat{g}_I^*)(O_j)$, the individual-level TMLE variance can be asymptotically estimated as

$$Var(\Psi^{II}(\hat{Q}^*)) \approx \frac{(\hat{\sigma}_J^{II})^2}{J},\ \text{where } (\hat{\sigma}_J^{II})^2 = \frac{1}{J}\sum_{j=1}^{J}D^{II}(\hat{Q}^*,\hat{g}_I^*)(O_j)^2$$

Then the 95% confidence interval for $\Psi^{II}(\mathbb{P}_0)$ is $\hat{\Psi}^{II} \pm \frac{\hat{\sigma}_J^{II}}{\sqrt{J}}$.

Chapter 3

tmleCommunity R Package for target maximum likelihood estimation for community-level data

3.1 Introduction

Motivation

Over the past years, many applications aim to assess the causal effect of treatments assigned at the community level, while data are still collected at the individual level among individuals of the community. In many cases, one wants to evaluate the effect of a stochastic intervention on the community, where all communities in the target population receive probabilistically assigned treatments based on a known specified mechanism (e.g., implementing a community-level intervention policy that target stochastic changes in the behavior of a target population of communities). The development of the **tmleCommunity** package for R was motivated by the increasing demand of a user-friendly tool to estimate the impact of community-level arbitrary exposures in community-independent data structures with a semi-parametric efficient estimator. The **tmleCommunity** package also extends some of the capabilities of **tmlenet** by optionally allowing fexible data-adaptive estimations through **SuperLearner**, **sl3** and **h2oEnsemble** packages, or even user-supplied machine learning algorithms. Besides, it allows for panel data transformation, such as with random effects and fixed effects. **tmleCommunity** is available on github at `https://github.com/chizhangucb/tmleCommunity`.

As a double-robust and asymptotically efficient substitution estimator that respects global constraints of the statistical model, targeted maximum likelihood (or minimum loss-based) estimation (TMLE) provides asymptotically valid statistical inference, with potential reduction in bias and gain in efficiency [79, 80]. In fact, there are two R [61] packages that have been instructive for the development of our package: The **tmle** [24] package performs parameter estimations for a single time point binary intervention for independent and identically distributed (IID) data, including the average treatment effect (ATE), controlled

direct effects (CDE), and the parameters of a marginal structural model (MSM). Besides, [68] developed another R package called **tmlenet**, which provides three estimators for average causal effects (and ATE) for single time point arbitrary interventions (univariate or multivariate; static, dynamic or stochastic) in the context of network-dependent (non-IID) data, including TMLE, the inverse-probability-of-treatment-weighting (IPTW) and the parametric G-computation formula (GCOMP). This package performs logistic regression through `glm` and `speedglm`.

Organization of this chapter

The chapter focuses on the practical usage of the **tmleCommunity** through multiple examples, therefore we omit many of the technical details. For a description of the TMLE framework for independent community data with static community-level interventions, we refer to [2]. For a description of the TMLE of the mean outcome under a stochastic shift intervention for i.i.d data, we refer to [33].

The rest of this chapter is organized as follows. Section 3.2 shows how the **tmleCommunity** package is used to estimate those parameters proposed in the prior section through a few examples, and summarizes the common features of the functions that may be useful to **tmleCommunity** users. Then section 3.3 uses three simulation studies to demonstrate implementation in different observational settings. Section 3.4 discusses the possible extensions to the methodology and the package in the future. In section 3.5 we answer some frequently asked questions regarding the package.

3.2 Implementation in the tmleCommunity package

Estimation of average causal effects for single time point arbitrary interventions in hierarchical data with the **tmleCommunity** package is performed with the main function `tmleCommunity()`, along with the auxiliary function, `tmleCom_Options()` setting additional options that control the estimation algorithms in the package. Note that `tmleCom_Options()` needs to be specified before calling the main function `tmleCommunity()`, otherwise the default settings of all arguments to the `tmleCom_Options()` function will be used in the estimation procedure.

Specification of observed data

The observed data set is passed to `tmleCommunity` through the data argument as a data frame, with the outcome column, the exposure column(s), the baseline covariates columns and possibly the community identifier column (usually numeric values, but no factor values are supported in the package). The data arguments include `Ynode, Anodes, WEnodes, communityID and YnodeDet`, which are all column names or indices in the data frame data that represent the outcome variable, exposure variable(s), community and individual level

baseline covariates, community identifier variable, and indicators of deterministic values of outcome `Ynode`, respectively. Only `Anodes` and `WEnodes` must be specified. If `Ynode` is left unspecified, the left-side of the regression formula in argument `Qform` will be treated as `Ynode`. If `YnodeDet` is not NULL (its corresponding column should be logical or binary), observations received TRUE or 1 as their `YnodeDet` values are assumed to have constant values for their `Ynode`, thus not being used in the final estimation step. If `communityID` is not NULL (its corresponding column could be integer, numeric or character), it supports three options in argument `community.step`, including `"community_level", "individual_level"` and `"PerCommunity"`. Otherwise, it assumes that the data set has no hierarchical structure (thus automatically choose the option `"NoCommunity"`). More details will be discussed in section 3.2.

The other optional arguments related to the data set - `obs.wts`, `community.wts`, and `fluctuation` - are the sampling weights for each observation, the sampling weights for each community, the choice of the fluctuation working model, respectively. Besides, if `fluctuation` is specified as `"logistic"` (the default), continuous outcomes $Y \in [a, b]$ will be bounded into the linear transformed outcome prior to estimating the outcome mechanism.

Specification of estimation method for hierarchical data

`communityID, community.step` and `pooled.Q` are the main arguments to determine the estimation methods for hierarchical data. First, in order to preserve the hierarchical data structure, the data should contain a column with one unique identifier per community and the user must provide the `communityID` argument as a column name or index in data. Failing to provide `communityID` will force the `community.step` argument to be automatically set to `"NoCommunity"` (the default) and to pool data across all communities and treat the data as non-hierarchical.

Second, If `community.step` is specified as `"community_level"`, the observed data will be aggregated to the community-level and the estimation of the corresponding statistical parameter will be analogous to non-hierarchical data structures. Note that `pooled.Q` is in regard to incorporate the pairing of individual-level covariates and outcomes (also known as the working model of "no covariate inference") in community-level TMLE although the working model is not assumed to hold. In other words, when `community.step` is set to `"community_level"`, if `pooled.Q = TRUE`, the pooled individual-level regressions will be added as candidates in the Super Learner library for initial estimation of the outcome mechanism; If `pooled.Q = FALSE`, both outcome and treatment mechanisms are estimated on the community-level and use no individual-level information. Moreover, `community.step` could be specified as `"individual_level"` under the assumption that the working model of "no covariate inference" holds. Third, the stratified TMLE that fits separate outcome and exposure mechanisms for each community can be implemented by setting `community.step` to `"perCommunity"`. Examples with more details will be provided in section 3.3.

Last but not least, the `community.wts` argument can be used to provide the community-level observation weights. If setting to `"size.community"` (the default), each community

is weighted by its (standard deviation scaled) number of individuals and this specification inflates the weight for communities who are underrepresented due to a large degree of missing data. If setting to `"equal.community"`, all communities will be weighted as the same. The user-specified `community.wts` may be passed as a matrix with 2 columns, where the first and second columns contain the identifiers of communities and the corresponding non-negative weights, respectively.

Specification of interventions

The user-supplied interventions are specified by the `f_g0, f_gstar1` and `f_gstar2` arguments. First, an intervention regimen that encodes model knowledge about values of Anodes is specified with the `f_g0` argument, which is either a function or a vector (or a matrix / data frame if exposures are multivariate) that provides true treatment mechanism under observed Anodes. If `f_g0` is specified as a function, a large vector (or a data frame if multivariate) of Anodes will be sampled from the `f_g0` function. Second, an intervention regimen of interest is defined by replacing the conditional density g_0 with a new user-supplied density g^*, and is specified by the `f_gstar1` argument, which takes a function of counterfactual exposures. The function must include "data" as one of its argument names, and return a vector or a data frame of counterfactual exposures evaluated based on `Anodes, WEnodes` (and possibly `communityID`) passed as a named argument `"data"`. In addition, the interventions defined by `f_gstar1` can be static, dynamic or stochastic. For example, for a data set with a binary treatment, a stochastic regime will randomly assign treatment to 30% of the observations, and another deterministically static regime will assign treatment for every observation. The corresponding `f_gstar1` function can be coded as

```
define_f.gstar <- function(prob.val) {
  f.gstar <- function(data, ...) {
    rbinom(n = NROW(data), size = 1, prob = prob.val)
  }
  return(f.gstar)
}
f.gstar_stoch.0.3 <- define_f.gstar(prob.val = 0.3)
f.gstar_determ.1 <- define_f.gstar(prob.val = 1)
```

Alternatively, a deterministic regime can be specified by passing a vector (or a matrix / data frame) to the `f_gstar1` argument with one element per observation (and one column per treatment variable if multivariate). Moreover, `f_gstar1` can be set to a numeric value if that constant exposure will be assigned to all observations. Thus, the other two ways to code `f.gstar_determ.1` in the example above would be

```
f.gstar_vector.1 <- rep(1L, NROW(data))
f.gstar_const.1 <- 1L
```

Specification of estimation algorithms for outcome regressions

As discussed in the previous chapter, the first-stage of the TMLE procedure is to estimate the conditional mean outcome $\bar{Q}_0$. A good initial fit of $\bar{Q}_0$ could reduce reliance on bias reduction from the subsequent targeting step and provide a target parameter estimate with smaller variance. The following optional arguments to the `tmleCom_Options()` and `tmleCommunity()` functions give users control over the initial estimation of $\bar{Q}_0$:

Relevant arguments in `tmleCom_Options()`:

1. `Qestimator` A string specifying the default estimator for fitting $\bar{Q}_0$, including both parametric estimations (`"speedglm__glm"` and `"glm__glm"`) and data-adaptive estimations (`"h2o__ensemble"` and `"SuperLearner"`).

2. `h2ometalearner` A string to pass to `h2o.ensemble`, specifying the prediction algorithm used to learn the optimal combination of the base learners.

3. `h2olearner` A vector of prediction algorithms for training the ensemble's base models.

4. `SL.library` A vector of prediction algorithms to pass to `SuperLearner`.

5. `CVfolds` Optional number of splits in the V-fold cross-validation step for data-adaptive estimation.

Relevant arguments in `tmleCommunity()`:

1. `Qform` Regression formula for $\bar{Q}_0$, with the form of `Ynode ~ Anodes + WEnodes`.

2. `Qbounds` A vector of 2 truncated levels on continuous Y and the initial estimate $\bar{Q}_n$.

3. `alpha` A value keeping $\bar{Q}_n$ bounded away from (0,1) for logistic fluctuation.

Note that a negative Bernoulli log likelihood could be used as a valid loss function for $\bar{Q}_0$ when setting `fluctuation = "logistic"`, even if Y is not binary. Compared to a regular linear fluctuation, a logistic fluctuation assures that all predicted means are in (0, 1) and the corresponding estimates for $\bar{Q}$ respect the global constraints of the observed data model, which reduces bias and variance in small samples [23]. Before performing the estimation procedure, outcomes Y will be bounded by `Qbounds`, and then will be automatically transformed into Y^*, continuous outcomes bounded in (0, 1) where $Y^* = \frac{Y-a}{b-a} \in [0, 1]$. If `Qbounds` is unspecified, the default choice of the range of Y, widened by 10% of its minimum and maximum values, will be used. Besides, if outcomes Y were originally transformed into Y^*, fitting values of the targeted estimates will be automatically mapped back to the original scale. Once `Qbounds` finish bounding the observed outcomes, it will be set to (1 - `alpha`, `alpha`) and used to bound the predicted values for the initial outcome mechanism.

The default estimation algorithm for $\bar{Q}_0$ is set to `"speedglm__glm"`, which relies on the **speedglm** package [14] and uses its function `speedglm.wfit` to fit parametric generalized linear models (GLM) to medium-large data sets. We note that a direct call to `speedglm.wfit`

that requires a model matrix as input is faster than the standard call to `speedglm`. Another regular parametric GLM fitting functoin `glm.fit` (the workhorse of `glm`) is also available for estimating $\bar{Q}_0$ by setting the `Qestimator` argument to `"glm__glm"`. When `speedglm.wfit` or `glm.fit` is called, logistic regression will be used for the initial estimation of $\bar{Q}_0$.

However, in a nonparametric model, the probability distribution of the data is typically unknown and thus parametric models with assumptions that do not use realistic background knowledge and not respect the global constraints of the statistcal model are easily incorrectly specified. The recommended solution for it is to use more flexible machine-learning estimators that adapt the regression function to the data without overfitting the data. **tmleCommunity** relies on the **SuperLearner** and **h2oEnsemble** packages to perform data-adaptive estimation. Based on the oracle properties of V-fold cross validation that minimizes the estimated expected squared error loss function [77], the super learning chooses the best weighted convex combination of candidate estimators in the user-specified library, possibly including both machine learning algorithms and parametric models. One of its most important advantages is that its "free lunch" principle - Including a large variety of prediction algorithms in the super learning library could increase the chance of being consistently estimated, especially when the functional form of the conditional density is unknown. Note that **h2oEnsemble** is another package that provides Super learning method and is based on the **h2o** package (usually used to build models on large datasets).

`Qform` can be used to specify a regression formula that includes `Anodes` and `WEnodes` for $\bar{Q}_0$. The functional form of the formula is only important to parametric estimation algorithms `speedglm.wfit` and `glm.fit`. When using data-adaptive estimation algorithms provided by **SuperLearner** and **h2oEnsemble**, all variables on the right hand side of `Qform` will be treated as predictor variables passed to the candidate estimators, ignoring the original regression formula. If `Qform` is somehow left unspecified, all variables in `Anodes` and `WEnodes` will be treated as predictor covariates. Besides, the library of the candidate estimators can specify different functions of the passed predictor variables, such as `SL.glm.interaction` for second order interaction terms in **SuperLearner**. For more details on creating new wrapper functions for prediction algorithms in **h2oEnsemble** and **SuperLearner**, please see [46] and [59], respectively.

Finally, we realize that sometimes the use of **SuperLearner** and **h2oEnsemble** may fail due to various reasons such as constant responses, so does the use of `speedglm`. Therefore, the `tmleCommunity()` function provides an insurance mechanism for guaranteeing the estimation procedure functions normally even if the chosen algorithm fails: If `"h2o__ensemble"` fails, it falls back on `"SuperLearner"`; If `"SuperLearner"` fails, it falls back on `"speedglm__glm"`; If `"speedglm__glm"` fails, it falls back on `"glm__glm"`. However, `tmleCommunity()` will terminate with an error if the last solution `glm.fit` also fails.

We demonstrate a simple application of the `tmleCommunity` function using user-specified parametric models and super learning library to estimate $\bar{Q}$ in the code chunk below. In this example, we have a simulated data of 1,000 i.i.d. observations with four baseline covariates (`W1, W2, W3` and `W4`), one binary exposure(`A`) and continuous outcome (`Y`). Its true ATE value is 2.80. Code to generate the example dataset is attached in the supplementary

material.

We begin with correctly specified models for $\bar{Q}$. Note that `tmleCommunity` will provide results of three estimators for ATE. In this section, we only care about the non-targeted substitution estimate that uses only an estimate of $\bar{Q}$, thus we can use `"gcomp"` to extract the corresponding results of the MLE estimator.

```
tmleCom_Options(Qestimator = "speedglm__glm")
tmleCom_Qc_ATE <-
  tmleCommunity(data = indSample.bA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"), f_gstar1 = 1,
                f_gstar2 = 0, Qform = "Y ~ W1 + W2 + W3 + W4 + A",
                alpha = 0.995)
c(tmleCom_Qc_ATE$ATE$estimates["gcomp", ],
  tmleCom_Qc_ATE$ATE$vars["gcomp", ])

# [1] 2.816628029 0.004813391
```

What if our assumption of the parametric model for $\bar{Q}$ is incorrect? The result of the misspecified outcome regression is shown next.

```
tmleCom_Options(Qestimator = "speedglm__glm")
tmleCom_Qm_ATE <-
  tmleCommunity(data = indSample.bA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"), f_gstar1 = 1,
                f_gstar2 = 0, Qform = "Y ~ W1 + A", alpha = 0.995)
c(tmleCom_Qm_ATE$ATE$estimates["gcomp", ],
  tmleCom_Qm_ATE$ATE$vars["gcomp", ])

# [1] 3.45848993 0.01460869
```

Next, suppose we do not know its parametric model and need to use the super learning algorithm to estimate $\bar{Q}$. Instead of using the default library, we specify one that contains three prediction algorithms: `SL.glm`, `SL.bayesglm` and `SL.gam` (Note that `SL.gam` calls the `gam` function in the suggested `gam` package that uses generalized additive models [28]).

```
require("SuperLearner")
tmleCom_Options(Qestimator = "SuperLearner", CVfolds = 5,
                SL.library = c("SL.glm", "SL.bayesglm", "SL.gam"))
tmleCom_QSL_ATE <-
  tmleCommunity(data = indSample.bA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"), f_gstar1 = 1,
                f_gstar2 = 0, rndseed = 12345)
c(tmleCom_QSL_ATE$ATE$estimates["gcomp", ],
```

```
tmleCom_QSL_ATE$ATE$vars["gcomp", ])

# [1] 2.809818 0.004797
```

Specification of estimation algorithms for treatment mechanisms

After finishing the initial fit for $\bar{Q}$ in the first stage of TMLE, the next step is to modify the initial estimator by using the estimation of nuisance parameter g_0, in order to make an optimal bias-variance trade off. Recall that the estimate g_n will be used in a clever covariate that defines a parametric fluctuation model to update the initial estimate of $\bar{Q}$. The following arguments to the `tmleCom_Options()` and `tmleCommunity()` functions provide flexibility in how to choose the estimator for g_0:

Relevant arguments in `tmleCom_Options()`:

- `gestimator` A string specifying the default estimator for fitting g_0,similar to `Qstimator` (except that `gestimator` also supports `"sl3_pipelines"`, another data-adaptive estimation method).
- `bin.method` Specify the method for choosing bins when discretizing the conditional continuous exposure variable, including `"equal.mass", "equal.len"` and `"dhist"`.
- `nbins` Number of bins when discretizing a continuous variable.
- `maxncats` The maximum number of unique categories a categorical variable can have.
- `maxNperBin` The maximum number of observations in each bin.
- `poolContinVar` Logical, when fitting a model for binarized continuous variable, if pooling bin indicators across all bins and fit one pooled regression or not
- `savetime.fit.hbars` Logical, if skipping estimation and prediction of exposure mechanism or not, when `f.gstar1 = NULL` and `TMLE.targetStep = "tmle.intercept"`.

Relevant arguments in `tmleCommunity()`:

- `hform.g0` Regression formula for g_0, with the form of `Anode ~WEnodes`.
- `hform.gstar` Regression formula for the user-supplied intervention g^*, with the form of `Anode ~WEnodes`.
- `lbound` One value for bounds on the ratio of the estimate of g^* to the estimate of g_0.
- `h.g0_GenericModel` An object of `GenericModel` **R6** class containing the previously fitted models for $P(A|W, E)$ under observed treatment mechanism g_0.
- `h.gstar_GenericModel` An object of `GenericModel` **R6** class containing the previously fitted models for $P(A^*|W, E)$ under observed treatment mechanism g^*.

- `TMLE.targetStep` TMLE targeting step method, either `"tmle.intercept"` (Default) or `"tmle.covariate"`.

The options for selecting estimation algorithms for the treatment mechanism are similar to those for estimating $\bar{Q}$, and they share the same choices of `h2ometalearner, h2olearner, SL.library`, and `CVfolds`. Beyond that, users can also the **sl3** package to perform data-adaptive estimation for g_0. Note that **sl3** is a modern implementation of the Super Learner algorithm for ensemble learning and model stacking. For model details on creating new wrapper functions in **sl3**, please see [10]. In order to provide a similar insurance mechanism for the estimation process of g_0, even if the chosen algorithm fails, the `tmlecommunity`() function will use the following rule: If `"h2o__ensemble"` fails, it falls back on `"sl3_pipelines"`; If `"sl3_pipelines"` fails, it falls back on `"SuperLearner"`; The rest of the mechanism will be the same as in the estimation process of $\bar{Q}_0$

Also, g_0 can be either estimated by parametric models or data-adaptive algorithms, depending on the choices of `gestimator`. Though the estimation algorithms for $\bar{Q}$ and g_0 could be different as long as the choices of `Qestimator` and `gestimator` differ, the same library is used for all factors of g. For example, the initial fit of $\bar{Q}$ is estimated by **h2oEnsemble** with `"h2o.glm.wrapper"` and `"h2o.randomForest.wrapper"`; A super learning library, including `SL.loess` (with `span = 0.8`), `SL.glmnet`, `SL.knn.20` (with neighborhood size `k = 20`) and `SL.step`, will be used for each factor of g.

It is important to choose the number and position of the bins when discretizing a continuous exposure variable, as the choices affect the variance of the density estimation [66]. Fortunately, the type of each variable will be automatically detected (can be binary, categorical, or continuous) based on the user-specified `maxncats` argument. Recall that `maxncats` provides the maximum number of unique categories a categorical variable can have. So if one variable has more unique categories, it is automatically considered as a continuous variable.

According to [11], a histogram can be used as a graphically descriptive tool where its location of the bins is determined by cutting the empirical cumulative distribution function (ecdf) by a set of parallel lines. First, the `nbins` argument is a tuning parameter that determines the total number of bins of discretization. A cross-validation selector can be applied to data-adaptively select the candidate number of bins, which minimizes variance and maximizes precision. Note that we do not recommend too many bins due to easily violating the positivity assumption.

Next, given a number of bins, we need to choose the most convenient locations (cutoffs) for the bins. There are three alternative approaches that use a histogram to define the bin cutoffs for a continuous variable: equal-mass, equal-length, and a combination of these two methods. In **tmleCommunity** package, the choice of methods `bin.method` together with the other arguments `nbins` and `maxNperBin` can be used to define the values of bin cutoffs. Note that `maxNperBin` provides a user-specified maximum number of observations in each bin.

The default discretization method, equal mass (aka equal area) interval method, is set by passing an argument `bin.method="equal.mass"` to `tmleCom_Options()` prior to calling the main function `tmleCommunity()`. The interval are defined by spanning the support of A into

non-equal length of bins, each containing (approximately) the same number of observations. It is data-adaptive since it tends to be wide where the population density is small, and narrow where the density is large. If `nbins` is `NA` (or is smaller than $\frac{n}{\texttt{maxNperBin}}$), `nbins` will be (re)set to the integer value of $\frac{n}{\texttt{maxNperBin}}$ where n is the total number of observations, and the default setting of `maxNperBin` is 500 observations per interval. This method could identify spikes in the density, but oversmooths in the tails and so could not discover outliers.

Besides, equal length interval method is set by passing an argument `bin.method = "equal.len"` to `tmleCom_Options()`. The intervals are defined by spanning the support of a continuous variable into `nbins` number of equal length of bins. This method describes the tails of the density and identifies outliers well, but oversmooths in regions of high density and so is poor at identifying sharp peaks. Moreover, as an alternative to find a compromise between equal mass and equal length approaches, the combination method is set by passing an argument `bin.method="dhist"`, where `dhist` is named for diagonally cut histogram. For consistency, We choose the slope $a = 5 \times \text{IQR(A)}$ as suggested by [11].

Similar to `Qform`, formulae that include `WEnodes` can be specified for estimating components of g and g^* using the `hform.g0` and `hform.gstar` arguments. The functional form of the formulae is unimportant when the data-adaptive estimation algorithms are used. Also, if the `hform.g0` and `hform.gstar` arguments are unspecified, the formulae will default to main term regressions that includes all variables in `WEnodes`.

The `lbound` argument is a tuning parameter, conforming with the theoretical assumption 2 in section 2.2 that the ratio of $g^*(a|E, \boldsymbol{W})$ to $g(a|E, \boldsymbol{W})$ must be bounded away from $+\infty$. Since the function $g^*(a|E, \boldsymbol{W})$ is user given, we can try to define it in a way so that it could be used to answer the causal question of interest, and yet it does not produce unstable weights. However, when there are unstable weights that cause extremely large value of $\frac{g^*(a|E,\boldsymbol{W})}{g(a|E,\boldsymbol{W})}$, this lack of identifiability will result in the estimates with high variance [79]. A common approach to reduce the variance of the consequent estimates is bounding $\frac{g^*(a|E,\boldsymbol{W})}{g(a|E,\boldsymbol{W})}$ away from the extremely large value, e.g., $0 \leq \frac{g^*(a|E,\boldsymbol{W})}{g(a|E,\boldsymbol{W})} \leq \frac{1}{\texttt{lbound}}$. However, truncation comes at a price of bias since the consistency of the estimator of $g(a|E, \boldsymbol{W})$ may be affected. Therefore, the `lbound` argument should be chosen carefully (it defaults to 0.005).

`TMLE.targetStep` specifies how to use weights $\frac{h_{g^*}}{h_{g_N}}$ in the TMLE targeting step. If it is set to `"tmle.intercept"` (default), it performs the weighted intercept-based TMLE that runs a intercept-only weighted logistic regression using offsets $\text{logit}(Q^*)$ and weights $\frac{h_{g^*}}{h_{g_N}}$ and so no covariate. If setting to `"tmle.covariate"`, it performs the unweighted covariate-based TMLE that run an unweighted logistic regression using offsets $\text{logit}(Q^*)$ and a clever covariate $\frac{h_{g^*}}{h_{g_N}}$.

The following example illustrates IPTW estimation of the average causal effect of individual-based continuous intervention at a single time point. A sample of 5,000 is generated, with each row i consisting of four baseline covariates (`W1, W2, W3` and `W4`), one continuous exposure (`A`) and continuous outcome (`Y`). The true value for the marginal treatment effect of the intervention for the simulated data is $\psi_0 = 3.46601$. For details on code to generated the

example dataset, please see the supplementary material.

Suppose we are interested in estimating the mean outcome ψ_0 under a truncated stochastic intervention g^*, which is defined by shifting the normal density of observed A until $\frac{g^*}{g_0} \geq 10$ and then its truncated to be equal to g_0. In this case, the `tmleCommunity()` function receives only one user-specified intervention, and we should utilize `$EY_gstar1` to extract the results of estimates under the intervention `f_gstar1`. Moreover, we use `"iptw"` to display the results of the IPTW estimator since it relies only on the estimate of g. Let's begin with correctly specified models for g_0 and g^* with a shift of 2 on the observed A, using the equal mass method that discretizes A into 5 bins (all default choices).

```
define_f.gstar <- function(shift.val, truncBD, rndseed = NULL) {
  f.gstar <- function(data, ...) {
    set.seed(rndseed)
    A.mu <- 0.86 * data$W1 + 0.93 * data$W3 * data$W4 + 0.41 * data$W4
    untrunc.A <- rnorm(n = nrow(data), mean = A.mu + shift.val, sd = 1)
    r.new.A <- exp(0.5 * shift.val * (untrunc.A - A.mu - shift.val / 2))
    trunc.A <- ifelse(r.new.A > truncBD, untrunc.A - shift.val, untrunc.A)
    return(trunc.A)
  }
  return(f.gstar)
}
f.gstar <- define_f.gstar(shift.val = 2, truncBD = 10, rndseed = 1)

gform.C <- "A ~ W1 + W3 * W4"  # correct gform
N <- NROW(indSample.cA.cY)
tmleCom_Options(gestimator = "speedglm__glm", bin.method = "equal.mass",
                nbins = 5, maxNperBin = N)
tmleCom_gc_default <-
  tmleCommunity(data = indSample.cA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"), f_gstar1 = f.gstar,
                rndseed = 1, hform.g0 = gform.C, hform.gstar = gform.C)
c(tmleCom_gc_default$EY_gstar1$estimates["iptw", ],
  tmleCom_gc_default$EY_gstar1$vars["iptw", ])

# [1] 3.4406569 0.0120417
```

Note that if the discretization method is equal mass (i.e., `bin.method = "equal.mass"`), and each bin is allowed to contain no more than 250 observations (i.e., `maxNperBin` = 250), the number of bins (or regressions) will be set to the larger value between the nearest interger of $\frac{n}{250}$ and the value of `nbins`, where n is the total number of observations. Thus, even if `nbins` defaults to 5, the real number of bins for a sample of 5000 will be 20. It is worth mentioning that during the estimation, $-\infty$ and $+\infty$ are added as leftmost and

rightmost cutoff points to make sure all future data points end up in one of the intervals. For example, if the real number of bins is 20, then the returned results will include 22 fitted models. However, the selection of `maxNperBin` doesn't have influence on the real number of bins when using the equal length and combination methods.

```
tmleCom_Options(maxNperBin = 250, bin.method = "equal.mass")
tmleCom_gmain_eqmass <-
  tmleCommunity(data = indSample.cA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"), f_gstar1 = f.gstar)
h.g0_models_mass <- tmleCom_gmain_eqmass$EY_gstar1$h.g0_GenericModel
length(h.g0_models_mass$getPsAsW.models()$`P(A|W).1`$bin_nms)

# [1] 22

tmleCom_Options(maxNperBin = 250, bin.method = "equal.len")
tmleCom_gmain_eqlen <-
  tmleCommunity(data = indSample.cA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"), f_gstar1 = f.gstar)
h.g0_models_len <- tmleCom_gmain_eqlen$EY_gstar1$h.g0_GenericModel
length(h.g0_models_len$getPsAsW.models()$`P(A|W).1`$bin_nms)

# [1] 7
```

As mentioned previously, when `f.gstar1` is inadvertently unspecified (i.e., `f.gstar1 = NULL`) and `TMLE.targetStep = "tmle.intercept"`, setting `savetime.fit.hbars` to `TRUE` allows the TMLE process to skip the estimation and prediction of exposure mechanism $P(A|W,E)$ under g_0 and g^*. It will directly set $\frac{h_{g^*}}{h_{g_N}}$ to 1 for all observations.

```
tmleCom_nofgstar <-
  tmleCommunity(data = indSample.cA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"), f_gstar1 = NULL,
                Qform = "Y ~ W1 + W2 + W3 + W4 + A")
c(tmleCom_nofgstar$EY_gstar1$h.g0_GenericModel,
  tmleCom_nofgstar$EY_gstar1$h.gstar_GenericModel)

# [1] NULL
```

If instead we would like to estimate the same parameter except using machine learning algorithms. R code that uses **SuperLearner** to estimate ψ_0 is shown next. It displays a satisfactory result of estimation with a super learning library containing `"SL.glm"`, `"SL.gam"` and `"SL.randomForest"`.

```
require("SuperLearner")
tmleCom_Options(gestimator = "SuperLearner", maxNperBin = N,
                SL.library = c("SL.glm", "SL.gam"))
tmleCom_gSL_default <-
  tmleCommunity(data = indSample.cA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"), f_gstar1 = f.gstar,
                Qform = "Y ~ W1 + W2 + W3 + W4 + A", rndseed = 1)
c(tmleCom_gSL_default$EY_gstar1$estimates["iptw", ],
  tmleCom_gSL_default$EY_gstar1$vars["iptw", ])

# [1] 3.4471211 0.0124127
```

Another choice for performing maching leraning algorithms, especially stacked ensemble learning, is to use the **sl3** package. Example R code for estimating ψ_0 is shown next. Both `Lrnr_glm_fast` and `Lrnr_glmnet` are used in the library.

```
require("sl3"); require("SuperLearner")
sl3Options("sl3.verbose", TRUE)
tmleCom_Options(gestimator = "sl3_pipelines", maxNperBin = N, CVfolds = 5,
                sl3_learner = list(glm_fast = make_learner(Lrnr_glm_fast),
                                   glmnet = make_learner(Lrnr_glmnet)),
                sl3_metalearner = make_learner(
                  Lrnr_optim, loss_function = loss_squared_error,
                  learner_function = metalearner_logistic_binomial))
tmleCom_gsl3_default <-
  tmleCommunity(data = indSample.cA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"),
                f_gstar1 = f.gstar, rndseed = 1)
c(tmleCom_gsl3_default$EY_gstar1$estimates["iptw", ],
  tmleCom_gsl3_default$EY_gstar1$vars["iptw", ])

# [1] 3.44918346 0.01457839
```

Recall that the **h2oEnsemble** package could also perform Super learning methods. In this case, we apply generalized linear models with penalized maximum likelihood for both base learners that are used to train the base models for the ensemble, and the metalearner that is used to learn the optimal combination of the base learners. Specifically, the base learners will include three regressions: Lasso ($\alpha = 1$), Ridge ($\alpha = 0$) and Elastic net models with $\alpha = 0.5$.

```
require("h2oEnsemble")
h2o.glm.1 <- function(..., alpha = 1, prior = NULL) {
  h2o.glm.wrapper(..., alpha = alpha, , prior=prior)
```

```
}
h2o.glm.0.5 <- function(..., alpha = 0.5, prior = NULL) {
  h2o.glm.wrapper(..., alpha = alpha, , prior=prior)
}
h2o.glm.0 <- function(..., alpha = 0, prior = NULL) {
  h2o.glm.wrapper(..., alpha = alpha, , prior=prior)
}
tmleCom_Options(gestimator = "h2o__ensemble", maxNperBin = N,
                h2ometalearner = "h2o.glm.wrapper",
                h2olearner = c("h2o.glm.1", "h2o.glm.0.5", "h2o.glm.0"))
tmleCom_gh2o_default <-
  tmleCommunity(data = indSample.cA.cY, Ynode = "Y", Anodes = "A",
                WEnodes = c("W1", "W2", "W3", "W4"),
                f_gstar1 = f.gstar, rndseed = 1)
c(tmleCom_gh2o_default$EY_gstar1$estimates["iptw", ],
  tmleCom_gh2o_default$EY_gstar1$vars["iptw", ])

# [1] 3.4321917 0.0118350
```

Summary of key arguments to the tmleCommunity function

For full details, see the documentation for the **tmleCommuity** package (cite ***).

- `data` Observed data, `data.frame` with named columns, containing `WEnodes`, `Anode`, `Ynode` and possibly `communityID`, `YnodeDet`.
- `Ynode` Column name or index in `data` of outcome variable. Outcome can be either binary or continuous (could be beyond 0 and 1). If `Ynode` undefined, the left-side of the regression formula in argument `Qform` will be treated as `Ynode`.
- `Anodes` Column names or indices in `data` of exposure (treatment) variables.
- `WEnodes` Column names or indices in `data` of individual-level (and possibly community-level) baseline covariates. Factors are not allowed.
- `YnodeDet` Optional column name or index in `data` of indicators of deterministic values of outcome `Ynode`, coded as (`TRUE` / `FALSE`) or (1 / 0). If TRUE or 1, value of `Ynode` is given deterministically / constant
- `obs.wts` Optional choice to provide/ construct a vector of individual-level observation (sampling) weights (of length `nrow(data)`). Currently supports a non-negative numeric vector, `"equal.within.pop"` (Default) and `"equal.within.community"`. If `"equal.within.pop"`, weigh individuals in the entire dataset equally (weigh to be

all 1); If `"equal.within.community"`, weigh individuals within the same community equally (i.e., 1 / (number of individuals in each community)).

- `community.step` Methods to deal with hierarchical data, user needs to specify one of the four choices: `"NoCommunity"` (Default), `"community_level"`, `"individual_level"`, `"PerCommunity"`. If `"NoCommunity"`, claim that no hierarchical structure in data; If `"community_level"`, use the community-level TMLE; If `"individual_level"`, use the individual-level TMLE cooperating with the assumption of no covariate interference, Finally if `"perCommunity"`, use stratified TMLE. If `communityID` = `NULL`, then automatically pool over all communities (i.e., treated it as `"NoCommunity"`).

- `communityID` Optional column name or index in `data` representing community identifier variable. If known, it can support the three options within `community.step`: `"community_level"`, `"individual_level"` and `"PerCommunity"`.

- `community.wts` Optional choice to provide/ construct a matrix of community-level observation weights (where dimension = $J \times 2$, where J = the number of communities). The first column contains the identifiers / names of communities (ie., `data[ , communityID]`) and the second column contains the corresponding non-negative weights. Currently only support a numeric matrix with 2 columns, `"size.community"` (Default) and `"equal.community"`. If setting `community.wts` = "size.community", treat the number of individuals within each community as its weight, respectively. And if `community.wts` = "equal.community", assumed weights to be all 1.

- `pooled.Q` Logical for incorporating hierarchical data to estimate the outcome mechanism. If `TRUE`, use a pooled individual-level regression for initial estimation of the mean outcome (i.e., outcome mechanism). Default to be `FALSE`.

- `f_g0` Optional function used to specify model knowledge about value of Anodes. It estimates $P(A|W, E)$ under `g0` by sampling a large vector/ data frame of `Anode` (of length `nrow(data)*n_MCsims` or number of rows if a data frame) from `f_g0` function.

- `f_gstar1` Either a function or a vector or a matrix/ data frame of counterfactual exposures, depending on the number of exposure variables. If a matrix/ data frame, its number of rows must be either `nrow(data)` or 1 (constant exposure assigned to all observations), and its number of columns must be `length(Anodes)`. Note that the column names should match with the names in `Anodes`. If a vector, it must be of length `nrow(data)` or 1. If a function, it must return a vector or a data frame of counterfactual exposures sampled based on `Anodes`, `WEnodes` (and possibly `communityID`) passed as a named argument "data". Thus, the function must include "data" as one of its argument names. The interventions defined by `f_gstar1` can be static, dynamic or stochastic.

- `f_gstar2` Either a function or a vector or a matrix/ data frame of counterfactual exposures, depending on the number of exposure variables. It has the same components and requirements as `f_gstar1` has.

- `Qform` Character vector of regression formula for Ynode. If not specified (i.e., `NULL`), the outcome variable is regressed on all covariates included in Anodes and WEnodes (i.e., `Ynode ~ Anodes + WEnodes`).

- `Qbounds` Vector of upper and lower bounds on Y and predicted value for initial Q. Default to the range of Y, widened by 10% of the min and max values.

- `alpha` Used to keep predicted values for initial Q bounded away from (0,1) for logistic fluctuation (set `Qbounds` to (1 - `alpha`), `alpha`).

- `fluctuation` Default to "logistic", it could also be "linear" (for targeting step).

- `hform.g0` Character vector of regression formula for estimating the conditional density of $P(A|W,E)$ under observed treatment mechanism g_0. If not specified, its form will be `Anodes ~WEnodes`. If there are more than one exposure, it fits a joint probability.

- `hform.gstar` Character vector of regression formula for estimating the conditional density $P(A|W,E)$ under user-supplied interventions `f_gstar1` or `f_gstar2`. If not specified, it use the same regression formula as used in `hform.g0`.

- `lbound` Value between (0,1) for truncation of predicted $P(A|W,E)$. Default to 0.005.

- `h.g0_GenericModel` An object of `GenericModel` **R6** class containing the previously fitted models for $P(A|W,E)$ under observed treatment mechanism g_0, one of the returns of `tmleCommunity` function. If known, predictions for $P(A=a|W=w,E=e)$ under g_0 are based on the fitted models in `h.g0_GenericModel`.

- `h.gstar_GenericModel` An object of `GenericModel` **R6** class containing the previously fitted models for $P(A^*|W,E)$ under intervention `gstar`, one of the returns of `tmleCommunity` function. If known, predictions for $P(A=a|W=w,E=e)$ under `gstar` are based on the fitted models in `h.gstar_GenericModel`.

- `TMLE.targetStep` TMLE targeting step method, either `"tmle.intercept"` (Default) or `"tmle.covariate"`.

- `n_MCsims` Number of simulations for Monte-Carlo analysis. Each simulation generates new exposures under `f_gstar1` or `f_gstar2` (if specified) or `f_g0` (if specified), with a sample size of nrow(data). Then these generated exposures are used when fitting the conditional densities $P(A|W,E)$ and estimating for IPTW and GCOMP under intervention `f_gstar1` or `f_gstar2` . Note that deterministic intervention only needs one simulation and stochastic intervention could use more simulation times such as 10 (Default to 1).

- `CI_alpha` Significance level (alpha) used in constructing a confidence interval. Default to 0.05.

- `rndseed` Random seed for controlling sampling A under `f_gstar1` or `f_gstar2` (for reproducibility of Monte-Carlo simulations)

- `verbose` Flag. If `TRUE`, print status messages. Default to `FALSE`. It can be turned on by setting `options(tmleCommunity.verbose = TRUE)`.

Summary of key arguments to the tmleComOptions function

For full details, see the documentation for the **tmleCommuity** package (cite ***).

- `Qestimator` A string specifying default estimator for outcome mechanism model fitting. The default estimator is `"speedglm__glm"`, which estimates regressions with `speedglm.wfit`; Estimator `"glm__glm"` uses `glm.fit`; Estimator `"h2o__ensemble"` implements the super learner ensemble (stacking) algorithm using the H2O R interface; Estimator `"SuperLearner"` implements the super learner prediction methods. Note that if `"h2o__ensemble"` fails, it falls back on `"SuperLearner"`. If `"SuperLearner"` fails, it falls back on `"speedglm__glm"`. If `"speedglm__glm"` fails, it falls back on `"glm__glm"`.

- `gestimator` A string specifying default estimator for exposure mechanism fitting. It has the same options as `Qestimator`.

- `bin.method` Specify the method for choosing bins when discretizing the conditional continuous exposure variable `A`. The default method is `"equal.mass"`, which provides a data-adaptive selection of the bins based on equal mass/ area, i.e., each bin will contain approximately the same number of observations as others. Method `"equal.len"` partitions the range of `A` into equal length `nbins` intervals. Method `"dhist"` uses a combination of the above two approaches. Please see Denby and Mallows "Variations on the Histogram" (2009) for more details.

- `nbins` When `bin.method = "equal.len"`, set to the user-supplied number of bins when discretizing a continuous variable. If not specified, then default to 5; If setting to as `NA`, then set to the nearest integer of `nobs/ maxNperBin`, where `nobs` is the total number of observations in the input data. When method is `"equal.mass"`, `nbins` will be set as the maximum of the default `nbins` and the nearest integer of `nobs/ maxNperBin`.

- `maxncats` Integer that specifies the maximum number of unique categories a categorical variable `A[j]` can have. If `A[j]` has more unique categories, it is automatically considered a continuous variable. Default to 10.

- `maxNperBin` Integer that specifies the maximum number of observations in each bin when discretizing a continuous variable `A[j]` (applies directly when `bin.method = "equal.mass"` and indirectly when `bin.method = "equal.len"`, but `nbins = NA`).

- `parfit` Logical. If `TRUE`, perform parallel regression fits and predictions for discretized continuous variables by functions `foreach` and `dopar` in `foreach` package. Default to `FALSE`. Note that it requires registering a parallel back-end prior to running `tmleCommunity` function, for example, using `doParallel` R package and running `registerDoParallel(cores = ncores)` for `ncores` parallel jobs.

- `poolContinVar` Logical. If `TRUE`, when fitting a model for binarized continuous variable, pool bin indicators across all bins and fit one pooled regression. Default to `FALSE`.

- `savetime.fit.hbars` Logical. If `TRUE`, skip estimation and prediction of exposure mechanism P(A|W,E) under $g_0 \& g^*$ when `f.gstar1 = NULL` and `TMLE.targetStep = "tmle.intercept"`, and then directly set `h_gstar_h_gN = 1` for each observation. Default to `TRUE`.

- `h2ometalearner` A string to pass to `h2o.ensemble`, specifying the prediction algorithm used to learn the optimal combination of the base learners. Supports both h2o and SuperLearner wrapper functions. Default to "h2o.glm.wrapper".

- `h2olearner` A string or character vector to pass to `h2o.ensemble`, naming the prediction algorithm(s) used to train the base models for the ensemble. The functions must have the same format as the h2o wrapper functions. Default to "h2o.glm.wrapper".

- `CVfolds` Set the number of splits for the V-fold cross-validation step to pass to both `SuperLearner` and `h2o.ensemble`. Default to 5.

- `SL.library` A string or character vector of prediction algorithms to being set to `SuperLearner`. Default to `c("SL.glm", "SL.step", "SL.glm.interaction")`. For more available algorithms see `SuperLearner::listWrappers()`.

3.3 Simulation studies with community-level interventions

Simulation 1 - Stochastic interventions

We perform a simulation study evaluating the finite sample bias and variance of the TMLE presented in the previous chapter, including both community-level and individual-level TMLE. Besides, we compare the performance of TMLE estimator with that of Inverse-Probability-of-Treatment-Weighted estimator (IPTW) and parametric G-computation formula estimator (GCOMP). In order to estimate the average causal effect of community-level intervention(s) at a single time point on an individual-based outcome, we simulate a data set consisting of 1000 independent communities, each (community j) containing n_j (non-fixed) number of individuals where n_j is drawn from a normal distribution with mean 50 and standard

deviation 10 and rounded to the nearest integer. First, we sample n_j i.i.d. community-level baseline covariates (E_1, E_2), distributed as

$$n_j \sim N(50, 10) \qquad E_{1,j} \sim Unif(0,1) \qquad E_{2,j} \sim Unif\{0.2, 0.4, 0.6, 0.8\}$$

Then 3 dependent individual-level baseline covariates (W_1, W_2, W_3) are drawn as a function of community-level baseline covariates, respectively.

$$\boldsymbol{W}_{1,n_j} \sim (Bern(expit(-0.4 + 1.2E_{1,j} - 1.3E_{2,j})))_{i=1,\ldots,n_j}$$
$$\begin{pmatrix} \boldsymbol{W}_{2,n_j} \\ \boldsymbol{W}_{3,n_j} \end{pmatrix} \sim N(\begin{array}{c} 1 - 0.8E_{1,j} - 0.4E_{2,j} \\ 0.5 + 0.2E_{1,j} \end{array}, \Sigma = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix})$$

And a community-level continuous treatment A is sampled conditionally on the values of all baseline covariates.

$$A_j \sim N(-1.2 + 0.8E_1 + 0.21E_2 + 3W^c_{1,n_j} - 0.7W^c_{2,n_j} + 0.3W^c_{3,n_j}, 1)$$
$$\text{where} \quad W^c_{1,n_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} \boldsymbol{W}_{1,n_j} \qquad W^c_{2,n_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} \boldsymbol{W}_{2,n_j} \qquad W^c_{3,n_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} \boldsymbol{W}_{3,n_j}$$

Also a truncated stochastic intervention g^* is defined by shifting the normal density of observed A by some known constant $shift > 0$ until $\frac{g^*}{g_0}$ exceeds a known constant $bound$, and otherwise, the intervention keeps the observed exposure A unchanged. So the intervened exposure A^*_j is distributed as

$$A^*_j = \begin{cases} A_j + shift & exp\{1.5 * shift * (A_j - \mu(E_j, W^c_{n_j})) - \frac{shift}{4}\} > truncbd \\ A_j, & o.w. \end{cases}$$
$$\text{where} \quad \mu(E_j, W^c_{n_j}) = -1.2 + 0.8E_1 + 0.21E_2 + 3W^c_{1,n_j} - 0.7W^c_{2,n_j} + 0.3W^c_{3,n_j}$$

Last, the individual-level binary outcome Y that is a function of treatment and all baseline covariates is simulated. Similarly, the post-intervened outcome Y^*, under stochastic intervention g^*, is defined as

- **Case 1**: Working model holds

$$Y_j \sim Bern(expit(-1.7 + 1.7A_j + 0.5E_{1,j} - 1.2E_{2,j} + 1.1\boldsymbol{W}_{1,n_j} + 1.3\boldsymbol{W}_{2,n_j} - 0.4\boldsymbol{W}_{3,n_j}))$$
$$Y^*_j \sim Bern(expit(-1.7 + 1.7A^*_j + 0.5E_{1,j} - 1.2E_{2,j} + 1.1\boldsymbol{W}_{1,n_j} + 1.3\boldsymbol{W}_{2,n_j} - 0.4\boldsymbol{W}_{3,n_j}))$$

- **Case 2**: Working model is not a reasonable approximation

$$Y_j \sim Bern(expit(-1.7 + 1.2A_j - 0.2E_{1,j} + 1.1E_{2,j} + 5.8\boldsymbol{W}^c_{1,n_j} - 3.1\boldsymbol{W}^c_{2,n_j} - \boldsymbol{W}^c_{3,n_j} + 0.4\boldsymbol{W}_{1,n_j} + 0.2\boldsymbol{W}_{2,n_j} - 0.4\boldsymbol{W}_{3,n_j}))$$
$$Y^*_j \sim Bern(expit(-1.7 + 1.2A^*_j - 0.2E_{1,j} + 1.1E_{2,j} + 5.8\boldsymbol{W}^c_{1,n_j} - 3.1\boldsymbol{W}^c_{2,n_j} - \boldsymbol{W}^c_{3,n_j} + 0.4\boldsymbol{W}_{1,n_j} + 0.2\boldsymbol{W}_{2,n_j} - 0.4\boldsymbol{W}_{3,n_j}))$$

The next code chunk shows how to simulate a data set according to the previous data generating distributions where the working model fails. The code also defines the stochastic intervention g^* that we are interested in. Assuming that we want to evaluate the effect of a constant shift of 1, given a truncation bound of 5, the true parameter value under this stochastic intervention is $\psi_0 = 0.558$.

```
getY <- function(A, E1, E2, W1, W2, W3, bs, n.ind) {
  prob.Y <- plogis(bs[1] + bs[2] * A + bs[3] * E1 + bs[4] * E2
                   + bs[5] * mean(W1) + bs[6] * mean(W2) + bs[7] * mean(W3)
                   + bs[8] * W1 + bs[9] * W2 + bs[10] * W3)
  rbinom(n = n.ind, size = 1, prob = prob.Y)
}

get.cluster.Acont <- function(id, n.ind, truncBD = 5, shift = 1,
                              working.model = T) {
  # Construct community- & individual-level baseline covariates E, W
  E1 <- runif(n = 1, min = 0, max = 1)
  E2 <- sample(x = c(0.2, 0.4, 0.6, 0.8), size = 1)
  prob.W1 <- plogis(- 0.4 + 1.2 * E1 - 1.3 * E2)
  W1 <- rbinom(n = n.ind, size = 1, prob = prob.W1)
  W2_mean <- 1 - 0.8 * E1 - 0.4 * E2
  W3_mean <- 0.5 + 0.2 * E1
  W2W3 <- MASS::mvrnorm(n = n.ind, mu = c(W2_mean, W3_mean),
                        Sigma = matrix(c(1, 0.6, 0.6, 1), ncol = 2))
  W2 <- W2W3[, 1]
  W3 <- W2W3[, 2]
  A.mu <- - 1.2 + 0.8 * E1 + 0.21 * E2 + 3 * mean(W1) -
    0.7 * mean(W2) + 0.3 * mean(W3)
  A <- rnorm(n = 1, mean = A.mu, sd = 1)
  untrunc.A.gstar <- A + shift
  r.new.A <- exp(1.5 * shift * (untrunc.A.gstar - A.mu - shift / 4))
  trunc.A.gstar <- ifelse(r.new.A > truncBD, A, untrunc.A.gstar)
  if (working.model) { # when working.model holds
    # betas <- c(-1.7, 1.2, 0.5, -1.2, 0, 0, 0, 0.7, 1.3, -0.4)
    betas <- c(-1.7, 1.7, 0.5, -1.2, 0, 0, 0, 1.1, 1.3, -0.4)
  } else { # when working.model fails
    betas <- c(-1.7, 1.2, -0.2, 1.1, 5.8, -3.1, -1, 0.4, 0.2, -0.4)
  }
  Y <- getY(A, E1, E2, W1, W2, W3, betas, n.ind)
  Y.gstar <- getY(trunc.A.gstar, E1, E2, W1, W2, W3, betas, n.ind)
  return(data.frame(cbind(id, E1, E2, W1, W2, W3, A, Y, Y.gstar)))
}
```

```
get.fullDat.Acont <- function(J, n.ind, truncBD = 5, shift = 1,
                              working.model = T, n.fix = F, only.Y = F) {
  if (n.fix) {
    n.ind <- rep(n.ind, J)
  } else {  # don't fix the number of obs in each community
    n.ind <- round(rnorm(J, n.ind, 10))
    n.ind[n.ind <= 0] <- n.ind
  }
  if (only.Y) {id <- Y <- Y.gstar <- NULL } else { full.dat <- NULL}
  for(j in 1:J) {
    cluster.data.j <- get.cluster.Acont(working.model = working.model,
      id = j, n.ind = n.ind[j], truncBD = truncBD, shift = shift)
    if (only.Y) {
      id <- c(id, cluster.data.j[, "id"])
      Y <- c(Y, cluster.data.j[, "Y"])
      Y.gstar <- c(Y.gstar, cluster.data.j[, "Y.gstar"])
    } else {
      full.dat <- rbind(full.dat, cluster.data.j)
    }
    print(j)
    if (!only.Y) { full.dat$id <- as.integer(full.dat$id) }
  }
  ifelse(only.Y,
         return(data.frame(cbind(id, Y, Y.gstar))), return(full.dat))
}

PopDat.wmT <- get.fullDat.Acont(J = 3000, n.ind = 1000, truncBD = 5,
                                shift = 1, working.model = T, only.Y = T)
PopDat.wmT.agg <- aggregate(PopDat.wmT, by=list(PopDat.wmT$id), mean)
truth.wmT <- mean(PopDat.wmT.agg$Y.gstar)

PopDat.wmF <- get.fullDat.Acont(J = 4000, n.ind = 1000, truncBD = 5,
                                shift = 1, working.model = F, only.Y = T)
PopDat.wmF.agg <- aggregate(PopDat.wmF, by=list(PopDat.wmF$id), mean)
truth.wmF <- mean(PopDat.wmF.agg$Y.gstar)

comSample.wmF <- get.fullDat.Acont(
  J = 1000, n.ind = 50, truncBD = 5, shift = 1, working.model = F)
comSample.wmF$Y.gstar <- NULL

define_f.gstar <- function(shift.val, truncBD, rndseed = NULL) {
```

```
  f.gstar <- function(data, ...) {
    set.seed(rndseed)
    A.mu <- - 1.2 + 0.8 * data$E1 + 0.21 * data$E2 +
      3 * mean(data$W1) - 0.7 * mean(data$W2) + 0.3 * mean(data$W3)
    untrunc.A <- rnorm(n = nrow(data), mean = A.mu + shift.val, sd = 1)
    r.new.A <- exp(1.5 * shift.val * (untrunc.A - A.mu - shift.val / 4))
    trunc.A <- ifelse(r.new.A > truncBD, untrunc.A - shift.val, untrunc.A)
    return(trunc.A)
  }
  return(f.gstar)
}
f.gstar <- define_f.gstar(shift.val = 1, truncBD = 5)
```

We first demonstrate how to use the two distinct approaches for leveraging a hierarchical data structure. Recall that the first approach treats community rather than individual as the unit of analysis and performs estimation on the aggregated data. It can also incorporate hierarchical structure for estimating outcome mechanism by adding a single pooled individual-level regression in the Super Learner library. The second approach, on the other hand, runs pooled individual-level regressions on both outcome and treatment mechanisms, because it utilizes the pairing of individual-level covariates and outcomes. Note that all approaches use the equal-mass method for choosing bins (for the continuous exposure), and `speedglm` as the estimators for both outcome and exposure mechanisms. Parameter estimates are obtained from 200 repetitions of the simulation.

```
niterations <- 200  # Number of repetitions
J <- 1000
n <- 50
res.wmF.Ia <- res.wmF.Ib <- res.wmF.II <-
  as.data.frame(matrix(NA, nrow = niterations, ncol = 9))
names(res.wmF.Ia) <- names(res.wmF.Ib) <- names(res.wmF.II) <-
  c("TMLE.est", "IPTW.est", "Gcomp.est", "TMLE.var", "IPTW.var",
    "Gcomp.var", "TMLE.cover", "IPTW.cover", "Gcomp.cover")

for (i in 1:niterations) {
  # Generate the full hierarchical data
  data <- get.fullDat.Acont(J = J, n.ind = n, truncBD = 5,
                            shift = 1, working.model = F)
  tmleCom_Options(maxNperBin = NROW(data), nbins = 5)

  # Check if the true value falls into the confidence interval
  getCover <- function(CI, truth) {
    return(as.integer(CI[, 1] <= truth & truth <= CI[, 2]))
```

```
  }

  # Community-level analysis without a pooled regression on outcome
  tmle_comQg <- tmleCommunity(
    data = data, communityID = "id", Ynode = "Y", Anodes = "A",
    WEnodes = c("E1", "E2", "W1", "W2", "W3"), f_gstar1 = f.gstar,
    community.step = "community_level", pooled.Q = FALSE,
    obs.wts = "equal.within.community", rndseed = 1)
  res.wmF.Ia[i, 1:6]<-
    unlist(sapply(tmle_comQg$EY_gstar1[1:2], as.vector))
  res.wmF.Ia[i, 7:9] <- getCover(tmle_comQg$EY_gstar1$CIs, truth.wmF)

  # Community-level analysis with a pooled regression on outcome
  tmle_cQ.pg <- tmleCommunity(
    data = data, communityID = "id", Ynode = "Y", Anodes = "A",
    WEnodes = c("E1", "E2", "W1", "W2", "W3"), f_gstar1 = f.gstar,
    community.step = "community_level", pooled.Q = TRUE,
    obs.wts = "equal.within.community", rndseed = 1)
  res.wmF.Ib[i, 1:6] <-
    unlist(sapply(tmle_cQ.pg$EY_gstar1[1:2], as.vector))
  res.wmF.Ib[i, 7:9] <- getCover(tmle_cQ.pg$EY_gstar1$CIs, truth.wmF)

  # Individual-level analysis
  tmle_poolQg <- tmleCommunity(
    data = data, communityID = "id", Ynode = "Y", Anodes = "A",
    WEnodes = c("E1", "E2", "W1", "W2", "W3"), f_gstar1 = f.gstar,
    community.step = "individual_level", rndseed = 1)
  res.wmF.II[i, 1:6] <-
    unlist(sapply(tmle_poolQg$EY_gstar1[1:2], as.vector))
  res.wmF.II[i, 7:9] <- getCover(tmle_poolQg$EY_gstar1$CIs, truth.wmF)
}
```

Results displayed in Table 3.1 shows the comparison of the performance of the two TMLEs when the assumption of "no covariate interference" holds and the assumption fails badly. As predicted by theory, the community-level targeted estimator (TMLE-Ia), which is always based on the aggregated data, has a good performance in both situations with negligible bias. However, the coverage rates of its influence-curve-based confidence intervals are lower than nominal (89.5% and 86.5%) due to small variances. In this case, TMLE-Ib, which uses a pooled individual-level outcome regression and then a community-level stochastic intervention, performs slightly worse than TMLE-Ia. When the working model holds, we observe that the coverage rate of TMLE-Ib (68.5%) has a notable decrease compared to that of the other estimators because of a relatively large bias. As expected, the individual-level

Table 3.1: Simulation study 1. Simulation-based performance of TMLE, IPTW, Gcomp estimators with stochastic exposures over 200 repetitions of the simulation, when the working model holds ($\psi_0 = 55.57\%$) and when the working model is not a reasonable approximation ($\psi_0 = 55.78\%$). TMLE-Ia indicates both the outcome regression and the treatment mechanism are adjusted at the community-level. TMLE-Ib uses the individual-level outcome regression and the community-level treatment mechanism. TMLE-II indicates both are adjusted at the individual-level. IPTW-I and Gcomp-I indicate the use of community-level treatment and community-level outcome, respectively. IPTW-II and Gcomp-II indicate the use of individual-level treatment and individual-level outcome, respectively. For each estimator, the columns denote $\hat{\psi}$ as the average point estimate, "Bias" as the absolute difference between the estimate $\hat{\psi}$ and the truth ψ, $\hat{\sigma}$ as the average standard error estimate, rMSE as the root mean squared error, and "Cover" as the proportion of times that the truth falls within the 95% CI. All outcome and treatment mechanisms are correctly specified. All reported bias, SE, rMSE and Coverage are multiplied by 100.

	Working Model Holds					**Working Model Fails**				
Estimator	$\hat{\psi}$	Bias	$\hat{\sigma}$	rMSE	Cover	$\hat{\psi}$	Bias	$\hat{\sigma}$	rMSE	Cover
TMLE-Ia	55.75	0.18	0.60	0.62	89.5	56.47	0.69	1.36	1.52	86.5
TMLE-Ib	56.48	0.91	0.64	1.12	68.5	56.50	0.72	1.60	1.75	89.5
TMLE-II	55.71	0.13	0.39	0.41	84.0	57.59	1.81	1.48	2.34	69.0
IPTW-I	56.63	1.06	2.60	2.81	100	56.16	0.38	2.91	2.94	100
IPTW-II	55.67	0.10	3.44	3.44	100	57.23	1.45	4.23	4.47	100
Gcomp-I	55.83	0.26	0.60	0.65	90.5	56.44	0.67	1.36	1.51	85.5
Gcomp-II	55.75	0.18	0.39	0.43	87.5	57.57	1.79	1.48	2.33	71.0

targeted estimator (TMLE-II) is biased and its confidence interval coverage has an obvious decrease when the working model fails. Even though the working model is not a reasonable approximation, surprisingly, the IPTW using individual-level stochastic intervention (IPTW-II) provides a reasonable estimate.

Simulation 2 - Static interventions

We now consider another common simulation study with binary community-level exposure(s), which is commonly used in the study of HIV prevention and treatment. Similar to the previous simulation, we generate 200 samples of size $J = 100$ communities, each con-

taining nj observation where $n_j \sim N(50, 10)$. The data generating mechanism is as follows.

$$W_{1,n_j} \sim (Bern(0.6))_{i=1,\ldots,n_j} \qquad W_{2,n_j} \sim (N(0,1))_{i=1,\ldots,n_j}$$
$$W^c_{1,n_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} \boldsymbol{W}_{1,n_j} \qquad W^c_{2,n_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} \boldsymbol{W}_{2,n_j}$$
$$A_j \sim Bern(expit(W^c_{1,n_j} + 0.56W^c_{2,n_j}))$$

However, the mechanism differs in the outcome distribution:

- **Case 1**: Working model is a reasonable approximation

$$Y_j \sim Bern(expit(0.15 + 0.3A_j + 0.1\boldsymbol{W}^c_{1,n_j} + 2\boldsymbol{W}_{1,n_j} + 0.9\boldsymbol{W}_{2,n_j}))$$

- **Case 2**: Working model is not a reasonable approximation

$$Y_j \sim Bern(expit(0.15 + 0.3A_j + 3\boldsymbol{W}^c_{1,n_j} - 0.9\boldsymbol{W}^c_{2,n_j} - 0.3\boldsymbol{W}_{1,n_j} + \boldsymbol{W}_{2,n_j}))$$

Table 3.2: Simulation study 2. Performance of TMLE, IPTW, Gcomp estimators with binary exposures over 200 repetitions of the simulation, when the working model approximately holds ($\psi_0 = 4.16\%$) and when the working model does not hold ($\psi_0 = 3.71\%$). All outcome and treatment mechanisms are correctly specified. All reported bias, SE, rMSE and Coverage are multiplied by 100.

	Working Model Holds				**Working Model Fails**			
Estimator	Bias	$\hat{\sigma}$	rMSE	Cover	Bias	$\hat{\sigma}$	rMSE	Cover
TMLE-Ia	0.03	1.15	1.16	95.0	0.03	1.07	1.08	92.5
TMLE-Ib	0.16	1.16	1.17	95.0	0.25	1.24	1.26	97.5
TMLE-II	0.01	1.14	1.14	95.00	0.04	1.22	1.22	96.0
IPTW-I	0.02	3.79	3.79	100	0.06	3.46	3.46	100
IPTW-II	0.04	15.99	15.99	100	0.02	17.56	17.56	100
Gcomp-I	0.03	1.15	1.16	95.5	0.03	1.07	1.08	91.5
Gcomp-II	0.01	1.14	1.14	95.0	0.04	1.22	1.22	96.0

As before, table 3.2 summarizes the performance of the estimators under different outcome generating distributions. First, TMLE-Ia performs well in both situations with negligible biases and great confidence interval coverages. As expected, TMLE-Ib performs similarly

to TMLE-Ia when the working model holds, and worse than TMLE-Ia when it fails, in terms of bias and variance. TMLE-II, on the other hand, performs very well when the working model provides a reasonable approximation, but exhibits slight increases in bias and variance (and so a more conservative confidence interval) when the working model fails. Theoratically, TMLE-II uses lower dimensional objects with size $N = \sum_{j=1}^{J} N_j$ and so may improve the finite sample efficiency if the working model holds. However, when the working model does not hold, the misspecification of both the outcome and treatment regressions will cause biased estimate and efficiency loss. Besides, both IPTW-I (with the community-level g) and IPTW- II (with the individual-level g) have larger variances compared to other estimators, and so provides 100% coverage rates. It could be explained that the IPTW estimator has relatively large variability, despite the large sample size. In other words, the range of the estimated values of IPTW can be wide and results in a large variance.

Simulation 3 - Stochastic interventions (N = 1)

In this simulation, we study the special case where each community has only one observation (i.e., $N = 1$) and the intervention is stochastic. As described in section 2.3, it's similar to data with only community-level baseline covariates (i.e., treat $(E, W) = E$). The data-generating distribution is described as follows:

$$E_1 \sim Bern(0.5) \qquad E_2 \sim Bern(0.3)$$
$$E_3 \sim N(0, 0.25) \qquad E_4 \sim Unif(0, 1)$$
$$A|E_1, E_2, E_3, E_4 \sim N(0.86E_1 + 0.41E_2 - 0.34E_3 + 0.93E_4, 1)$$
$$Y|A, E_1, E_2, E_3, E_4 \sim N(3.63 + 0.11A - 0.52E_1 - 0.36E_2 + 0.12E_3 - 0.13E_4, 1)$$

Given a shift value and a truncation bound, the intervened exposure A^* is distributed as:

$$A_j^* = \begin{cases} A_j + shift & exp\{0.5 * shift * (A_j - \mu(E_1, E_2, E_3, E_4) - \frac{shift}{2}\} > truncbd \\ A_j, & o.w. \end{cases}$$
$$\text{where} \quad \mu(E_1, E_2, E_3, E_4) = 0.86E_1 + 0.41E_2 - 0.34E_3 + 0.93E_4$$

Given a shift of 2 and a truncation bound of 10, the marginal treatment effect of the individual-based intervention is $\psi_0 = 3.505$. In the next step, we will explore the estimation performance of the targeted estimators with different choices of binarization methods and the number of bins. Also, we are interested in the performance of the estimators under different model specifications, including correctly specified and misspecified models for the outcome regression and the density of the conditional treatment distribution. Again, code to generate the example dataset is attached in the supplementary material.

In figure 3.1, the outcome $\bar{Q}_0(A, E)$ is estimated with the correctly specified main terms regression model and a misspecified regression model, only adjusting for A and E_3. Besides, the stochastic exposure $g^*(a|E)$ is estimated with a correctly specified model, as well as a $g(a|E)$ misspecified model, only adjusting for E_3. The simulations results are consistent with

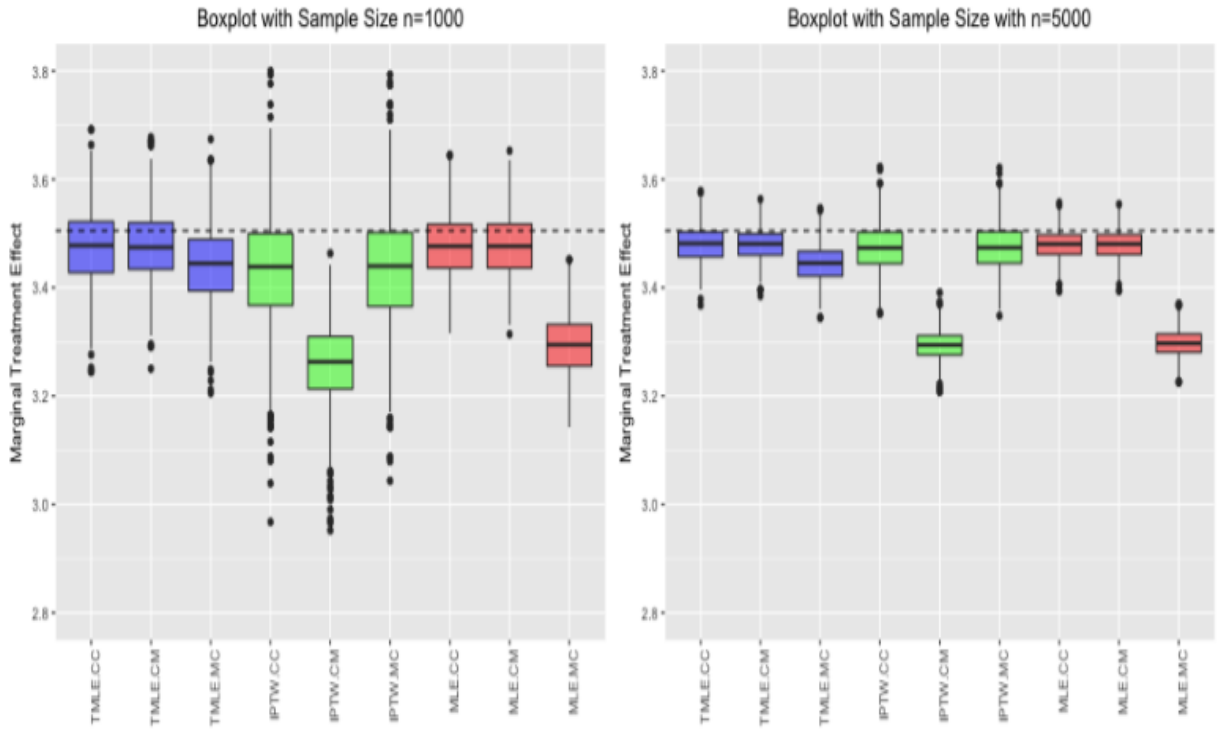


Figure 3.1: Box plots of the point estimates from three algorithms for sample sizes $n = 1000$ (left) and $n = 5000$ (right) in Simulation study 3. The x-axis denotes the combination of the estimator and the model specification. CC indicates correctly specified outcome regression and exposure mechanism. CM indicates the outcome regression is correctly specified, while the exposure mechanism misspecified. MC indicates the exposure mechanism is correctly specified, while the outcome regression misspecified. The dashed line indicates the true value $\psi_0 = 3.505$.

the theoretical predictions. TMLE performs quite well if either the outcome regression or the exposure mechanism is correctly specified. IPTW exhibits low bias when $g^*(a|E)$ is correctly specified, but is biased otherwise. This bias decreases but does not disappear with an increase in sample size. Besides, IPTW has much higher variance than other estimators even with a correctly specified $g^*(a|E)$, which may be explained by practical positivity violations such as small $g^*(a|E)$ causes large weights on few individuals. Weight truncation could be a possible solution for this practical violation - We can implement more restrictive bounds on g^* (in the simulation analysis, a less restrictive set of bounds of $[0.005, 1]$ is used). When the model for $\bar{Q}_0(A, E)$ is misspecified, MLE performs poorly in precision, but MLE is unbiased when $\bar{Q}_0(A, E)$ is correctly specified. Furthermore, sample size does help reduce variance. Last but not least, it is worth mentioning that the all the estimation are not fully unbiased even with correctly specified mechanisms. One possible explanation is that that $\frac{\hat{g}^*}{\hat{g}}$ is not bounded well, which leads to the violation of positivity assumption. Therefore, what we can do is to

always restrict $\frac{\hat{g}^*}{\hat{g}} \leq 20$.

3.4 Discussion

The **tmleCommunity** package was developed to offer a flexible, easily customizable implementation of the TMLE algorithm for hierarchical data structure, along with community-level multivariate arbitrary interventions. The main class of causal parameters that is estimated by the package is the treatment specific mean effect, which can be easily extended to ATE. A neophyte only needs to supply the data and specify the data arguments `Ynode`, `Anodes`, `WEnodes` and `f_gstar1`. On that basis, experienced users can control the estimation procedure by providing the user-supplied regression models for $\bar{Q}_0$, g_0 and g^*, and choosing preferred methods allowed for arguments, such as the method dealing with hierarchical data, whether including hierarchical structure to estimate $\bar{Q}_0$, either linear or logistic fluctuation for targeting, and the TMLE targeting step method. Remarkably, `obs.wts` and `community.wts` can be used to correct for case-control sampling (when the outcome is rare). Besides, the `tmleCommunity` function can either internally estimate all factors of the likelihood, or use the values for g_n and g_n^* from the external estimation procedure through `h.g0_GenericModel` and `h.gstar_GenericModel`. The choices of data-adaptive machine learning techniques and other more advanced estimation methods can be specified in the `tmleCom_Options` function.

Planned extensions to the package include several areas. First, we plan to include TMLE estimation of casual effects of multiple time-point interventions, adjusting for time-dependent covariates for hierarchical longitudinal data. Second, since considering only complete cases in the data is inefficient and may cause bias when missingness is informative. The package will then be extended to allow missingness on the outcome vector, so that the corresponding covariate information can be utilized for reducing bias and increasing efficiency in estimates. Third, as mentioned in section (2.3), when one individual's outcome is affected by the individual-level covariates of the subset of other individuals from the same community, the strength of the "no covariate interference" assumption should be weakened by including this knowledge of dependence. Another planned addition to this package will so allow estimation of community-level TMLE under this setting.

Additionally, this package estimates variances and standard errors through estimated influence curves. Double robustness makes these estimates asymptotically correct if both the outcome and treatment mechanisms are estimated consistently at reasonable rates, and conservative if only one of them is estimated consistently. However, variance estimation is difficult when violations or near violations of positivity happen in finite samples due to chance [57]. This is usually a problem in small samples or when the exposure is continuous, since discretization of the support of the exposure could lead to lack of data in some bins. This sparsity results in poor finite sample performance, particularly for estimations of variances and confidence interval coverages, and even threatens valid inference. One alternative method for variance estimation is the non-parametric bootstrap, especially when

central limit theorem may not apply due to sparsity. Thus, we plan to include the alternative variance estimates in the future.

In this package, we use known stochastic interventions such as a shifted version of the current exposure mechanism g_0 given a known shift function. In practice, a stochastic intervention g^* could also be unknown (i.e., not a function of g_0 anymore). If we consider the estimation of an optimal treatment rule where the rule is defined to maximize the mean outcome under the treatment, without cross validation, we will use the same information from the observed data to estimate both the user-specified mechanism g^* and the mean outcome under the fitted mechanism, which may result in finite sample bias. According to [41] and [51], the cross-validated TMLE (cv-TMLE) approach avoids empirical process conditions and for each sample split, it estimates an empirical mean over a validation sample, under a stochastic (or deterministic) intervention estimated based on the training sample. Therefore it may reduce finite sample bias, and including cv-TMLE in the package can be one of our future work.

3.5 Answers to some frequently asked questions (FAQ)

Can I call the ***tmleCommunity*** *function a second time without having to re-do the estimation of exposure mechanism?*

Yes. Users can use command `result$EY_gstar1$h.g0_GenericModel` to obtain an object of `GenericModel` **R6** class containing the previously fitted models for $P(A|W,E)$ under observed mechanism g_0 (assuming the result of the first call to `tmleCommunity` is returned to the variable named `result`, and only one intervention function `f_gstar1` is user given). Similarly, an object `GenericModel` class containing the previously fitted models for $P(A|W,E)$ under intervention `f_gstar1` is returned as `result`EY_gstar1`h.gstar_GenericModel`. The two objects can be passed into the second call to `tmleCommunity` by specifying the values for `h.g0_GenericModel` and `h.gstar_GenericModel`, respectively. Assuming we are using the simulated data and the first estimation result in section 3.2, the next code chunk illustrates how this is done.

```
tmleCom_gc_default2 <- tmleCommunity(
  data = indSample.cA.cY, rndseed = 1, Ynode = "Y", Anodes = "A",
  WEnodes = c("W1", "W2", "W3", "W4"), Qform = "Y ~ W1 + W2 + A",
  h.g0_GenericModel = tmleCom_gc_default$EY_gstar1$h.g0_GenericModel,
  h.gstar_GenericModel = tmleCom_gc_default$EY_gstar1$h.gstar_GenericModel)
cbind(tmleCom_gc_default2$EY_gstar1$estimates,
      tmleCom_gc_default2$EY_gstar1$vars)
```

Can I define and fit the multivariate conditional density under the user-specified intervention function directly without having to call the ***tmleCommunity*** *function?*

Yes, the package provides an individual function named `fitGenericDensity` to define and fit regression models for the conditional density $\mathbb{P}(A = a|W = w)$ where a is generated

under a user-specified arbitrary (can be static, dynamic or stochastic) intervention function. Its arguments are similar to those for estimating treatment mechanisms in `tmleCommunity`, except hierarchical data structure is not supported in this function. Therefore, this function is purely for estimating the multivariate conditional density.

With the same data set simulated in section 3.2, we may be interested in the mean counterfactual outcome under a stochastic intervention g^* where the observed A is shifted to the left by the half of its mean.

```
define_f.gstar <- function(shift.rate, ...) {
  eval(shift.rate)
  f.gstar <- function(data, ...) {
    print(paste0("rate of shift: ", shift.rate))
    shifted.new.A <- data[, "A"] - mean(data[, "A"]) * shift.rate
    return(shifted.new.A)
  }
  return(f.gstar)
}
f.gstar <- define_f.gstar(shift.rate = 0.5)

tmleCom_Options(maxNperBin = N, bin.method = "dhist", nbins = 8)

# Under current treatment mechanism g0
fit_gN <- fitGenericDensity(data = indSample.cA.cY, Anodes = "A",
                            Wnodes = c("W1", "W2", "W3", "W4"),
                            f_gstar = NULL, gform = gform.C)
# Under stochastic intervention gstar
fit_gstar <- fitGenericDensity(data = indSample.cA.cY, Anodes = "A",
                               Wnodes = c("W1", "W2", "W3", "W4"),
                               f_gstar = f.gstar, gform = gform.C)
```

Are there any sample data provided in the package so that users can play analysis on them?

Yes, the package comes with four sample datasets. `comSample.wmT.bA.bY_list` is an example of a hierarchical data containing a community-level binary exposure with a Individual-Level binary outcome. And `indSample.iid.cA.cY_list` is an example of a non-hierarchical data containing a continuous ex- posure with a continuous outcome. One non-hierarchical sample dataset is `indSample.iid.bA.bY.rareJ1_list`, which contains a binary exposure with a rare outcome (i.e., independent case-control scenario where $J = 1$). Beside, the data structure of another dataset `indSample.iid.bA.bY.rareJ2_list` is identical to this of `indSample.iid.bA.bY.rareJ1_list`, except that now the ratio of the number of controls to the number of case J is 2.

Can the **tmleCommunity** *package handle panel data transformation before performing TMLE analysis?*

Yes. The `panelData_Trans` function provides a wide variety of ways of data transformation for panel datasets, such as fixed effect and pooling model. It also allows users to only apply transformation on regressors of interests, instead of on the entire dataset. For example, before running the `tmleCommunity` function on the data set simulated in section 3.3 when the working model fails, we want to apply fixed effect transformation where the individual effect is introduced, then we can use

```
pData.FE <- panelData_Trans(
   data = comSample.wmF, xvar = c("E1", "E2", "W1", "W2", "W3", "A"),
   yvar = "Y", index = "id", effect = "individual",
   model = "within", transY = TRUE)
```

Besides, we can keep the outcome variable fixed during the panel transformation by setting `transY = False`. Additional details can be found in the package manual `https://github.com/chizhangucb/tmleCommunity/blob/master/tmleCommunity.pdf`.

3.6 Acknowledgments

This work was supported by National Institutes of Health Director's New Innovator Award Program DP2HD080350 (PI: Jennifer Ahern).

Chapter 4

Prediction of diagnoses of nonalcoholic steatohepatitis in a large administrative claims database using ensemble learning

4.1 Introduction

Motivation

In contemporary clinical and public health research, administrative claims databases have been widely used due to their up-to-date, broad coverage of a large enrolled population, low cost relative to primary data collection, and ability to capture longitudinal data. These datasets capture medical diagnoses, dispensed prescriptions, and procedures submitted for reimbursement using standardized sets of codes. On October 1, 2015, the US transitioned to the most recent version of the International Classification of Diseases, version 10 Clinical Modification (ICD-10-CM) for diagnostic claim coding. In ICD-10-CM, non-alcoholic steatohepatitis (NASH) was assigned an explicit diagnostic code, however in the ICD-9-CM coding era, it was grouped with other chronic non-alcoholic liver disease conditions. The development of an algorithm to predict ICD-10-CM NASH from claims available in the ICD-9-CM coding era, including not only diagnostic codes but also codes for medications, would permit much longer longitudinal follow-up time for patients with NASH, thereby permitting improve understanding of the long-term natural history of this condition.

Despite the existence of more advanced approaches, health-related research relies heavily on parametric models like logistic regression. Because the functional form of the relationship between predictors and outcome is commonly unknown, parametric models that impose strong assumptions are likely misspecified in practice. In addition, the large number of unique codes that could serve as potential predictors in an administrative claims database make it impossible for researchers to correctly specify the parametric statistical model. On the other hand, data-adaptive approaches such as random forest and kernel-based methods like support vector machines offer flexible alternatives by making fewer structural assumptions

on the functional form, particularly when complicated interactions among predictors exist [29]. Since no single algorithm will always be optimal in every data-generating scenario, an alternative, with the goal of improving prediction performance, is to combine the predictions of several learning algorithms, a process known as "stacking" [84]. Super Learner (SL) [42] is a general loss-based ensemble learning method that creates the best weighted combination of candidate algorithms from a user-specified library, with a goal of minimizing a V-fold cross-validated empirical risk associated with a loss function specified by the user. Theoretical properties [43, 13, 74], especially the oracle inequality for cross validation (CV) selectors, guarantee that SL performs asymptotically at least as well as the best convex combination of candidate algorithms in the library with respect to the loss-based dissmilarity, and SL converges at the parametric rate of $log(n)/n$ if one of the algorithms achieves the parametric rate of convergence to the truth. Furthermore, depending on the goal of the analyses, different loss functions can be used in SL. For example, if the goal is to maximize the Area under the ROC curve (AUC) in a binary classification problem, then a SL with an AUC-maximizing metalearner may have significant advantages with respect to AUC performance, compared to those with other metalearners that do not target AUC [47].

An important task in all high-dimensional mathematical problems is feature selection. According to Fan and Fan [15], classification using all features in high-dimensional feature space can be as bad as random guessing because of noise accumulation in estimating population centroids. Traditional variable selection methods including best subset regressions and step-wise regression suffer from several drawbacks, one of the most severe of which is the very expensive computational cost in high-dimensional data settings. Recently, a group of machine learning methods including least absolute shrinkage and selection operator (LASSO) has shown good performance on estimation accuracy by effectively identifying the subset of important features, and helping to reduce the computational burden significantly when the solution is sufficiently sparse [16, 17]. In SL, screening can be either supervised or unsupervised because it's part of the algorithm and thus can be included when calculating the cross-validated risk of an algorithm in a user-specified library [58]. Also, screening algorithms can be paired with prediction models to create new candidate algorithms in the library, such as two logistic regressions that regress the dependent variable on all features and on the subset of only demographic variables [80]. Moreover, in many practical settings, there exists a tension between the richness of a search over the solution space and the feasibility of such a search in massive, high-dimensional datasets. A complex prediction algorithm can be computationally intractable when the dataset is too large. Additionally, the process of V-fold CV in SL makes the computation problem more severe [48]. In order to resolve this tension, Gruber *et al* [25] proposed using a concise SL library that can be computationally-reasonable, and more importantly, using SL built on leave-one-group-out CV (LOGOCV) instead of V-fold CV. In practice, SL built on LOGOCV significantly reduces the computation burden in large datasets without sacrificing prediction strength [25, 36].

The main goal of this chapter was to construct a SL model for the prediction of ICD-10 NASH diagnoses in a high-dimensional administrative claims database. To that end, we

evaluated the performance of SL for predictive modeling using different individual prediction algorithms and user-specified screening methods. We also explored the performance trade-offs in SL by specifying a larger library that contained 19 diverse individual learners, and a smaller library that contained 5 relatively fast individual learners.

Organization of this chapter

The rest of this chapter is organized as follows. In Section 4.2, we first describe how the study cohort was created and what candidate predictors were included in the dataset. Next, we formulate the AUC-maximizing Super Learner algorithm built on LOGOCV for a binary classification problem. Then, we provide details of the individual machine learning algorithms that were included in the SL library, and propose Super Learner classifiers that consider different sets of individual prediction algorithms mentioned previously. Finally, we review the use of different performance measures in binary classification problems. In Section 4.3, 4.4 and 4.5 we apply the proposed Super Learner classifiers, along with the individual machine learning algorithms to predict NASH diagnoses in the created datasets. The article concludes with a discussion in Section 4.6.

4.2 Methods

Dataset, cohort definition and candidate predictors

We used the IQVIA/PharMetrics Plus^{TM} data set, a fully adjudicated US administrative claims database with medical and pharmacy claims for approximately 140 million patient lives available from January 1, 2006 through December 30, 2017. This raw data was transformed into tables of enrollment and demographic information, as well as tables for all claimed conditions (using ICD-9-CM and ICD-10-CM codes), prescriptions (using generic product identifiers; GPIs), and procedures (using current procedural terminology (CPT), healthcare common procedure coding system (HCPCS), and ICD Procedure codes).

For this analysis, we identified a cohort of patients with any NASH claim (K75.81) in the first year of ICD-10-CM coding (October 1, 2015 – October 1, 2016) and enrollment on or before October 1, 2013 to capture the final two years of claims in the ICD-9-CM era. For comparison, we identified patients enrolled through at least the first year of the ICD-10-CM era with no NASH claims, and selected a random 1% sample of those enrolled on or before October 1, 2013. After excluding patients under 18 years or with no recorded sex, 212,021 patients remained. Only the two years of claims from Oct 1st, 2013 to Sept 30th, 2015 were considered for the NASH prediction algorithms. There were 13,215 unique claims codes occuring during this time period (11,354 diagnostic codes and 1,861 medication dispensing codes).

Through literature review and consultation with medical experts, we identified 11 comorbid conditions (10 medical conditions and smoking) associated with a NASH diagnosis.

For each of these 11 conditions (represented by 133 individual diagnostic and 78 medical dispensing claim codes, in total 211), a 1/0 flag was created in the dataset. If a patient had a claim for any of the 211 codes during the 2-year observation period, the associated condition flag was coded as '1', the absence of any representative claim codes for a particular condition resulted in a '0' for that flag. We referred to these 11 comorbidities as "researcher-specified claims variables" since they reflected expert knowledge on NASH diagnoses. After excluding patients with no claims for any of the 11 researcher-specified conditions, 96,639 patients remained (16,966 NASH and 79,673 non-NASH patients). For all other diagnostic, or medication dispensing claims occurring during the 2 year observation period that were not already captured in the 211 codes associated with the researcher-specified conditions, the earliest claim date was retained and converted to a 1/0 flag for each patient.

In order to have a better understanding of the role of the researcher-specified claims variables, we constructed two datasets: one that considered all claims codes individually with no grouped researcher-specified claims variables. and another considered the 11 grouped researcher-specified variables and the remaining individual claims codes. In summary, the two datasets had the following structure:

- *NASH_nogroup*: The dataset included patient ID, 2 demographic variables (age and sex), and 1/0 flags for each of the 13,215 unique claims codes (11,354 diagnostic and 1,861 medication dispensing) occurring during the 2 year observation window.
- *NASH_pregroup*: The dataset included patient ID, 2 demographic variables (age and sex), 1/0 flags for each of the 11 researcher-specified claims variables (NAFLD, insulin resistance, obesity, hyperlipidemia, type-II diabetes, hypertension, hypertriglyceridemia, stroke, myocardial infarction, coronary atherosclerosis and smoking), and 1/0 flags for each of the 13,004 remaining unique claims codes (11,221 diagnostic and 1,783 medication dispensing) occurring during the 2 year observation window.

Note that both researcher-specified claims variables and demographic variables were considered as baseline variables.

Formulation of AUC-maximizing Super Learner built on LOGOCV

Consider the observed data structure $O = (Y, W) \sim P_0 \in \mathcal{M}$, where $\mathcal{M}$ is a model space that includes all possible probability distributions for O and P_0 is the true unknown distribution in $\mathcal{M}$. Here, $Y \in \{0, 1\}$ is a binary class of interest with 1 as the positive class and 0 as the negative class, and W is a p-dimensional set of covariates. Let $\Psi : \mathcal{M} \to \mathbb{R}$ denote a mapping from any distribution $P \in \mathcal{M}$ to a score function $\psi = \Psi(P)$ that maps W into $(0, 1)$. Then the true target parameter $\psi_0 = \Psi(P_0)$ is given by the mapping appied to the true P_0, and the objective is to estimate the conditional expectation $\psi_0(W) = \mathbb{E}(Y|W)$.

Recall that the Area Under the ROC curve (AUC) can be defined as $AUC(P_0, \psi) = \int_0^1 P_0(\psi(W) > \mu|Y = 1)P_0(\psi(W) = \mu|Y = 0)d\mu = P_0(\psi(W_1) > \psi(W_2)|Y_1 = 1, Y_2 = 0)$, where both (Y_1, W_1) and (Y_2, W_2) are sampled from P_0 [48]. Suppose there were n

independent and identically distributed (i.i.d.) copies of $O_i = (Y_i, W_i)$ sampled from P_0, with n_1 positive cases and n_0 negative cases. We used P_n to denote the empirical distribution of the n i.i.d. observation, $P_n^{\mathcal{T}}$ the empirical distribution of the training set ($\mathcal{T}$), and $P_n^{\mathcal{V}}$ the empirical distribution of the validation set ($\mathcal{V}$). Therefore the AUC of the empirical distribution can be calculated as [27]:

$$\begin{aligned} AUC(P_n, \psi) &= \frac{1}{n_1 n_0} \sum_{i=1}^{n_1} \sum_{j=1}^{n_0} \mathbf{1}_{\psi(W_i) > \psi(W_j)} \mathbf{1}_{Y_i=1, Y_j=0} \\ &= \frac{1}{n_1 n_0} \sum_{i=1}^{n_1} \sum_{j=1}^{n_0} \mathbf{1}_{\psi(W_i) > \psi(W_j)} \end{aligned} \tag{4.1}$$

Let $\mathcal{L} = \{\Psi_1, ..., \Psi_J\}$ denote a SL library of J alogrithms, where each alogrithm can be either a pre-specified parametric model or a non-parametric machine learning approach, with a corresponding function $\psi(W) = \Psi(P)(W)$. The core part of the AUC-maximizing Super Learner algorithm built on LOGOCV is outlined in the following steps:

(1) Split the sample of n observations with n_1 positive and n_0 negative cases into a training set with np observations, and a validation set with $n(1-p)$ observations, where $p \in (0, 1)$ is a user-specified split ratio (e.g., a common choice of p is 0.8).

(2) Fit each algorithm in $\mathcal{L}$ on the training set and obtain the predictions on the validation set. Stack the predictions from each algorithm to yield a $n(1-p) \times J$ level-one design matrix:

$$Z = \begin{bmatrix} \hat{\Psi}_1(P_n^{\mathcal{T}})(W_1^{\mathcal{V}}) & \hat{\Psi}_2(P_n^{\mathcal{T}})(W_1^{\mathcal{V}}) & \dots & \hat{\Psi}_J(P_n^{\mathcal{T}})(W_1^{\mathcal{V}}) \\ \hat{\Psi}_1(P_n^{\mathcal{T}})(W_2^{\mathcal{V}}) & \hat{\Psi}_2(P_n^{\mathcal{T}})(W_2^{\mathcal{V}}) & \dots & \hat{\Psi}_J(P_n^{\mathcal{T}})(W_2^{\mathcal{V}}) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\Psi}_1(P_n^{\mathcal{T}})(W_{n(1-p)}^{\mathcal{V}}) & \hat{\Psi}_2(P_n^{\mathcal{T}})(W_{n(1-p)}^{\mathcal{V}}) & \dots & \hat{\Psi}_J(P_n^{\mathcal{T}})(W_{n(1-p)}^{\mathcal{V}}) \end{bmatrix}_{n(1-p) \times J}$$

(3) Propose a family of weighted combinations of J algorithms indexed by a weight vector α:

$$\hat{\Psi}_\alpha(P_n^{\mathcal{T}})(W^{\mathcal{V}}) = \hat{\psi}_\alpha(W^{\mathcal{V}}) = \sum_{j=1}^{J} \alpha_j \hat{\Psi}(P_n^{\mathcal{T}})(W^{\mathcal{V}}) \text{ s.t. } \sum_{j=1}^{J} \alpha_j = 1 \text{ and } \alpha_j \geq 0, \forall j,$$

where Ψ_α is an ensemble algorithm that combines the J algorithms through a linear combination given by α.

(4) Determine the α that minimizes the cross validated risk associated with some bounded loss function of interest, which is rank loss here (i.e., 1 - AUC), over all allowed α-

combinations. Based on the empirical AUC function (4.1), calculate $\hat{\alpha}$ as

$$\hat{\alpha} = \arg\min_{\alpha} \Big(1 - \frac{1}{n_1(1-p) \times n_0(1-p)} \sum_{i=1}^{n_1(1-p)} \sum_{j=1}^{n_0(1-p)} \mathbf{1}_{\hat{\psi}_\alpha(W_i^{\mathcal{V}}) > \hat{\psi}_\alpha(W_j^{\mathcal{V}})}\Big),$$

where subject i is from the positive class and subject j is from the negative class.

(5) Obtain the predictions for all n observations by re-fitting each algorithm that has a non-zero weight from the above step. Stack the predictions to yield a $n \times K$ matrix:

$$Z' = \begin{bmatrix} \hat{\Psi}_1(P_n)(W_1) & \hat{\Psi}_2(P_n)(W_1) & \dots & \hat{\Psi}_K(P_n)(W_1) \\ \hat{\Psi}_1(P_n)(W_2) & \hat{\Psi}_2(P_n)(W_2) & \dots & \hat{\Psi}_K(P_n)(W_2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\Psi}_1(P_n)(W_n) & \hat{\Psi}_2(P_n)(W_n) & \dots & \hat{\Psi}_K(P_n)(W_n) \end{bmatrix}_{n \times K},$$

where $K = \sum_{j=1}^{J} \mathbf{1}_{\hat{\alpha}_j > 0}$ represents the number of algorithms with non-zero coefficients.

(6) Combine $\hat{\alpha}$ with Z' to generate the Super Learner:

$$\hat{\Psi}_{\hat{\alpha}}(P_n) = \sum_{k=1}^{K} \hat{\alpha}_k z'_k,$$

where $k \in \{j : \hat{\alpha}_j > 0\}$ and z'_k represents the k^{th} column of the final prediction matrix Z'.

R code for implementing a AUC-maximizing Super Learner build on LOGOCV is provided in Appendix C.

Machine learning prediction algorithms

Due to computational constraints on large datasets, we considered a diverse set of 19 machine learning prediction algorithms that was computationally less intensive in our analyses, such as main term logistic regression, elastic-net regularized logistic regression [85], boosted classification and regression trees (CARTs)[6], and generalized boosted model[19]. Each of the prediction algorithms, also known as base learners, can be implemented by the **caret** package [40, 38] in R v.3.4.4 [61]. One common aspect of machine learning methods, especially those aimed at high-dimensional covariates, is he need to perform hyper-parameter optimization (i.e., tuning parameter selection). It can be very time- and memory-intensive to find an optimal hyper-parameter setting due to the complexity of the model and the large volume of data available for tuning parameters. Among the 19 base learners, 15 of them had at least one hyper-parameter that needed to be tuned. Considering the trade-off between

hyper-parameter optimization and time consumption, we used a grid search procedure with 5-fold cross validation, allowing 3 different values to test for each tuning parameter, to determine the optimal hyper-parameter setting for each of the 15 need-to-be-tuned learners. Table A1, which provides details for each of the 19 prediction algorithms, including the corresponding R package with its tuning parameter(s), can be found in Appendix A. R code for generating the base learners is also available in Appendix A.

Bayesian risk ratio algorithm

One well-known feature of health care claims databases is high-dimensionality given the massive number of diagnostic, medication dispensing, and procedure codes (in this study $p = 13,215$). Therefore, dimension reduction and variable selection are a necessity before running any prediction algorithms. Moreover, claims databases are typically sparse at the level of individual claim codes, limited to a handful of diagnoses, dispensing, and procedure codes for each patient. To address these issues, we proposed a novel variable selection method, namely Bayesian risk ratio (RR) selection, to select important claims codes based on the univariate association between individual code and the outcome. The Bayesian risk ratio can be understood as the weighed average of the raw proportion of events among the exposed with the prior mean of any event among the same group, divided by the same weighted average among the non-exposed.

As a key statistic for most case-control studies, the RR evaluates whether the risk of a certain event happening in one group is the same as of the same event happening in another group. In this study, the RR measures the risk that an outcome will occur given a particular claims code, compared to the risk of the outcome occurring in the absence of that code claim [71]. We outlined its core analytical part in the following steps:

(1) Specify data sources based on coding system, such as ICD-9 diagnoses and Generic Product Identifier Codes. It can be further specified if other sources can be identified. Note that steps (2) - (3) will be repeated for each of the data sources.

(2) For each code i, computes its number of instances of an event, being this a diagnosis, medication or procedure between the case and control groups, respectively, denoted by $m_{ca,i}$ and $m_{co,i}$. Compute the sum and the average of number of claims for all codes that appear within the case group, denoted by T_{ca} and $\bar{X}_{ca}$, where $T_{ca} = \sum_{i=1}^{n_{ca}} m_{ca}$ and $\bar{X}_{ca} = \frac{1}{n_{ca}} \sum_{i=1}^{n_{ca}} m_{ca}$, with $n_{ca} = \#\{i : m_{ca,i} > 0\}$. Similary, compute the sum and the average of number of claims for all codes that appear within the control group, denoted by T_{co} and $\bar{X}_{co}$, where $T_{co} = \sum_{i=1}^{n_{co}} m_{co}$ and $\bar{X}_{co} = \frac{1}{n_{co}} \sum_{i=1}^{n_{co}} m_{co}$, with $n_{co} = \#\{i : m_{co,i} > 0\}$.

Then, calculate its Bayesian risk ratio (BRR) as

$$BRR_i = \frac{(m_{ca,i} + \bar{X}_{ca})/(T_{ca} + 0.5)}{(m_{co,i} + \bar{X}_{co})/(T_{co} + 0.5)},$$

where BRR_i is unknown if either $m_{ca,i}$ or $m_{co,i}$ is 0.

(3) Identify candidate codes by choosing three OR cutoff points $\{a, b, c\}$ (considered as tuning parameters, usually $b = \frac{1}{a}$) and selecting code i with ($BRR_i < a$ or $BRR_i > b$) and $m_{ca,i} > c$ and $m_{co,i} > c$. (e.g., keep all codes with ($BRR < 0.5$ or $BRR > 2$) and $m_{ca} > 25$ and $m_{co} > 25$).

R code for running the Bayesian risk ratio algorithm is provided in Appendix B.

Super Learner

Recall that the Super Learner prediction is the optimal combination of the predicted values from a user-specified library, where "optimal" is defined in terms of minimizing the V-fold cross-validated risk associated with a user-specified bounded loss function, usually the negative log-likelihood loss or the squared-error loss [80]. Thus, the selection of a diverse library and a loss function is crucial for SL prediction performance. Because our main performance metrics were AUC and computation time, a rank loss (i.e., 1 - AUC) function, along with the 19 computationally less intensive base learners described in Section 4.2 were used in the SL. Although SL built on V-fold CV is preferred to SL built on LOGOCV for small datasets, the LOGOCV approach performs asymptotically well in large datasets [25]. Considering the large amount of the data in this study, we chose SL built on LOGOCV. This version of SL can be performed in R by the function *SampleSplitSuperLearner()* using the **SuperLearner** package [59]. Table 4.1 shows the eight libraries that were made available to SL for our analyses.

Note that in SL analyses, the Bayesian RR method can be used as either a prediction algorithm or a screening algorithm: When considered as prediction algorithms, Bayesian RR prediction algorithms are actually the logistic regression models that utilize different subsets of the claims codes screened by Bayesian RR selection using different sets of tuning parameters; When considering as screening algorithms, any prediction algorithm in the SL library can be augmented with various Bayesian RR screening algorithms with user-specified cutoff points. In SL3 and SL4, nine Bayesian RR prediction algorithms with different sets of tuning parameters, including $a = \{0.22, 0.33, 0.45, 0.50, 0.53, 0.56, 0.59, 0.65, 0.68\}, b = \frac{1}{a}$ and $c = 25$, were contained in the libraries. making each library 28 prediction algorithms in total. In SL5 and SL6, the Bayesian RR method was considered as a screening algorithm, and nine fixed pairs of Bayesian RR tuning parameters were used to generate different subsets of claims codes that the 19 base learners would utilize. SL7 and SL8 were identical to SL5 and SL6, respectively, except that the Bayesian RR screening step was performed outside the SL (in this study, performed in a relational database software named Teradata).

Table 4.1: Details of the Super Learner libraries considered. Five fast learners includes GLM, Bayesian GLM, naive bayes, shrinkage discriminant analysis, penalized discriminant analysis.

(a) *Smaller Super Learner Libraries*

	Library	Covariates
SL1	5 fast ML base learners	Only baseline covariates
SL3	5 fast ML base learners, and 9 BRR prediction algorithms*	Baseline covariates and claims codes: only base learners utilize baseline covariates, and only BRR algorithms utilize claims codes.
SL5	5 fast ML base learners*	Baseline covariates, and claims codes screened by the BRR algorithm within the SL
SL7	5 fast ML base learners*	Baseline covariates, and claims codes screened by the BRR algorithm outside the SL

(b) *Larger Super Learner Libraries*

	Library	Covariates
SL2	All 19 ML base learners	Only baseline covariates
SL4	All 19 ML base learners, and 9 BRR prediction algorithms*	Baseline covariates and claims codes: only base learners utilize baseline covariates, and only BRR algorithms utilize claims codes.
SL6	All 19 ML base learners*	Baseline covariates, and claims codes screened by the BRR algorithm within the SL
SL8	All 19 ML base learners*	Baseline covariates, and claims codes screened by the BRR algorithm outside the SL

Note: * indicates that those prediction algorithms consist of screening-algorithm pairs. In SL3 and SL4, BRR prediction algorithms were paired with a screening process that only kept codes claimed by at least 300 patients and with non-zero coefficients within a LASSO regression. In SL5 and SL6, each prediction algorithm was paired with a screening algorithm that used all baseline variables, and only codes screened by the BRR algorithm, claimed by at least 300 patients and with non-zero LASSO coefficients. In SL7 and SL8, since claims codes were screened by the BRR algorithm outside the SL first, each prediction algorithm was then coupled with a screenig algorithm that used all baseline variables, and only codes claimed by at least 300 patients and with non-zero LASSO coefficients. Note that baseline covariates include both 2 demographic variables and 11 researcher-specified claims variables.

Performance measures for prediction models

The two main metrics that we used to assess the predictive performance of each prediction individual and SL algorithm were AUC and computation time. An ROC (receiver operating characteristic) curve is a graph capturing the performance of a binary classifier at all classification thresholds, and AUC, measuring the area under the entire ROC curve, represents the probability that the classifier ranks a random point from the positive case higher than a random point from the negative case. Unlike accuracy-based performance metrics, AUC will not be affected by the prior class distribution and thus more suitable for problems with imbalance binary outcomes [26] (in this study, 17.56% of positive cases). Also, when the operational misclassification costs are unknown or unequal, the AUC is a better indicator of classifier performance than the overall accuracy. In general, models with higher AUCs are preferred over those with lower AUCs. As discussed in Section 4.1, computational burden can be a big concern while implementing SL in large, high-dimensional datasets, especially when complex models are involved. Therefore, a smaller computation time indicates a more computationally efficient model. By the same token, it is computationally intractable to use bootstrap methods for variance estimation. LeDell *et al* [48] proposed a more computationally efficient influence curve (IC) - based approach to estimate the variance for cross-validated AUC. In this study, standard error estimates were calculated based on this IC-based method, using the **cvAUC** package [49]. Negative log-likelihood of each of the individual and SL algorithms is also provided in Appendix D. Note that the negative log-likelihood here means log loss (i.e., cross-entropy loss), measuring the uncertainty of the probabilities of the classifier by comparing them to the true outcome labels. The smaller log loss indicates a better model with smaller uncertainty.

For binary classification problems, other performance metrics of interest could be F1-score (when having a very small positive class) [18], H-measure (overcoming the fundamentally incoherent manner in terms of misclassification costs for AUC comparison) [26], partial AUC (focusing on regions that are frequency relevant to practical applications) [35, 82], and Kappa statistic (comparing an observed accuracy with an expected accuracy) [9]. LeDell *et al.* showed how to construct a Super Learner involving a metalearner that targets a user-specified metric in R [47].

4.3 Using the Bayesian risk ratio prediction algorithms within the SL

Figure 4.1 shows the AUC and computation time for each of the Bayesian RR prediction algorithms based on the individual claims codes in the *NASH_nogroup* and *NASH_pregroup* datasets, respectively. Among the unregularized Bayesian RR models that implement logistic regressions on the set of claims codes selected by BRR criteria only, BRR $RR147068$ that included claims codes with $BRR > 1.47$ or $BRR < 0.68$ generally performed best in terms of maximizing the AUC with the AUC approximately 88.3% (95% CI 87.29%, 89.35%) for

the *NASH_nogroup* data, and 81.60% (95% CI 80.32%, 82.88%) for the *NASH_pregroup* data. As more claims codes were added into the Bayesian RR prediction models, the AUC tended to decrease. Similar patterns were observed among the L1-regularized (LASSO) BRR models and the column sparsity regularized BRR models that only used codes claimed by at least 300 patients. The AUC was generally higher in the regularized BRR models than in the unregularized models, especially after BRR $RR190053$, indicating regularization can slightly increase the AUC performance. In this study, the relatively large sample size may make the benefits of regularization less obvious. It is interesting that regularization based on column sparsity had an effect similar to L1 regularization in terms of AUC maximization, which reveals that controlling the sparseness of claims codes can sometimes be very effective in reducing the risk of overfitting. Figure 4.1 also shows that the computation times were similar for all models with a smaller number of claims codes. As the number of selected codes increased, the computation time for unregularized BRR models increased substantially relative to the regularized models. With a goal of maximizing the AUC with a reasonable computation time, we chose the BRR model with $BRR > 1.47$ or $BRR < 0.68$ in the rest of the study. The number of claims codes selected by different screening methods is shown in Figure 4.2, which further indicates the importance of feature selection and dimension reduction in the claims database.

In Table 4.2, we compared the AUC and computation time performance of SL1, SL2, SL3, SL4 and the best L1-regularized BRR model for both the *NASH_nogroup* and *NASH_pregroup* dataset. The SL2 and SL4 were not applicable for the *NASH_nogroup* data since it didn't have the 11 researcher-specified claims variables and thus only two baseline covariates (age and sex) could be used by individual ML algorithms. Some of the 19 algorithms needed more than 2 predictors for model training. Even though the 5 algorithms in SL1 could estimate the conditional probability of the outcome given just 2 baseline covariates, it would not reveal much information and we chose to skip this case. It's not surprising that SL2 and SL4 always achieved higher AUC values than SL1 and SL3, respectively, because we used all 19 diverse ML base learners in SL2 and SL4 but only 5 relatively fast ML base learners in SL1 and SL3. With the help of the L1-regularized BRR prediction models that used the remaining claims codes, the SL3 and SL4 resulted in an obvious increase in AUC when compared to SL1 and SL2 based on the *NASH_pregroup* dataset, respectively. The four SLs performed at least as well as, if not better than, each of the individual algorithms in their libraries including the BRR prediction models (not shown). Table 4.2 further shows that the best L1-regularized BRR model based on the *NASH_nogroup* dataset had a better AUC performance compared to the *NASH_pregroup* dataset, whereas the SL3 had the opposite AUC results. Computation time reported in Table 4.2 is installation-specific but clearly shows that SL1 and SL3 outperformed SL2 and SL4, respectively. Recall that the computation time for a SL built on LOGOCV is approximately twice the sum of the computation time for each base learner in the library, plus a small processing time for calculating the contribution of each algorithm to the final prediction. because SL usually fits each individual algorithm twice unless the algorithm provides a zero contribution to its convex optimization (i.e., $\hat{\alpha} = 0$ in step (4) from Section 4.2).

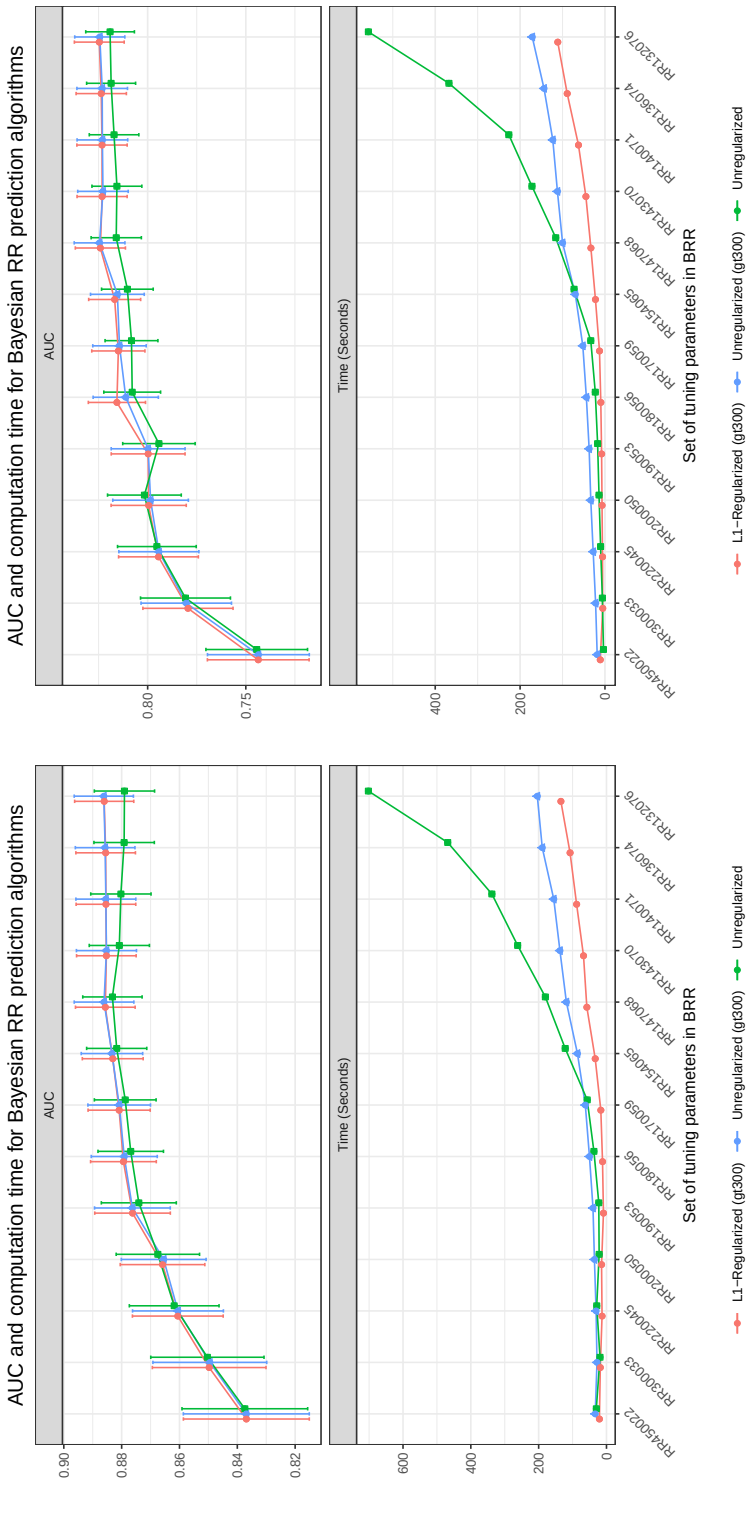
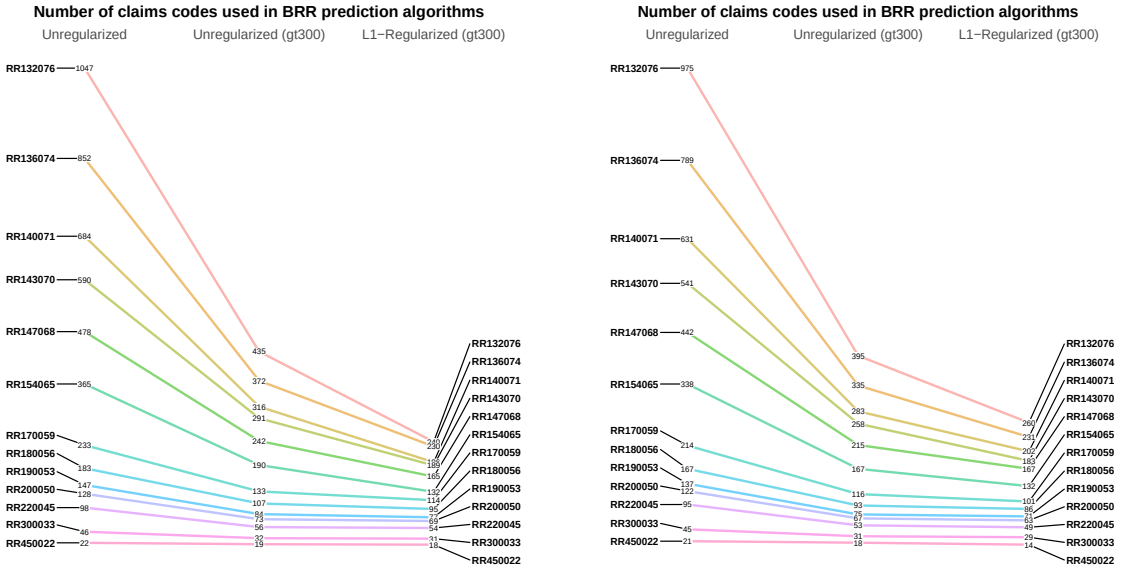


Figure 4.1: AUC and running time for each Bayesian RR prediction algorithm with a unique set of tuning parameters and any further screening step. (a) and (b) are results based on the *NASH_ nogroup* and *NASH_ pregroup* datasets, respectively. RR450022 is a logistic model regressing on claims codes whose Bayesian RR are greater than 4.5 or smaller than 0.22, whereas RR147068 is a logistic model regressing on claims codes whose Bayesian RR are greater than 1.47 or smaller than 0.68. Unregularized BRR (green line) uses claims codes satisfying the BRR selection criteria; Unregularized BRR (gt300) (blue line) uses claims codes satisfying the BRR selection criteria and being claimed by at least 300 paitents; L1 regularized BRR (gt300) uses claims codes satisfying the BRR selection criteria, being claimed by at least 300 paitents, and having non-cofficients in the LASSO regression. The vertical lines represent 95% confidence intervals for each of the point estimates.

Compared to SL4, SL3 only needed half of the running time with a small decrease in its

AUC performance, indicating it could a good alternative to SL4 if computation time is a big concern.



(a) NASH_nogroup (b) NASH_pregroup

Figure 4.2: Number of claims codes selected by different screening methods. Number of codes selected by the Bayesian RR method (the left column, named unregularized BRR), column sparsity based regularization (the middle column, named unregularized BRR (gt300)), and L1-regularization (the right column, named L1-regularized BRR) for the NASH_nogroup (a) and NASH_pregroup (b) dataset. Column sparity based regularization selects codes claimed by at least 300 patients (i.e., gt300). From left to right, each screening method uses the dataset screened by the previous screening method(s). For top to bottom, the set of OR cutoffs becomes tighter and less number of codes will be included. $RR132076$ indicates selecting codes with $BRR > 1.32$ or $BRR < 0.76$, and $RR450022$ selecting codes with $BRR > 4.5$ or $BRR < 0.22$.

4.4 Using the Bayesian risk ratio screening algorithms within the SL

In the previous section, individual machine learning algorithms were restricted to only baseline covariates, whereas the Bayesian RR prediction algorithms could utilize the remaining individual claims codes. The results clearly illustrates that including the additional BRR prediction algorithms in the SL library improved the prediction performance in terms of AUC (and negative log-likelihood, according to Table B1 in Appendix D). To further understand the effect of the individual claims codes on the prediction of NASH diagnoses, we utilized

the Bayesian RR method as a part of the screening process before passing the claims codes to the machine learning algorithms. Recall that in this section, each base learner in the SL was coupled with a comprehensive screening algorithm that included the Bayesian RR method, column sparsity-based regularization and L1-regularization. We used 9 fixed pairs of tuning parameters for the BRR method, where each pair would select a subset of claims codes that was further screened by the rest of the screening algorithm and combined with the baseline covariates. Then each of the individual ML learners in the SL library was fitted on the screened dataset.

Table 4.2: AUC and running time for SL1, SL2, SL3, SL4 and the best LASSO BRR model based on the *NASH_nogroup* (a) and *NASH_pregroup* (b) datasets. The best L1-regularized BRR model is *RR*147068 for both datasets. N.A. indicates those SL methods do not apply to such cases.

(a) *NASH_nogroup*

Performance Metric	SL1*	SL2	SL3*	SL4	Best LASSO BRR
AUC (95% CI)	N.A.	N.A.	0.885 (0.874, 0.895)	N.A.	0.885 (0.874, 0.895)
Running Time (secs)	N.A.	N.A.	1104.21	N.A.	118.25

(b) *NASH_pregroup*

Performance Metric	SL1*	SL2	SL3*	SL4	Best LASSO BRR
AUC (95% CI)	0.843 (0.831, 0.855)	0.850 (0.838, 0.861)	0.888 (0.878, 0.898)	0.889 (0.879, 0.899)	0.825 (0.812,, 0.837)
Running Time (secs)	51.07	702.57	1081.08	2194.36	99.90

Note: * indicates smaller SL libraries.

Figure 4.3 depicts the computation time for SL5 and SL6 across different screening algorithms based on the 9 fixed pairs of Bayesian RR tuning parameters. Because of the larger number of covariates, both SL5 and SL6 had a substantial increase in the computation time even though the time in SL5 was approximately one-tenth of that in SL6. From Figure 4.4, we see that increasing the number of screened variables in the dataset for individual ML learners improved the AUC performance, especially for the *NASH_nogroup* dataset. In the *NASH_pregroup* dataset, there were significant increases in AUC from the baseline results to the *RR*450022 results in both SL5 and SL6, but the subsequent growth in AUC became less pronounced as the number of covariates increased. The results suggest that the remaining claims codes could help improve prediction performance of the machine learning algorithms, and the most useful claims codes are already captured by the most restricted pair of BRR parameters *RR*450022. In the *NASH_nogroup* dataset, the *RR*450022 results

in both SL5 and SL6 were not better than the baseline results, indicating the importance of the 11 researcher-specified claims variables.

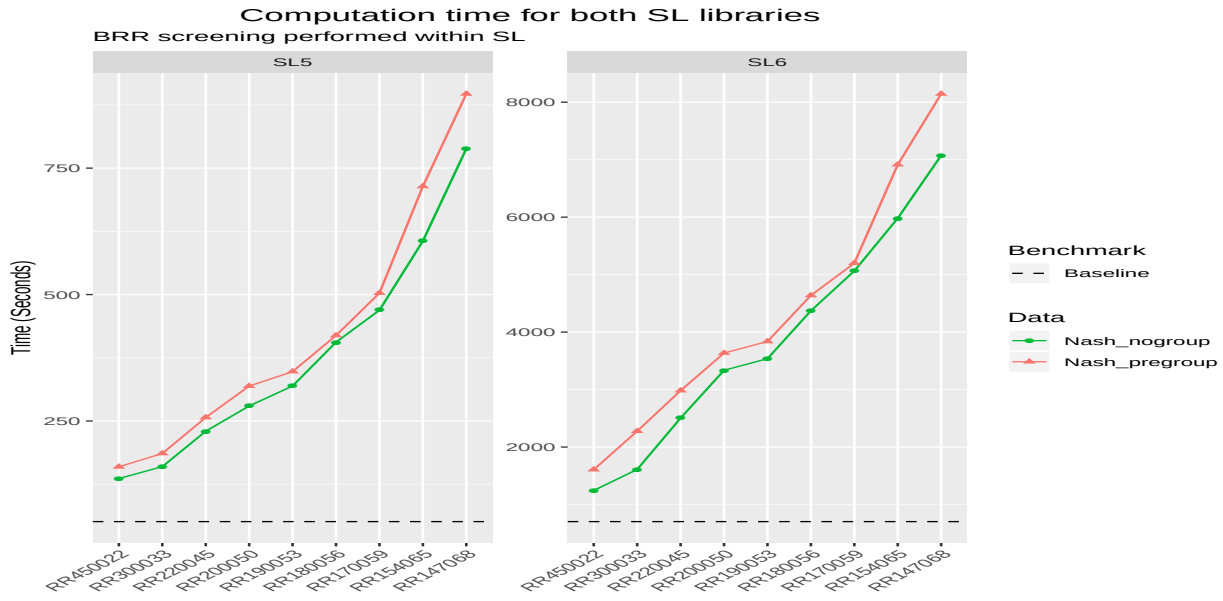


Figure 4.3: Computation times for SL5 and SL6 with 9 different screening algorithms in the *Nash_nogroup* and *Nash_pregroup* datasets. SL5 is a smaller SL library containing 5 relatively fast prediction algorithms, whereas SL6 is a larger SL library containing all 19 pre-specified prediction algorithms. *RR*450022 represents the criteria for the Bayesian RR screening part of the entire screening process.

Figure 4.4 further shows that the AUC performance based on the *NASH_nogroup* and *NASH_pregroup* datasets became closer as the pair of BRR tuning parameters was less tight. In Figure 4.5, we compared the performance of AUC for SL5, SL6 and each of the 19 machine learning algorithms for the *NASH_nogroup* and *NASH_pregroup* datasets, where each of the 9 panels represents the AUC results based on the specific pair of BRR tuning parameters. The SL6 outperformed SL5 and all the other individual machine learning algorithms in terms of maximizing the AUC, however, the SL5 that included the five fast ML learners was outperformed by some individual algorithms in some cases such as gradient boosting in *RR*450022 and penalized multinomial regression in *RR*220045. Overall, gradient boosting was the best individual algorithm in terms of maximizing the AUC.

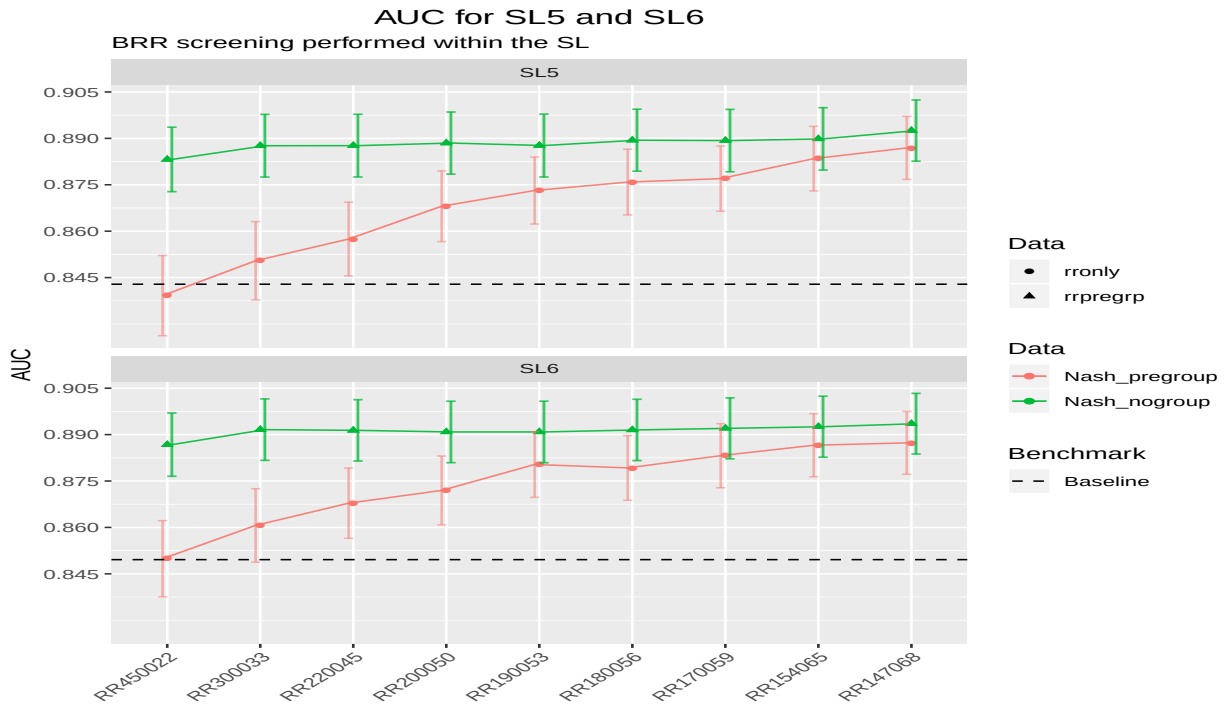


Figure 4.4: AUC for SL5 and SL6 with 9 different screening algorithms in the *Nash_nogroup* and *Nash_pregroup* datasets. SL5 is a smaller SL library containing 5 relatively fast prediction algorithms, whereas SL6 is a larger SL library containing all 19 pre-specified prediction algorithms. The horizontal axis represents the criteria for the Bayesian RR screening part of the entire screening process. For example, $RR450022$ indicates codes with $BRR > 4.5$ or $BRR < 0.22$ will be kept. The vertical lines represent 95% confidence intervals for each of the point estimates.

4.5 Using the Bayesian risk ratio screening algorithms outside the SL

In the previous section, the Bayesian RR screening algorithm was part of the cross-validation step that was performed within the SL, indicating the screening results were honest. However, the computational cost associated with a high-dimensional data set can be prohibitive when there are too many observations. In addition, administrative claims data contains a vast number of unique codes which could act as potential predictors, and neither R nor Python could handle such a wide dataset directly. A solution would be to generated a subset of claims codes via the Bayesian RR screening algorithm outside the SL (in this study, performed using an analytical platform from Teradata), and to keep everything else identical to the procedure in Section 4.4. In this section, we again used the 9 fixed pairs of Bayesian RR tuning parameters to generate 9 subsets of claims codes.

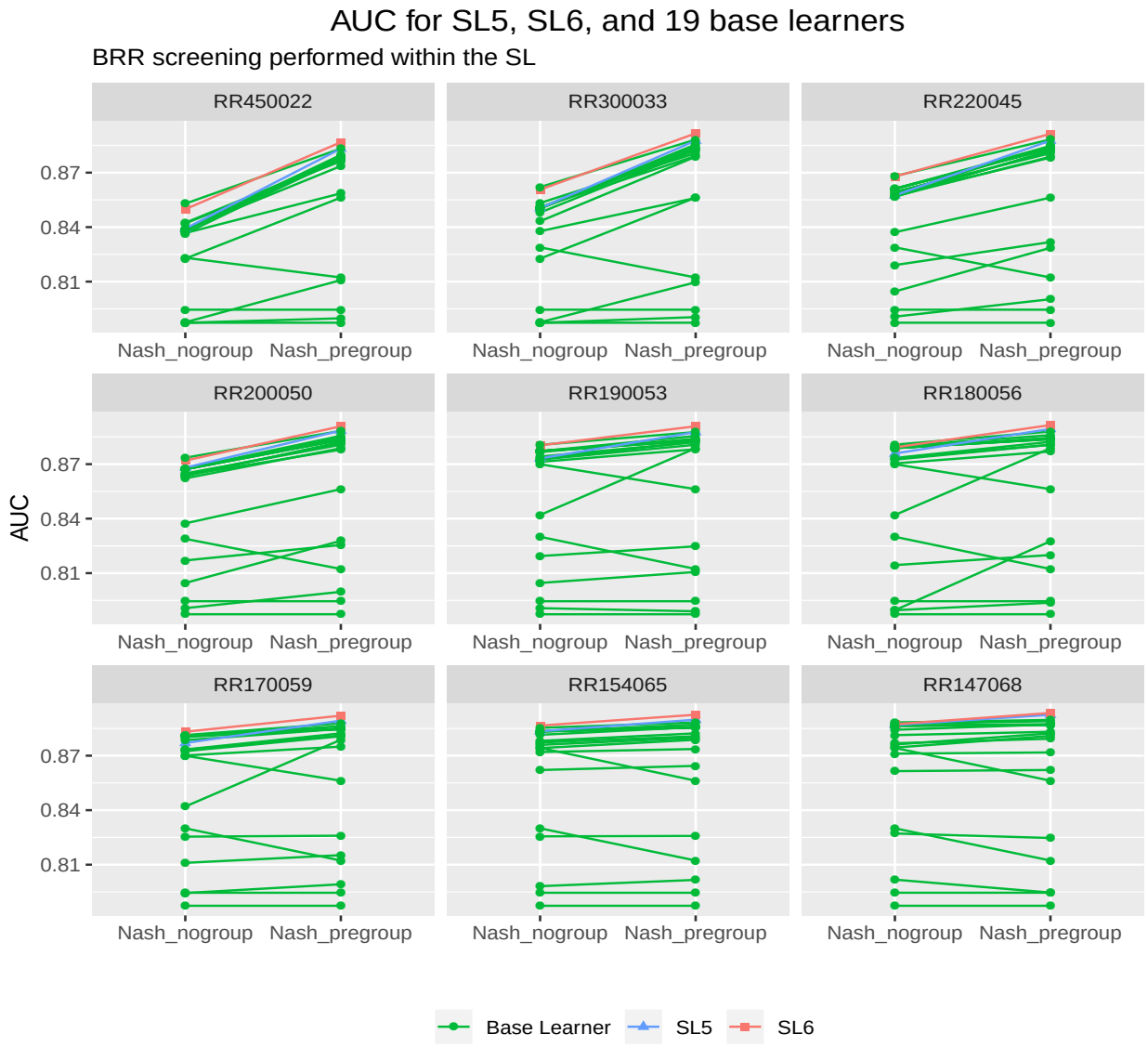


Figure 4.5: AUC for each of the individual algorithms, SL5 and SL6 with a 9 different screening algorithms in the *Nash_nogroup* and *Nash_pregroup* datasets. SL5 is a smaller SL library containing 5 relatively fast prediction algorithms, whereas SL6 is a larger SL library containing all 19 pre-specified prediction algorithms. The horizontal axis represents the criteria for the Bayesian RR screening part of the entire screening process. For example, $RR450022$ indicates codes with $BRR > 4.5$ or $BRR < 0.22$ will be kept.

Then each of the subsets was combined with the baseline covariates to generate an augmented dataset that would be utilized by SL7 and SL8. In order to further understand how the 211 claims codes associated with the 11 researcher-specified conditions would affect the performances of SL7 and SL8, we constructed a third dataset in which all 211

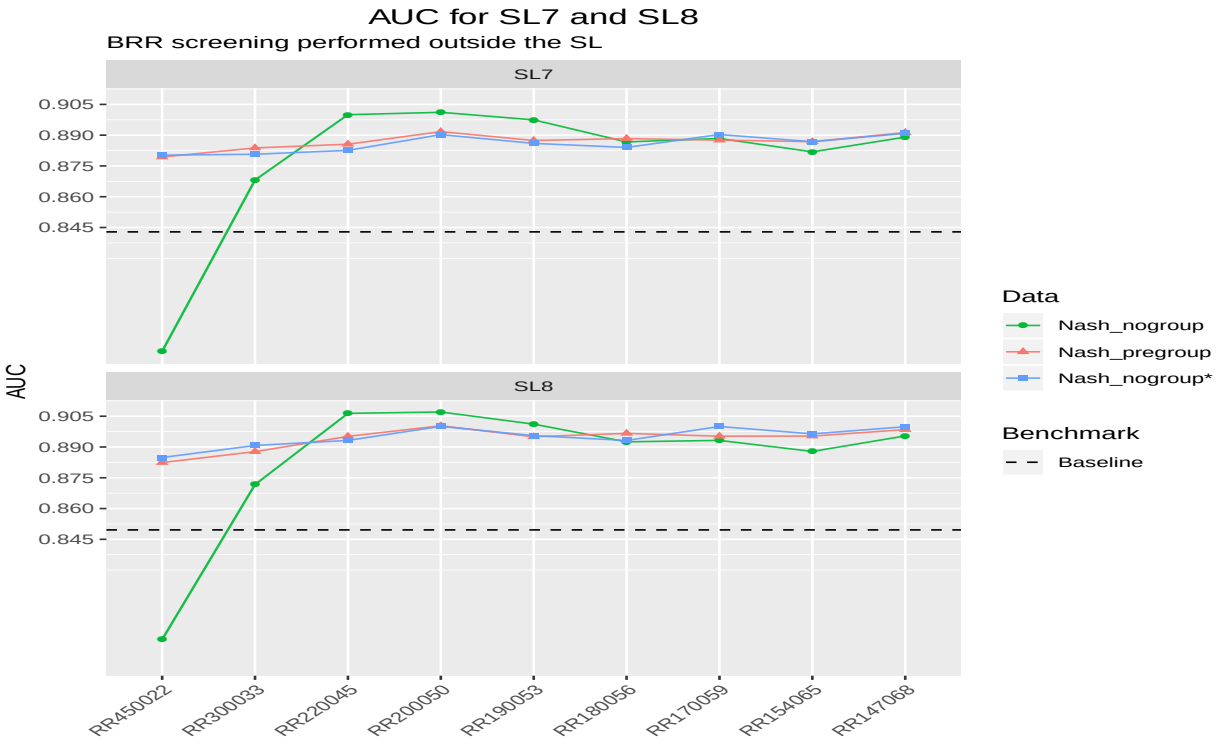


Figure 4.6: AUC for SL7 and SL8 with 9 differenet screening algorithms in the *Nash_nogroup*, *Nash_pregroup* and *Nash_nogroup** datasets. SL7 is a smaller SL library containing 5 relatively fast prediction algorithms, whereas SL8 is a larger SL library containing all 19 pre-specified prediction algorithms. The horizontal axis represents the criteria for the Bayesian RR screening part of the entire screening process. For example, $RR450022$ indicates codes with $BRR > 4.5$ or $BRR < 0.22$ will be kept.

codes were forced to stay in the screened subset after the Bayesian RR screening step regardless of whether they had been selected by the BRR method. For simplicty, we used *NASH_nogroup** to denote the third dataset. In Figure 4.6, we compared the AUC performance of SL7 and SL8 across 9 different external Bayesian screening algorithms based on the *NASH_nogroup*, *NASH_pregroup*, and *NASH_nogroup** datasets. The AUC pattern for the *NASH_pregroup* dataset was similar when the BRR screening step was performed within the SL, except that it reached the top early at the criterion of $RR200050$ (i.e., selected claims codes whose Bayesian RR were greater than 2.00 or smaller than 0.50). A similar pattern was found for the *NASH_nogroup** dataset, suggesting the cross-validated screening step that includes column sparsity regularization and LASSO regularization would automatically select the most relevant predictors and help SLs maintain a good AUC performance even if the individual codes were not grouped into the 11 researcher-specified variables. Unlike in Figure 4.4, the AUC for the *NASH_nogroup* dataset increased substantially from $RR450022$

to *RR*220045, but then decreased slightly to a relatively stable value.

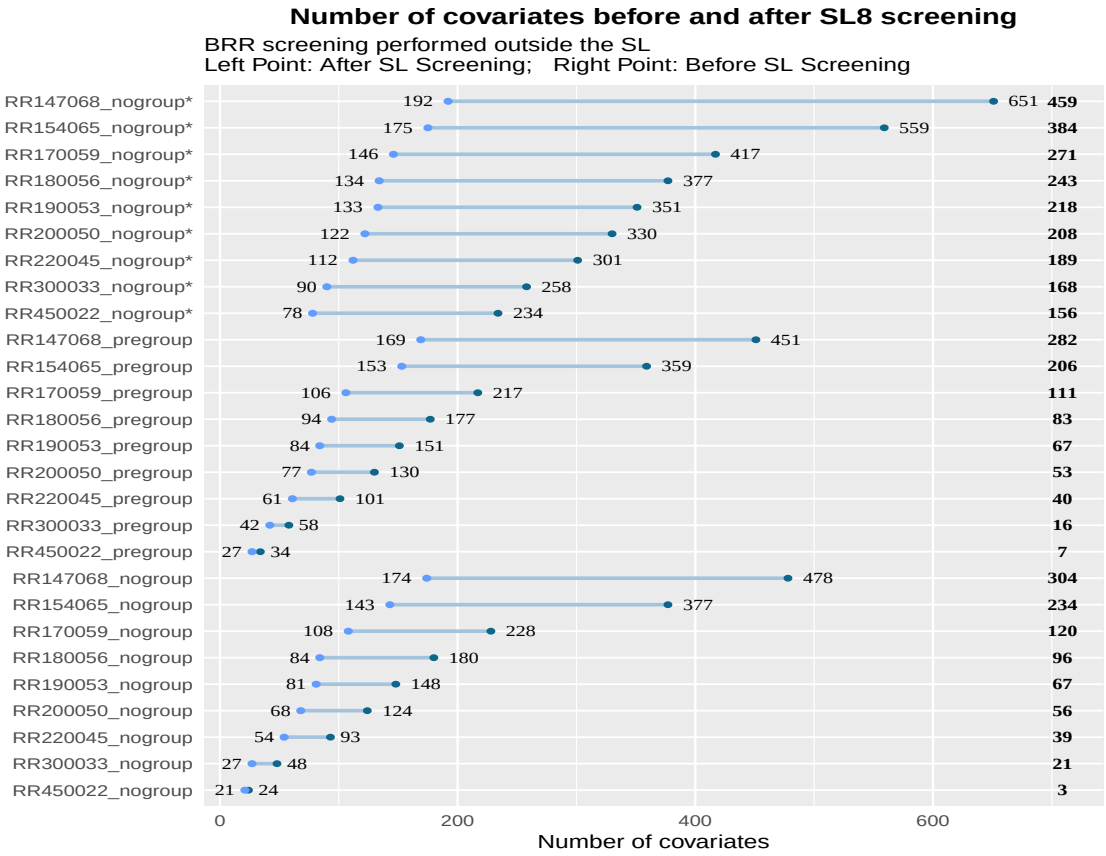


Figure 4.7: Number of covariates before and after SL8 screening step based on each of the Bayesian RR screened datasets. RR147068_only* indicates the SL8 is fitted on both baseline covariates and claims codes selected by Bayesian *RR*147068 screening step outside the SL based on the *NASH_nogroup* dataset.RR450022_ pregroup indicates the SL8 is fitted on both baseline covariates and claims codes selected by Bayesian *RR*450022 screening step outside the SL based on the *NASH_pregroup* dataset. The right endpoint of a line segment represents the number of covariates in the original BRR screened data. whereas the left endpoint represents the number of covariates passing the SL screening algorithm. The bold number next to each line segment represents the difference between the two endpoints.

Figure 4.7 depicts the number of covariates before and after the SL8 screening step when fitting SL8 on each of the Bayesian RR screened dataset. Even when all 211 codes were included in the *NASH_nogroup** dataset prior to the SL process, the two-step screening algorithm did a good job selecting the important features for training the model. It further provides evidence for the AUC pattern similarities for the *NASH_pregroup*, and *NASH_nogroup** datasets (Figure 4.6). Table 4.3 shows that in the *NASH_pregroup* dataset, the AUC performances improved from SL2 to SL4 and SL4 to SL6. Apparently, the increase

Table 4.3: AUC and running time for larger SL libraries SL2, SL4, SL6 and SL8 based on the *NASH_nogroup*, *Nash_pregroup* and *Nash_nogroup** datasets. SL2 included ML algorithms that utilized baseline covariates. SL4 expanded SL2 libraries with the 9 BRR prediction algorithms that utilized claims codes. SL6 included ML algorithms that were coupled with a three-step screening algorithm (See Table 4.1), whereas the screening algorithm in SL8 was two-step since the Bayesian RR step was performed outside the SL. N.A. indicates those SL methods do not apply in such cases.

(a) *NASH_nogroup*

Performance Metric	SL2	SL4	SL6*	SL8*
AUC (95% CI)	N.A.	N.A.	0.887 (0.877, 0.897)	0.907 (0.895, 0.920)
Running Time (secs)	N.A.	N.A.	7073.21	3544.80

(b) *NASH_pregroup*

Performance Metric	SL2	SL4	SL6*	SL8*
AUC (95% CI)	0.850 (0.838, 0.861)	0.889 (0.879, 0.899)	0.894 (0.886, 0.903)	0.900 (0.893, 0.907)
Running Time (secs)	702.57	2194.36	8144.24	5246.27

* indicates the results are based on the BRR $RR147068$ screening method.

in AUC from SL2 to SL4 was large but the increase from SL4 to SL6 was small. It indicates that the Bayesian RR prediction models performed well in the datasets we constructed. Fitting more complicated machine learning algorithms on the individual claims codes doesn't yield a significant gain in AUC. Also, the AUC difference between SL6 and SL8 based on the *NASH_pregroup* dataset was quite small, and the running time for SL8 was about half that of SL6. This suggests that performing the Bayesian RR screening method outside the SL is a good choice when the original dataset includes more predictors than can be managed by R or Python. Similar patterns were found for smaller SL libraries SL1, SL3, SL5, and SL8 (See Table B2 in Appendix B).

Table 4.4 depicts the relative contributions to the SL2, SL4, SL6 and SL8 predictions from each of the 19 algorithms whose contribution is greater than 0.01. Overall, the gradient boosting algorithm (gbm) made the largest contribution to the prediction. Multivariate adaptive regression splines (gcvEarth) contributed a great deal in SL2, SL4 and SL6 but much less in SL8. When the Bayesian RR prediction models were included in the SL, some of the BRR models were major contributors such as $RR450022$ and $RR147068$.

Table 4.4: Contribution of each algorithm to the final convex combination for SL2, SL4, SL6 and SL8 based on the *NASH_pregroup* dataset. Only algorithms whose contributions are greater than 0.02 are presented here. SL6 shows the algorithm contributions based on the BRR $RR147068$ screening process, and SL8 shows the contributions based on the baseline covariates dataset augmented by the $RR200050$ screened covariates.

SL2

Base Learner	Weights
gcvEarth_All	0.168
earth_All	0.168
gbm_All	0.112
sda_All	0.060
pda_All	0.060
fda_All	0.057
rpart_All	0.057
C5.0Rules_All	0.057
C5.0Tree_All	0.057
ctree2_All	0.057
LogitBoost_All	0.053
glmboost_All	0.039

SL4

Base Learner	Weights
gbm_screen.baselines	0.188
BRR.450022_Codes	0.140
BRR.147068_Codes	0.140
gcvEarth_screen.baselines	0.109
fda_screen.baselines	0.046
rpart_screen.baselines	0.046
earth_screen.baselines	0.046
C5.0Rules_screen.baselines	0.046
C5.0Tree_screen.baselines	0.046
ctree2_screen.baselines	0.046
BRR.154065_Codes	0.045
BRR.300033_Codes	0.022
naivebayes_screen.baselines	0.022

SL6

Base Learner	Weights
sda_screen.gt300.lasso	0.200
gbm_screen.gt300.lasso	0.194
gcvEarth_screen.gt300.lasso	0.101
pam_screen.gt300.lasso	0.083
pda_screen.gt300.lasso	0.081
C5.0Tree_screen.gt300.lasso	0.070
rpart_screen.gt300.lasso	0.061
naivebayes_screen.gt300.lasso	0.060
LogitBoost_screen.gt300.lasso	0.040
fda_screen.gt300.lasso	0.034
glm_screen.gt300.lasso	0.020

SL8

Base Learner	Weights
LogitBoost_screen.gt300.lasso	0.136
gbm_screen.gt300.lasso	0.135
pam_screen.gt300.lasso	0.095
C5.0Tree_screen.gt300.lasso	0.082
sda_screen.gt300.lasso	0.069
C5.0Rules_screen.gt300.lasso	0.060
rpart_screen.gt300.lasso	0.058
glmboost_screen.gt300.lasso	0.057
gcvEarth_screen.gt300.lasso	0.055
pda2_screen.gt300.lasso	0.043
fda_screen.gt300.lasso	0.041
ctree2_screen.gt300.lasso	0.039
bayesglm_screen.gt300.lasso	0.030
multinom_screen.gt300.lasso	0.029
glm_screen.gt300.lasso	0.028
pda_screen.gt300.lasso	0.021

4.6 Discussion

In this study, we have shown that stacked ensemble learning such as Super Learner for predictive modeling in high-dimensional administrative claims databases can be advantageous. We considered SL libraries that treated the Bayesian risk ratio algorithms as individual base learners, and libraries that coupled each prediction algorithm with a comprehensive screening algorithm where the Bayesian risk ratio method was part of the screening process.

Let's focus on the results based on the *Nash_pregroup* dataset which compressed 211 individual claims codes into 11 researcher-specified claims variables. We found that the AUC performance was improved by incorporating the Bayesian RR prediction algorithms that utilized individual codes as base learners within the SL (from 85% in SL2 to 88.9% in SL4), indicating that the screening selected claims codes can be used to complement domain knowledge for variable selection and may improve the performance of predictive modeling. We also found that the effects of column sparsity-based regularization and L1 regularization on dimension reduction were similar, indicating the column sparsity is a very good indicator for its predictive importance in high-dimensional datasets. Another advantage of selecting codes based on its sparsity prior to some penalized regression is to reduce computation time. Recall that the SL focuses on selecting a combination of the individual learners that are optimal in terms of reducing a cross-validated loss function (e.g., rank loss here) for final predictions. In this data setting, gradient boosting and the Bayesian RR prediction algorithm had the largest contributions to the final prediction within the SL4.

When fitting SL6, we allowed all prediction algorithms to utilized BRR-screened individual claims codes rather than just logistic regressions, which improved the SL's flexibility in model training. However, the small increase in AUC from SL4 to SL6 suggests there is not much hidden information within the claims codes that can be exploited by using more flexible models than main-term logistic regressions, at least in this study. The purpose of SL8 was to check the severity of model overfitting if performing the Bayesian RR screening outside the SL. In this case, the SL was fitted on the dataset that combined the baseline covariates with the subset of claims codes screened by the BRR. The results show that the AUC values and trends were quite similar between SL6 and SL8 although SL8 reached the top earlier at the criteria of $RR200050$, which indicates there is a small degree of overfitting. SL8 ran in half the time that it took to run SL6. Therefore, SL8 could be an alternative to SL6 if there are concerns about computation time and data storage space (e.g., a data set with over 100,000 columns that is hard to handle in R or Python).

It is interesting that the AUC performance achieved with the *Nash_nogroup* dataset was quite similar that with the *Nash_pregroup* dataset. Recall that all claims codes were included in the *Nash_nogroup* as individual predictors. Combined with the results in SL2, this suggests two things: First, the feature engineering step is effective since those 11 researcher-specified covariates are quite informative for the predictive modeling. Second, the cross-validated screening step within SL can effectively capture important codes even without the expert knowledge. We also noticed that the AUC trend in SL8 based on *Nash_nogroup* was more aggressive than the trend in SL6, indicating a larger degree of overfitting than the re-

sults based on *Nash_pregroup*. Furthermore, we compared the AUC performance of SL using smaller libraries (SL1, SL3, SL5, and SL7) to SL using larger libraries (SL2, SL4, SL6, and SL8). The data analysis illustrates that a smaller library could reduce computation cost without sacrificing much performance. Therefore, future research can explore the performance of SL by only including the individual algorithms with large contributions if computation time is a big concern. It is worth mentioning that there are other popular strategies for variable selection in high-dimensional claims datasets such as the high-dimensional propensity score (hdPS) [65] that can be combined with SL [36]. Further analysis can explore which variable selection method does better in different data settings.

Overall, we conclude that constructing a Super Learner with a rich library of diverse prediction algorithms coupled with a comprehensive screening algorithm is promising for predictive modeling in high-dimensional administrative claims databases.

4.7 Chapter Appendix

Base learners description and demo R code for generating them

Recall that If a prediction model had at least one hyperparameter, we would use a grid search procedure with 5-fold cross validation, allowing 3 different values to test for each tuning parameter, to select the optimal hyperparameter setting, otherwise, no cross validation would be necessary.

The following R code shows how to create such base learners for a Super Learner library, using the **caret** package[40].

```
# ------------------------------------------------------------------------
# SL wrappers: generating base learners for SL prediction
# ------------------------------------------------------------------------
# stochastic gradient boosting
SL.caret.gbm <- function(..., method = "gbm", tuneLength = 3,
                         trControl = trainControl(method="cv", number=5,
                                                  verboseIter=TRUE)) {
  SL.caret(..., method = method,
           tuneLength = tuneLength, trControl = trControl)
}
# classification and regression trees
SL.caret.rpart <- function(..., method = "rpart", tuneLength = 3,
                           trControl = trainControl(method="cv", number=5,
                                                    verboseIter=TRUE)) {
  SL.caret(..., method = method,
           tuneLength = tuneLength, trControl = trControl)
}
# Note, for learners without hyper-parameter, no need to use cross CV
```

Table 4.5: List of 19 prediction algorithms that have been used in NASH data analysis and hyperparameter(s) used in the corresponding R package

Algorithm	Description	R package(s)	Hyperparameter(s)
glm	generalized linear model	stats [61]	None
bayesglm	Bayesian generalized linear model	arm [22]	None
glmnet	elastic-net regularized generalized linear model	glmnet [20]	Mixing parameter Regularization Parameter
glmboost	boosted generalized linear model	plyr [83] mboost [8]	Number of Boosting Iterations AIC Prune?
multinom	penalized multinomial regression	nnet [81]	Weight Decay
LogitBoost	boosted logistic regression	caTools [72]	Number of Boosting Iterations
naivebayes	naive Bayes	naivebayes [52]	Laplace Correction Distribution Type Bandwidth Adjustment
pam	nearest shrunken centroids	pamr [31]	Shrinkage Threshold
sda	shrinkage discriminant analysis	sda [1]	Diagonalize shrinkage
pda	penalized discriminant analysis	mda [30]	Shrinkage Penalty Coefficient
pda2	penalized discriminant analysis	mda [30]	Degrees of Freedom
fda	flexible discriminant analysis	earth [12] mda [30]	Product Degree Number of Terms
Rpart	classification and regression trees (CART)	rpart [70]	Complexity Parameter
gcvEarth	multivariate adaptive regression splines	earth [12]	Product Degree
Earth	multivariate adaptive regression splines	earth [12]	Number of Terms Product Degree
C5.0Rules	single C5.0 rule-based models	C50 [39]	None
C5.0Tree	single C5.0 decision tree models	C50 [39]	None
ctree2	conditional inference tree	party [32]	Max Tree Depth 1 - P-Value Threshold
gbm	stochastic gradient boosting	gbm [63] plyr [83]	Number of Boosting Iterations Max Tree Depth Shrinkage Min. Terminal Node Size

```
# generalized linear model
SL.caret.glm <- function(..., method = "glm", tuneLength = 3,
                         trControl = trainControl(method="none",
                                                  verboseIter=TRUE)) {
  SL.caret(..., method = method,
           tuneLength = tuneLength, trControl = trControl)
}
```

Demo R code for a comprehensive screening method

The following R code shows how to implement the Bayesian Risk Ratio variable selection method, and then how to incorporate with column sparsity based regularization and L1 regularization screening methods.

```
myspread <- function(df, key, value) {
  # quote key
  keyq <- rlang::enquo(key)
  # break value vector into quotes
  valueq <- rlang::enquo(value)
  s <- rlang::quos(!!valueq)
  df %>% gather(variable, value, !!!s) %>%
    unite(temp, !!keyq, variable) %>%
    spread(temp, value)
}

# ------------------------------------------------------------------------
# RR_method: function to apply Bayesian Risk Ratio for variable selection.
# claims_df: data frame structured with "patid" (patient ID) in col 1,
#             "code" (code name) in col 2, and "nash" (outcome) in col 3.
#             It includes all code claims from one data source.
# X: data frame with "patid", outcome , baseline covariates, and all claim
#     codes that are avaible for the list of patients.
# RR_upp: upper cutoff for BRR (i.e., keep any code with RR above it).
# RR_low: lower cutoff for BRR (i.e., keep any code with RR below it).
# num_cutoff: Remove code with number of claims smaller than it
#              (Default to 0)
# ------------------------------------------------------------------------
RR_method <- function(claims_df, X, RR_upp, RR_low, num_cutoff = 0) {
  ## Step 2.1 Select codes satisfying RR conditions
  sub_claims_df <- claims_df[which(claims_df$patid %in% X$patid), ]
  sub_codes_ct <- sub_claims_df %>%
    # Compute # of claims for each code among NASH & non-NASH group, resp
    group_by_(.dots=c("code", "nash")) %>% dplyr::summarise(num = n()) %>%
    # Compute total and mean of # of claims, among NASH & non-NASH group
    group_by(nash) %>% dplyr::mutate(sumN = sum(num),
                                     avgN = mean((num))) %>%
    # Spread a key-value pair across multiple columns
    myspread(key = nash, value = c(num, sumN, avgN)) %>%
    # Rename column names (magrittr::set_names)
    set_names(c("code", "avgN_0", "num_0", "sumN_0",
```

```
                "avgN_1", "num_1", "sumN_1")) %>%
    # Drop any row with NA (either in NASH or non-NASH group)
    tidyr::drop_na() %>%
    # Compute Bayesian RR
    dplyr::mutate(ratio = ((num_1 + avgN_1)/(sumN_1 + 0.5)) /
                    ((num_0 + avgN_0)/(sumN_0 + 0.5))) %>%
    # Condition arguments:
    # number of claims per code cutoff (among NASH & non-NASH group)
    dplyr::filter((ratio > RR_upp | ratio < RR_low) &
                    num_0 > num_cutoff & num_1 > num_cutoff)
  code_ind_RRkeep <- which(names(X) %in% tolower(sub_codes_ct$code))
  return(code_ind_RRkeep)
}

# ------------------------------------------------------------------------
# screen.RR.gt300.lasso: function to implement a
#                          comprehensive screening algorithm.
# 1. Apply the Bayesian risk ratio screening method;
# 2. Select codes claimed by at least a certain amount of patients;
# 3. Select codes with non-zero coefs in a L1-regularized regression.
# alpha: Elastic net parameter, range [0, 1]. 0 = RIDGE and 1 = LASSO.
# minscreen: Minimal number of covariates allowed after screening.
# nlambda: Number of lambda values to check, recommended to be >= 100.
# nfolds: Number of folds for internal CV to optimize lambda.
# obs_cutoff: Minimal number of non-zero values per column.
# RR_upp, RR_low, code_cutoff, X, claims_df: see the function above.
# ...: Additional arguments passed to the screen.glmnet function.
# ------------------------------------------------------------------------
screen.RR.gt.lasso <- function(
  alpha = 1, minscreen = 5, nlambda = 100, nfolds = 5, obs_cutoff = 300,
  RR_upp = 2.0, RR_low = 0.5, code_cutoff = 25, X, claims_df, ...) {

  ## Step 0. identify indices for dx, rx, group, demographic variables
  dxrx_ind <- grep("^i[a-zA-Z]?[a-zA-Z]?[0-9]|^m[0-9]",
                   names(X), value = FALSE)
  grp_ind <- grep("^is_", names(X), value = FALSE)
  patid_ind <- grep("patid", names(X), value = FALSE)
  demogra_ind <- which(names(X) %in% c("age", "isFemale"))
  useless_ind <- setdiff(1:NCOL(X),
                         c(dxrx_ind, patid_ind, grp_ind, demogra_ind))

  if (length(dxrx_ind) == 0) {
```

```
    ## Step 1. If no RR applied, keep all variables
    whichVariable <- rep(TRUE, ncol(X))
    whichVariable[c(patid_ind, useless_ind)] <- FALSE  # patid removed
    return(whichVariable)
  } else {
    ## Step 2. Variable selection
    whichVariable <- rep(FALSE, ncol(X))

    ## Step 2.1 Always keeped demographic and grouped covariates
    whichVariable[c(grp_ind, demogra_ind)] <- TRUE

    ## Step 2.2 Select dx and rx codes satisfying RR conditions
    dx_ind_RRkeep <- RR_method(
      claims_df = condit_dx, X = X, RR_upp = RR_upp,
      RR_low = RR_low, code_cutoff = code_cutoff
    )
    rx_ind_RRkeep <- RR_method(condit_rx, X, RR_upp, RR_low, code_cutoff)
    dx_ind_RRkeep <- dx_ind_RRkeep$code_ind_RRkeep
    rx_ind_RRkeep <- rx_ind_RRkeep$code_ind_RRkeep
    dxrx_ind_RRkeep <- c(dx_ind_RRkeep, rx_ind_RRkeep)
    message(paste0("By setting RR > ",  RR_upp, " or RR < ", RR_low,
                   ", its selects ", length(dx_ind_RRkeep), " dx and ",
                   length(rx_ind_RRkeep), " rx codes, in total ",
                   length(dxrx_ind_RRkeep), " codes."))

    ## Step 2.4 Select columns with more than some number non-zero values
    dxrx_RRkeep_col_sum <- as.integer(colSums(X[, dxrx_ind_RRkeep]))
    dxrx_ind_RRkeep <- dxrx_ind_RRkeep[dxrx_RRkeep_col_sum >= obs_cutoff]
    message(paste0("By choosing to codes claimed by more than ",
                   obs_cutoff, " patients, ", length(dxrx_ind_RRkeep),
                   " codes are kept."))

    ## Step 2.5 Use LASSO to do the last screening step
    dxrx_lasso <-
      screen.glmnet(alpha=alpha, minscreen=minscreen, nfolds=nfolds,
                    nlambda=nlambda, X=X[, dxrx_ind_RRkeep], ...)
    whichVariable[dxrx_ind_RRkeep] <- dxrx_lasso
    message(paste0("By applying LASSO, ", sum(dxrx_lasso),
                   " codes are kept."))
    return(whichVariable)
  }
}
```

Demo R code for performing a AUC-maximizing Super Learner build on LOGOCV

The following R code gives an example that shows how to implement a AUC-maximizing Super Learner build on LOGOCV, where each prediction algorithm is coupled with a 3-step screening process described above.

```
# ------------------------------------------------------------------------
# LOGOCV.SL.traintest: function to implement SL build on LOGOCV
# Y: outcome vector.
# X: predictor data frame.
# SL.lib: either a character vector of prediction algorithms or
#          a list containing character vectors.
# meta_method: how to estimate the weights of the individual algorithms
#               in SL. Default to "method.AUC" (i.e., AUC-maximizing).
# train_frac: training test sets split ratio.
# rndseed: random seed to ensure model reproducibility.
# alpha: significance level. Default to 0.05.
# ------------------------------------------------------------------------
LOGOCV.SL.traintest <- function(Y, X, SL.lib, meta_method = "method.AUC",
                                train_frac = 0.9, rndseed = 1,
                                alpha = 0.05) {
  Yfamily <- ifelse(length(unique(Y)) > 2, gaussian(), binomial())
  set.seed(rndseed)
  train_ind <-
    sample(1:NROW(X), size = round(NROW(data) * train_frac), replace = F)
  trainX <- X[train_ind, ]; trainY <- Y[train_ind]
  testX <- X[-train_ind, ]; testY <- Y[-train_ind]

  # Create vectors to store results
  SL_pred <- rep(NA, length(testY))
  SL_res <-rep(NA, 10)
  names(SL_res) <-
    c("Screening", "AUC", "NLogLik", "Time", "Allvar",
      "UsedVar", "cvAUC", "se", "lowCI_cvAUC", "uppCI_cvAUC")
  SL_res["Allvar"] <- NCOL(trainX)
  SL_weights <- c()

  set.seed(rndseed)
  require(SuperLearner)

  start.time <- Sys.time()
```

```
  model.SL <- SuperLearner::SampleSplitSuperLearner(
    Y = trainY, X = trainX, family = Yfamily, SL.library = SL.lib,
    split = 0.9, method = meta_method, verbose = TRUE)
  end.time <- Sys.time()
  time.taken <- difftime(end.time, start.time, units = "secs")

  pred_SL <- predict(model.SL, newdata = testX, onlySL = TRUE)
  require(Metrics)
  auc_SL <- Metrics::auc(testY, pred_SL$pred)
  logloss_SL <- Metrics::logLoss(testY, pred_SL$pred)
  require(cvAUC)
  cvAUC_SL <- ci.cvAUC(pred_SL$pred, testY, confidence = 1 - alpha)[1:3]

  SL_pred <- pred_SL$pred
  SL_res["AUC"] <- round(auc_SL, digits = 5)
  SL_res["NLogLik"] <- round(logloss_SL, digits = 5)
  SL_res["Time"] <- round(as.numeric(time.taken), digits = 2)
  SL_res["UsedVar"] <- sum(model.SL$whichScreen[1, ])
  SL_res[7:10] <- round(unlist(cvAUC_SL), digits = 5)
  SL_weights <- as.vector(model.SL$coef)
  return(list(pred = SL_pred, res = SL_res, weight = SL_weights))
}

## Tuning grids for RR cutoff hypers
Grid_RR  <- data.frame(RR_upp = c(4.50, 1.47), RR_low = c(0.22, 0.68),
                       obs_cutoff = 300, code_cutoff = 25)
Grid_names <- paste0(as.character(Grid_RR[, 1] * 100),
                       paste0("0", as.character(Grid_RR[, 2] * 100)))

## Generate screening algorithms in SL based on cutoffs for Bayesian Risk Ratio
for (hyper in seq(length(Grid_names))) {
  eval(parse(file = "", text = paste(
      "screen.RR.gt300.lasso", Grid_names[hyper], " <- function (
        RR_upp = ", Grid_RR[hyper, 1], ", RR_low = ", Grid_RR[hyper, 2],
        ", obs_cutoff = ", Grid_RR[hyper, 3],
        ", code_cutoff = ", Grid_RR[hyper, 4], ", ...) {
        screen.RR.gt300.lasso(RR_upp = RR_upp, RR_low = RR_low,
                              obs_cutoff = obs_cutoff,
                              code_cutoff = code_cutoff, ...) }",
      sep = "")))}

## Create Superlearner libraries
```

```
SL_MLonly <- c("SL.caret.glm", "SL.caret.glmnet", "SL.caret.rpart",
               "SL.caret.gbm","SL.caret.naivebayes",  "SL.caret.pam")
SL_screen_Lib_lists <- lapply(RR_tunes, function(RR) {
  lapply(SL_MLonly, function(x) { c(x, RR)})
})

SL_results <- LOGOCV.SL.traintest(Y, X, SL.lib = SL_screen_Lib_lists[[1]],
                                  meta_method = "method.AUC",
                                  train_frac = 0.9, alpha = 0.05)
print(SL_results$res)
```

Negative log-likelihood results from NASH data analysis

Using the Bayesian risk ratio prediction algorithms within the SL

Figure 4.8 shows the negative log-likelihood for each of the Bayesian RR prediction algorithms based on the individual claims codes in the *NASH_nogroup* and *NASH_pregroup* datasets, respectively. General patterns were similar to AUC performance shown in Figure 1 with the negative log-likelihood tended to decrease as the number of covariates increased, but the effects became much less obvious once the RR criteria were less restricted than the BRR *RR*147068 model. Among the unregularized Bayesian RR models, the larger number of covariates in the last 4 models even resulted in higher negative log-likelihoods, which is likely a consequence of severe overfitting. Thus, the L1-regularized BRR *RR*147068 model is considered as the best model in terms of reducing negative log-likelihood with a reasonable computation time.

Table 4.6: Negative log-likelihood for SL1, SL2, SL3, SL4 and the best LASSO BRR model based on the *NASH_nogroup* (a) and *NASH_pregroup* (b) datasets. The best LASSO (L1-regularized) BRR model is *RR*147068 for both datasets. N.A. indicates those SL methods do not apply to such cases.

(a) *NASH_nogroup*

Performance Metric	SL1*	SL2	SL3*	SL4	Best LASSO BRR
Negative log-likelihood	N.A.	N.A.	0.285	N.A.	0.276

(b) *NASH_pregroup*

Performance Metric	SL1*	SL2	SL3*	SL4	Best LASSO BRR
Negative log-likelihood	0.328	0.310	0.281	0.286	0.331

Note: * indicates smaller SL libraries.

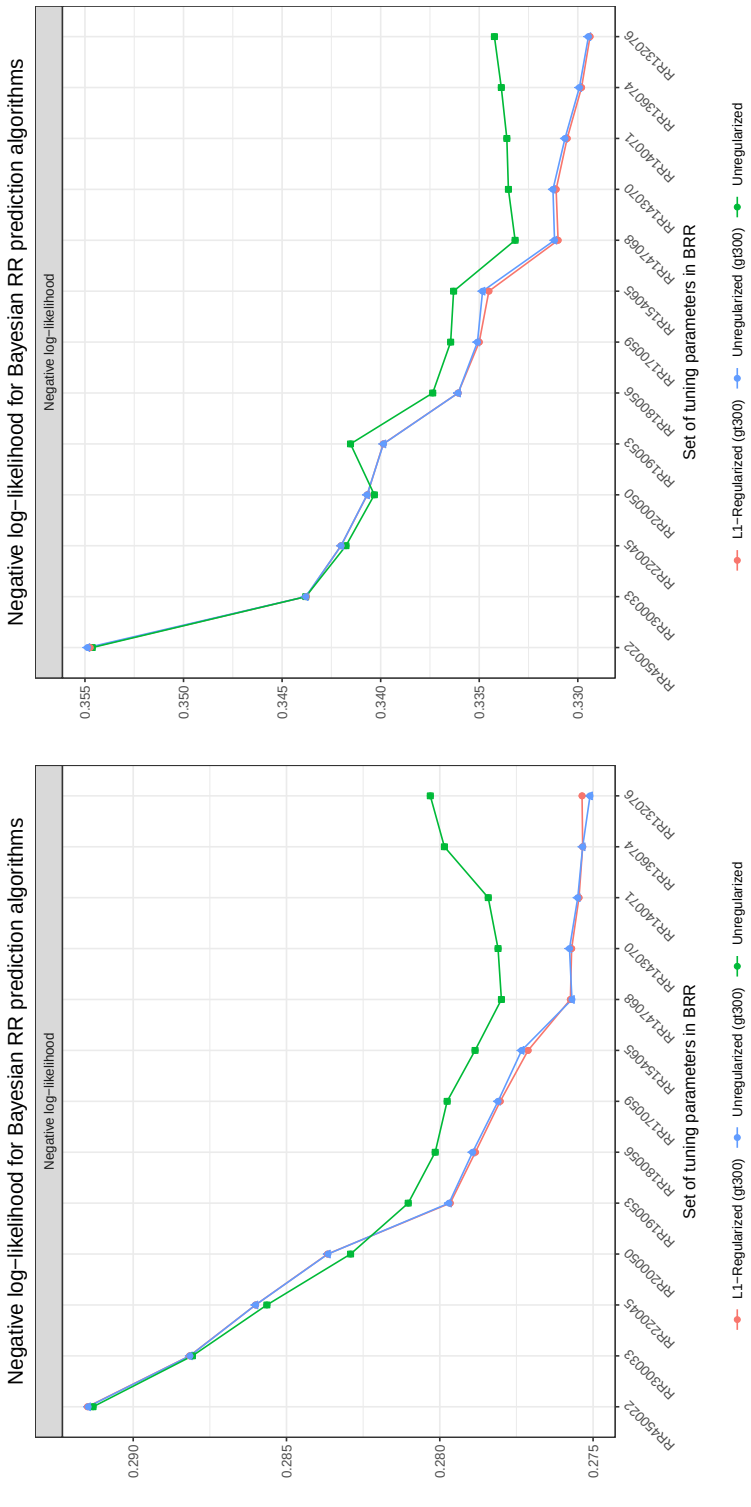


Figure 4.8: Negative log-likelihood for each Bayesian RR prediction algorithm with a unique set of tuning paramters and any further screening step. (a) and (b) are results based on the *NASH_nogroup* and *NASH_pregroup* datasets, respectively. $OR450022$ is a logistic model regressing on claims codes whose Bayesian RR are greater than 4.5 or smaller than 0.22, whereas $RR147068$ is a logistic model regressing on claims codes whose Bayesian RR are greater than 1.47 or smaller than 0.68. Unregularized BRR (green line) uses claims codes satisfying the BOR selection criteria; Unregularized BRR (gt300) (blue line) uses claims codes satisfying the BRR selection criteria and being claimed by at least 300 paitents; L1 regularized BRR (gt300) uses claims codes satisfying the BRR selection criteria, being claimed by at least 300 paitents, and having non-cofficients in the LASSO regression.

Although the goal of Super Learner in this study was to minimize the rank loss function (i.e., maximizing the AUC) for predicting NASH diagnoses, SL1 to SL4 showed good performance in terms of minimizing the negative log-likelihood (Table B1). In the *NASH_nogroup* dataset, the best L1-regularized BRR method achieved a lower negative log-likelihood than SL3, which may be a result of AUC-maximizing metalearning methods.

Using the Bayesian risk ratio screening algorithms within the SL

Figure 4.9 shows the negative log-likelihood performance for SL5 and SL6 with 9 differenet screening algorithms in the *Nash_nogroup* and *Nash_pregroup* datasets. The pattern for SL6 was similar to AUC performance shown in Figure 4. However, unlike in Figure 4 that AUC gradually increased for SL5 when the number of screened variables in the dataset for individual ML learners became larger, the negative log-likelihoods at $RR180056$ and $RR147068$ based on the *Nash_pregroup* dataset were much larger than the rest of the results. One of the reasons could be a large standard deviation in the predicted outcomes, which requires a further investigation here.

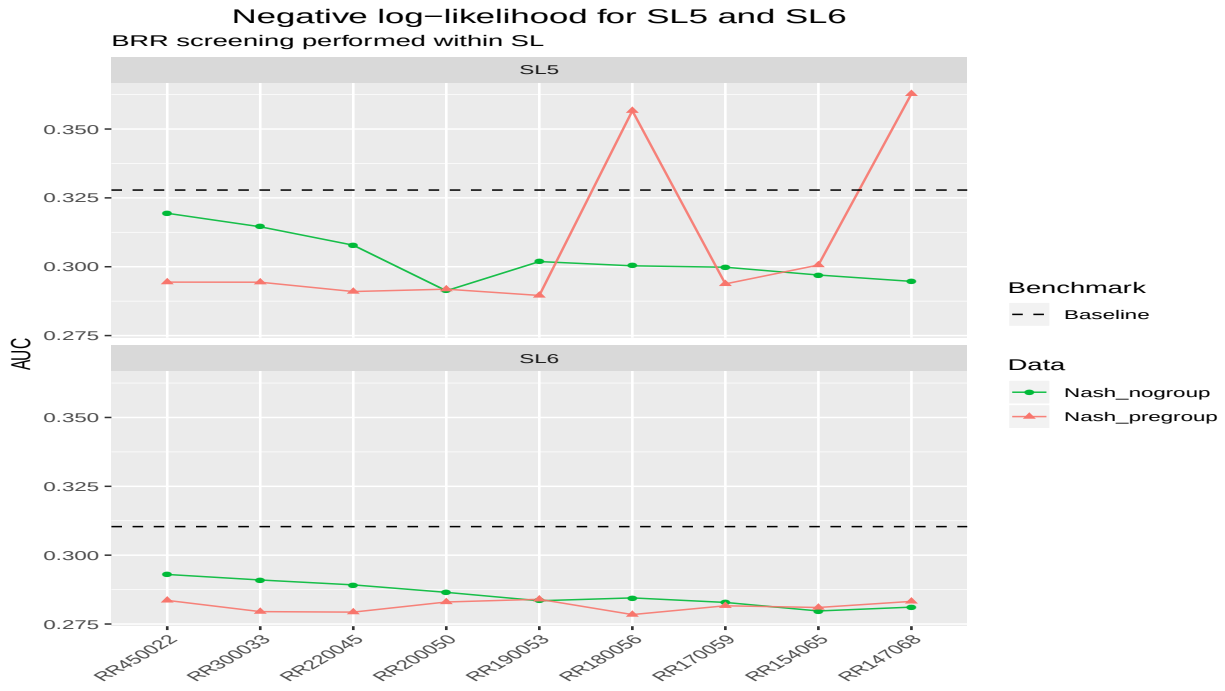


Figure 4.9: Negative log-likelihood for SL5 and SL6 with 9 differenet screening algorithms in the *Nash_nogroup* and *Nash_pregroup* datasets. SL5 is a smaller SL library containing 5 relatively fast prediction algorithms, whereas SL6 is a larger SL library containing all 19 prespecified prediction algorithms. The horizontal axis represents the criteria for the Bayesian RR screening part of the entire screening process. For example, $RR450022$ indicates codes with $BRR > 4.5$ or $BRR < 0.22$ will be kept.

Figure 4.10: Negative log-likelihood for each of the individual algorithms, SL5 and SL6 with a 9 different screening algorithms in the *Nash_nogroup* and *Nash_pregroup* datasets. SL5 is a smaller SL library containing 5 relatively fast prediction algorithms, whereas SL6 is a larger SL library containing all 19 pre-specified prediction algorithms. The horizontal axis represents the criteria for the Bayesian RR screening part of the entire screening process. For example, $RR450022$ indicates codes with $BRR > 4.5$ or $BRR < 0.22$ will be kept.

Figure 4.10 depicts the performance of negative log-likelihood for SL5, SL6 and each of the 19 machine learning algorithms for the *NASH_nogroup* and *NASH_pregroup* datasets. We can see there were always some individual algorithms that outperformed both SL5 and SL6. Again, it may be a result of AUC-maximizing metalearning methods (instead of log-

likelihood maximizing).

Using the Bayesian risk ratio screening algorithms outside the SL

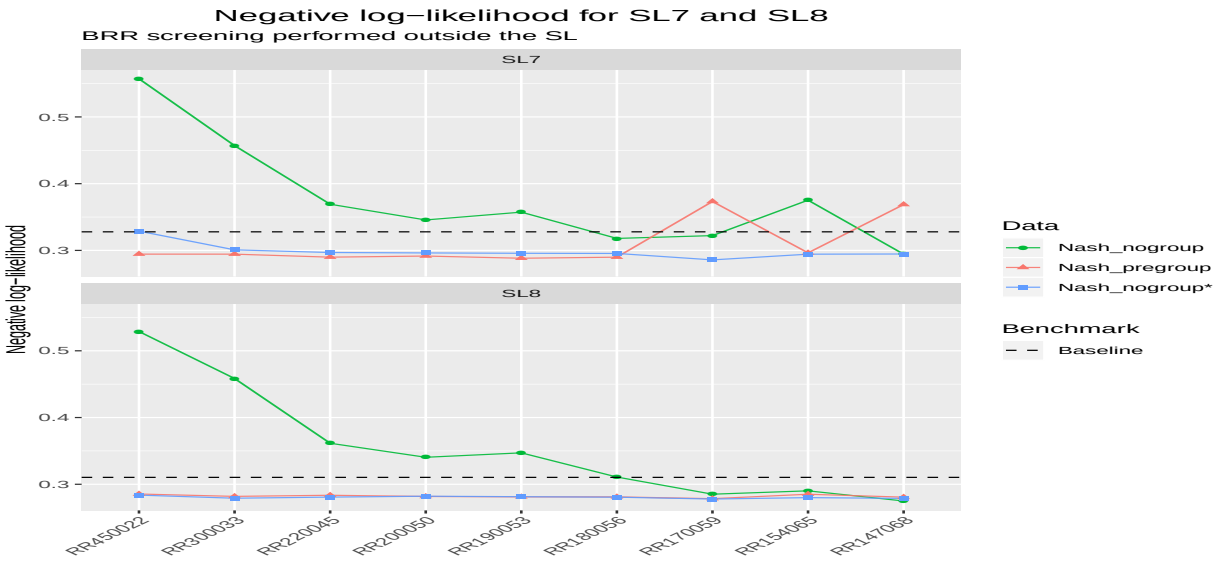


Figure 4.11: Negative log-likelihood for SL7 and SL8 with 9 differenet screening algorithms in the *Nash_ nogroup*, *Nash_ pregroup* and *Nash_ nogroup** datasets. SL7 is a smaller SL library containing 5 relatively fast prediction algorithms, whereas SL8 is a larger SL library containing all 19 pre-specified prediction algorithms.

Table 4.7: Table C2. Negative log-likelihood for larger SL libraries SL2, SL4, SL6 and SL8 based on the *NASH_ nogroup* (a) and *Nash_ pregroup* (b) datasets. N.A. indicates those SL methods do not apply in such cases.

(a) *NASH_ nogroup*

Performance Metric	SL2	SL4	SL6*	SL8**
Negative log-likelihood	N.A.	N.A.	0.281	0.340

(b) *NASH_ pregroup*

Performance Metric	SL2	SL4	SL6*	SL8**
Negative log-likelihood	0.310	0.286	0.283	0.282

* indicates the results are based on the $RR147068$ screening method.
** indicates the results are based on the $RR200050$ screening method.

For the *NASH_nogroup*, *Nash_pregroup* and *Nash_nogroup** datasets, general patterns in the negative log-likelihood (Figure 4.11) were similar to AUC in Figure 4.6. The fluctuation between $RR180056$ and $RR147068$ may be explained by a large standard deviation.

More AUC results from NASH data analysis

Table 4.8: AUC and Running time for smaller SL libraries SL1, SL3, SL5 and SL7 based on the *NASH_nogroup* and *Nash_pregroup* datasets. SL1 included ML algorithms that utilized baseline covariates. SL3 expanded SL1 libraries with the 9 BRR prediction algorithms that utilized claims codes. SL5 included ML algorithms that were coupled with a three-step screening algorithm, whereas the screening algorithm in SL7 were two-step since Bayesian RR step was performed outside the SL. N.A. indicates those SL methods do not apply in such cases.

(a) *NASH_nogroup*

Performance Metric	SL1	SL3	SL5*	SL7**
AUC	N.A.	0.885	0.887	0.901
(95% CI)		(0.874, 0.895)	(0.877, 0.897)	(0.887, 0.915)
Running Time (secs)	N.A.	N.A.	7073.21	3544.80

(b) *NASH_pregroup*

Performance Metric	SL1	SL3	SL5*	SL7**
AUC	0.843	0.888	0.893	0.892
(95% CI)	(0.831, 0.855)	(0.878, 0.898)	(0.882, 0.902)	(0.876, 0.908)
Running Time (secs)	702.57	2194.36	8144.24	5246.27

* indicates the results are based on the $RR147068$ screening method.
*** indicates the results are based on the $RR200050$ screening method.

Table 4.9: Contribution of each algorithm to the final convex combination for SL1, SL3, SL5 and SL7 based on the *NASH_pregroup* dataset. Only algorithms with weights larger than 0.02 are presented. SL5 shows the algorithm contributions based on the BRR *RR*147068 screening process, and SL7 shows the contributions based on the baseline covariates dataset augmented by the *RR*200050 screened covariates.

SL1

Base Learner	Weights
sda_All	0.201
pda_All	0.201
glm_All	0.200
bayesglm_All	0.200

SL3

Base Learner	Weights
BRR.147068_Codes	0.193
glm_screen.baselines	0.175
bayesglm_screen.baselines	0.175
sda_screen.baselines	0.117
pda_screen.baselines	0.046
glm_screen.RR450022	0.117
earth_screen.baselines	0.092
BRR.154065_Codes	0.070
BRR.300033_Codes	0.038
naivebayes_screen.baselines	0.022

SL5

Base Learner	Weights
sda_screen.RR147068	0.563
pda_screen.RR147068	0.276

SL7

Base Learner	Weights
sda_screen.gt300.lasso	0.252
pda_screen.gt300.lasso	0.247
glm_screen.gt300.lasso	0.244
bayesglm_screen.gt300.lasso	0.241

Bibliography

[1] Miika Ahdesmaki et al. *sda: Shrinkage Discriminant Analysis and CAT Score Variable Selection*. R package version 1.3.7. 2015. URL: https://CRAN.R-project.org/package=sda.

[2] Laura B. Balzer et al. "A New Approach to Hierarchical Data Analysis: Targeted Maximum Likelihood Estimation of Cluster-Based Effects Under Interference." In: *ArXiv e-print arXiv:1706.02675* (2017).

[3] Oliver Bembom and Mark J. van der Laan. "A Practical Illustration of the Importance of Realistic Individualized Treatment Rules in Causal Inference". In: *Electronic Journal of Statistics* 1 (2007), pp. 574–596.

[4] Peter J Bickel. *Efficient and adaptive estimation for semiparametric models*. Springer, 1998.

[5] Rhonda C. Boyd et al. "The Impact of Community Violence Exposure on Anxiety in Children of Mothers with Depression". In: *Journal of Child & Adolescent Trauma* 5.4 (2008), pp. 287–300.

[6] Leo Breiman et al. *Classification and Regression Trees, The Wadsworth Statistics/Probability Series*. Wadsworth International Group: Chapman and Hall, New York, 1984.

[7] Jeanne Brooks-Gunn, J. Lawrence Aber, and Greg J Duncan. *Neighborhood Poverty. Context and Consequences for Children. Volume I*. Russell Sage Foundation, 1997.

[8] Peter Bühlmann and Torsten Hothorn. "Boosting Algorithms: Regularization, Prediction and Model Fitting". In: *Statistical Science* 22.4 (2007), pp. 477–505.

[9] Jacob Cohen. "A Coefficient of Agreement for Nominal Scales". In: *Educational and Psychological Measurement* 20.1 (1960), pp. 37–46.

[10] Jeremy R Coyle et al. *sl3: Modern Pipelines for Machine Learning and SuperLearning*. R package version 1.1.0. 2018. URL: https://doi.org/10.5281/zenodo.1342294.

[11] Lorraine Denby and Colin Mallows. "Variations on the Histogram". In: *Journal of Computational and Graphical Statistics* 1 (2009), pp. 21–31.

[12] Stephen Milborrow. Derived et al. *earth: Multivariate Adaptive Regression Splines*. R package version 4.6.3. 2018. URL: https://CRAN.R-project.org/package=earth.

[13] Sandrine Dudoit and Mark J. van der Laan. "Asymptotics of cross-validated risk estimation in estimator selection and performance assessment". In: *Statistical Methodology* 2.2 (2005), pp. 131–154.

[14] Marco Enea. *speedglm: Fitting Linear and Generalized Linear Models to Large Data Sets*. R package version 0.3-2. 2017. URL: `https://CRAN.R-project.org/package=speedglm`.

[15] Jianqing Fan and Yingying Fan. "High-dimensional classification using features annealed independence rules". In: *The Annals of Statistics* 36.6 (2008), pp. 2605–2637.

[16] Jianqing Fan and Runze Li. "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties". In: *Journal of the American Statistical Association* 96.456 (2001), pp. 1348–1360.

[17] Jianqing Fan and Jinchi Lv. "A selective overview of variable selection in high dimensional feature space". In: *Statistica Sinica* 20.1 (2010), pp. 101–148.

[18] C. Ferri, J. Hernández-Orallo, and R. Modroiu. "An experimental comparison of performance measures for classification". In: *Pattern Recognition Letters* 30.1 (2009), pp. 27–38.

[19] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29.5 (2001), pp. 1189–1232.

[20] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1 (2010).

[21] Joseph C. Gardiner, Zhehui Luo, and Lee Anne Roman. "Fixed effects, random effects and GEE: What are the differences?" In: *Statistics in Medicine* 28.2 (2009), pp. 221–239.

[22] Andrew Gelman and Yu-Sung Su. *arm: Data Analysis Using Regression and Multilevel/Hierarchical Models*. R package version 1.10-1. 2018. URL: `https://CRAN.R-project.org/package=arm`.

[23] Susan Gruber and Mark J. van der Laan. "A Targeted Maximum Likelihood Estimator of a Causal Effect on a Bounded Continuous Outcome". In: *The International Journal of Biostatistics* 6.1 (2010).

[24] Susan Gruber and Mark J. van der Laan. "tmle: An R Package for Targeted Maximum Likelihood Estimation". In: *Journal of Statistical Software* 51.13 (2012).

[25] Susan Gruber et al. "Ensemble learning of inverse probability weights for marginal structural modeling in large observational datasets". In: *Statistics in Medicine* 34.1 (2014), pp. 106–117.

[26] David J. Hand. "Measuring classifier performance: a coherent alternative to the area under the ROC curve". In: *Machine Learning* 77.1 (2009), pp. 103–123.

[27] J A Hanley and B J McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1 (1982), pp. 29–36.

[28] Trevor Hastie. *gam: Generalized Additive Models*. R package version 1.14.4. 2017. URL: `https:%20//CRAN.R-project.org/package=gam`.

[29] Trevor Hastie, Robert Tibshirani, and J. H Friedman. *The elements of statistical learning*. Springer, 2001.

[30] Trevor Hastie et al. *mda: Mixture and Flexible Discriminant Analysis*. R package version 0.4-10. 2017. URL: `https://CRAN.R-project.org/package=mda`.

[31] T. Hastie et al. *pamr: Pam: prediction analysis for microarrays*. R package version 1.55. 2014. URL: `https://CRAN.R-project.org/package=pamr`.

[32] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. "Unbiased Recursive Partitioning: A Conditional Inference Framework". In: *Journal of Computational and Graphical Statistics* 15.3 (2006), pp. 651–674.

[33] Dáaz Muñoz Iván and Mark J. van der Laan. "Population Intervention Causal Effects Based on Stochastic Interventions". In: *Biometrics* 68.2 (2011), pp. 541–549.

[34] Dáaz Muñoz Iván and Mark J. van der Laan. "Super Learner Based Conditional Density Estimation with Application to Marginal Structural Models". In: *The International Journal of Biostatistics* 7.1 (2011), pp. 1–20.

[35] Y Jiang, C E Metz, and R M Nishikawa. "A receiver operating characteristic partial area index for highly sensitive diagnostic tests." In: *Radiology* 201.3 (1996), pp. 745–750.

[36] Cheng Ju et al. "Propensity score prediction for electronic healthcare databases using Super Learner and High-dimensional Propensity Score Methods". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* (2016), Working Paper 351.

[37] Jeffrey R. Kling, Jens Ludwig, and Lawrence F. Katz. "Neighborhood Effects on Crime for Female and Male Youth: Evidence From a Randomized Housing Voucher Experiment". In: *Quarterly Journal of Economics* 120.1 (2005), pp. 87–130.

[38] Max Kuhn. "Building Predictive Models in R Using the caret Package". In: *Journal of Statistical Software, Articles* 28.5 (2008), pp. 1–26.

[39] Max Kuhn and Ross Quinlan. *C50: C5.0 Decision Trees and Rule-Based Models*. R package version 0.1.2. 2018. URL: `https://CRAN.R-project.org/package=C50`.

[40] Max Kuhn et al. *caret: Classification and Regression Training*. R package version 6.0-80. 2018. URL: `https://CRAN.R-project.org/package=caret`.

[41] Mark J. van der Laan and Alexander R. Luedtke. "Targeted Learning of the Mean Outcome under an Optimal Dynamic Treatment Rule". In: *Journal of Causal Inference* 3.1 (2015).

[42] Mark J. van der Laan, Eric C Polley, and Alan E. Hubbard. "Super Learner". In: *Statistical Applications in Genetics and Molecular Biology* 6.1 (2007).

[43] Mark J. van der Laan and Dudoit Sandrine. "Unified Cross-Validation Methodology For Selection Among Estimators and a General Cross-Validated Adaptive Epsilon-Net Estimator: Finite Sample Oracle Inequalities and Examples". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* (2003), Working Paper 130.

[44] Mark van der Laan and Susan Gruber. "One-Step Targeted Minimum Loss-based Estimation Based on Universal Least Favorable One-Dimensional Submodels". In: *The International Journal of Biostatistics* 12.1 (2016), pp. 351–378.

[45] Nan M. Laird and James H. Ware. "Random-Effects Models for Longitudinal Data". In: *Biometrics* 38.4 (1982), p. 963.

[46] Erin LeDell. *h2oEnsemble: H2O Ensemble Learning*. R package version 0.1.8. 2016. URL: `https://github.com/h2oai/h2o-3/tree/master/h2o-r/ensemble/h2oEnsemble-package`.

[47] Erin LeDell, Mark J. van der Laan, and Maya Petersen. "AUC-Maximizing Ensembles through Metalearning". In: *The International Journal of Biostatistics* 12.1 (2016), pp. 203–218.

[48] Erin LeDell, Maya Petersen, and Mark van der Laan. "Computationally efficient confidence intervals for cross-validated area under the ROC curve estimates". In: *Electronic Journal of Statistics* 9.1 (2015), pp. 1583–1607.

[49] Erin LeDell, Maya Petersen, and Mark van der Laan. *cvAUC: Cross-Validated Area Under the ROC Curve Confidence Intervals*. R package version 1.1.0. 2014. URL: `https://CRAN.R-project.org/package=cvAUC`.

[50] Kung-Yee Liang and Scott L. Zeger. "Longitudinal Data Analysis Using Generalized Linear Models". In: *Biometrika* 73.1 (1986), p. 13.

[51] Alexander R. Luedtke and Mark J. van der Laan. "Super-Learning of an Optimal Dynamic Treatment Rule". In: *The International Journal of Biostatistics* 12.1 (2016).

[52] Michal Majka. *naivebayes: High Performance Implementation of the Naive Bayes Algorithm*. R package version 0.9.2. 2018. URL: `https://CRAN.R-project.org/package=naivebayes`.

[53] J.Michael Oakes. "The (Mis)estimation of Neighborhood Effects: Causal Inference for a Practicable Social Epidemiology". In: *Social Science & Medicine* 58.10 (2004), pp. 1929–1952.

[54] Judea Pearl. "Causal Diagrams for Empirical Research". In: *Biometrika* 82.4 (1995), pp. 702–710.

[55] Judea Pearl. "Causal inference in statistics: An overview". In: *Statistics Surveys* 3 (2009), pp. 96–146.

[56] Maya L. Petersen and Mark J. van der Laan. "Causal Models and Learning from Data". In: *Epidemiology* 25.3 (2014), pp. 418–426.

[57] Maya L Petersen et al. "Diagnosing and Responding to Violations in the Positivity Assumption". In: *Statistical Methods in Medical Research* 21.1 (2010), pp. 31–54.

[58] Eric C. Polley and Mark J. van der Laan. "Super Learner In Prediction". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* (2010), Working Paper 266.

[59] Eric C Polley et al. *SuperLearner: uper Learner Prediction*. R package version 2.0-22. 2017. URL: `https://CRAN.R-project.org/package=SuperLearner`.

[60] Melanie Prague et al. "Accounting for interactions and complex inter-subject dependency in estimating treatment effect in cluster-randomized trials with missing outcomes". In: *Biometrics* 72.4 (2016), pp. 1066–1077.

[61] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2017. URL: `https://www.R-project.org/`.

[62] Stephen W. Raudenbush and JDouglas Willms. "The Estimation of School Effects". In: *Journal of Educational and Behavioral Statistics* 20.4 (1995), pp. 307–335.

[63] Greg Ridgeway. *gbm: Generalized Boosted Regression Models*. R package version 2.1.3. 2017. URL: `https://CRAN.R-project.org/package=gbm`.

[64] James Robins. "A New Approach to Causal Inference in Mortality Studies with a Sustained Exposure Period - Application to Control of the Healthy Worker Survivor Effect". In: *Mathematical Modelling* 7.9-12 (1986), pp. 1393–1512.

[65] Sebastian Schneeweiss et al. "High-dimensional Propensity Score Adjustment in Studies of Treatment Effects Using Health Care Claims Data". In: *Epidemiology* 20.4 (2009), pp. 512–522.

[66] David W Scott. *Multivariate Density Estimation*. Wiley, 1992.

[67] Oleg Sofrygin and Mark J. van der Laan. "Semi-Parametric Estimation and Inference for the Mean Outcome of the Single Time-Point Intervention in a Causally Connected Population". In: *Journal of Causal Inference* 5.1 (2016).

[68] Oleg Sofrygin and Mark J. van der Laan. *tmlenet: Targeted Maximum Likelihood Estimation for Network Data*. R package version 0.1.0. 2015. URL: `https://CRAN.R-project.org/package=tmlenet`.

[69] Jennifer L. Steele. "Race and General Strain Theory: Examining the Impact of Racial Discrimination and Fear on Adolescent Marijuana and Alcohol Use". In: *Substance Use & Misuse* 51.12 (2016), pp. 1637–1648.

[70] Terry Therneau and Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-13. 2018. URL: `https://CRAN.R-project.org/package=rpart`.

[71] G. Tripepi et al. "Measures of effect: Relative risks, odds ratios, risk difference, and 'number needed to treat'". In: *Kidney International* 72.7 (2007), pp. 789–791.

[72] Jarek Tuszynski. *caTools: Tools: moving window statistics, GIF, Base64, ROC AUC, etc.* R package version 1.17.1.1. 2018. URL: `https://CRAN.R-project.org/package=caTools`.

[73] University of California San Francisco. "Sustainable East Africa Research in Community Health (SEARCH)". In: (2013). URL: `https://clinicaltrials.gov/ct2/show/study/NCT01864603`.

[74] AW. van der Vaart, Sandrine Dudoit, and Mark J. van der Laan. "Oracle inequalities for multi-fold cross validation". In: *Statistics & Decisions* 24 (2006).

[75] Mark J. van der Laan. "Causal Inference for a Population of Causally Connected Units". In: *Journal of Causal Inference* 2.1 (2014).

[76] Mark J. van der Laan. "Estimation of Causal Effects of Community Based Interventions". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* (2010), Working Paper 268.

[77] Mark J. van der Laan, Eric C Polley, and Alan E. Hubbard. "Super Learner". In: *Statistical Applications in Genetics and Molecular Biology* 6.1 (2007).

[78] Mark J. van der Laan, Sherri Rose, and Susan Gruber. "Readings in Targeted Maximum Likelihood Estimation". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* (2009), Working Paper 254.

[79] Mark J. van der Laan and Daniel Rubin. "Targeted Maximum Likelihood Learning". In: *The International Journal of Biostatistics* 2.1 (2006).

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

[81] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S.* Fourth. ISBN 0-387-95457-0. New York: Springer, 2002.

[82] S. D. Walter. "The partial area under the summary ROC curve". In: *Statistics in Medicine* 24.13 (2005), pp. 2025–2040.

[83] Hadley Wickham. "The Split-Apply-Combine Strategy for Data Analysis". In: *Journal of Statistical Software* 40.1 (2011), pp. 1–29.

[84] David H. Wolpert. "Stacked generalization". In: *Neural Networks* 5.2 (1992), pp. 241–259.

[85] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320.