Evaluating uses of machine learning in propensity score estimation on time-to-event data: A simulation study

By

Xiangyu Ji

Master's Thesis
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE
in
Biostatistics

December 18th, 2021
Nashville, Tennessee

Approved:

Amber J. Hackstadt, Ph.D.
Thomas G. Stewart, Ph.D.

Acknowledgments

I sincerely appreciate Dr. Amber Hackstadt's efforts and time throughout this project. Her mentorship and instruction led me through the whole process of research and thesis writing. Her positive energies inspired my confidence in finishing the plans and cleared my worries. I have been and will always be grateful that Dr. Hackstadt directed my thesis work and be glad for the time working with her.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Observational studies with time-to-event outcomes in electronic healthcare records have been widely used to estimate the effects of treatments, exposures, and medical interventions on health outcomes in a typical clinical setting. Observational studies allow researching on target groups that are not typically studied in clinical trials and exploring the effects of harmful exposures that cannot be studied in randomized trials. In randomized trials, randomization can ensure that treated sample will not differ systematically on average from control sample in both measured and unmeasured baseline characteristics. Treatment effect can be estimated by directly comparing outcomes between treatment and control groups. Compared to random clinical trials, the lack of random distribution assignment in observational studies confounds the effects of exposures, due to the potential differences in covariate distributions between two groups. Observational studies could be subject to treatment-selection bias. Thus, the effect of treatment should not be estimated by simply comparing outcomes between treatment and control groups. It is essential to minimize the confounding effects for improvement of internal validity in observation studies with statistical methods such as propensity scores (PS) (Austin & Schuster 2016). Insurance claim data are commonly used in observational studies, which mainly provide diagnosis, prescription, and insurance information. But electronic health records (EHR) offer potential confounding clinical characteristics and laboratory measures that are usually unavailable in insurance claims data, which can improve effect estimation in observation studies (Polsky et al. 2009).

Propensity score is defined as the probability of a subject receiving treatment conditioning on the observed baseline covariates (Rosenbaum & Rubin 1983). It can be considered as a balancing score that attempts to balance the distribution of measured covariates between treatment and control groups (Joffe & Rosenbaum 1999). The distribution of measured baseline covariates between the two groups should be similar within a subset of patients with the same propensity score (Austin 2011). Propensity scores can be used in studies when the treatment assignment is strongly ignorable, which required two conditions (D'Agostino 2007).

One is the assumption that all variables that affect treatment assignment and outcome have been measured in the study and used for PS methods. Including variables that are actually unassociated may slightly increase variance in estimation. But this is acceptable since excluding potentially associated variables can be very costly in terms of bias increasing (Shadish et al. 2008; Stuart 2010). The other condition is that every subject should have a probability larger than 0 of receiving either the treatment or control (Austin 2011). With these two conditions met, propensity score methods can help with obtaining a balanced distribution in covariates between compared groups.

Parametric models are commonly used for PS estimation (e.g., logistic models based on baseline covariates) and would be efficient if models are correctly specified. Variables used in the model should be pretreatment covariates that affect the outcome. But parametric models usually hold strong assumptions for optimal estimation and model misspecification might affect covariate balance in covariates and increase bias in treatment effect estimates. Recent alternative approaches tend to address these issues with data-driven models like machine learning (ML) methods. Gradient boosting model and random forests have been suggested as helpful ML algorithms for PS estimation (Pirracchio2014; Setoguchi et al. 2008: Lee et al. 2010). Super Learner (SL) allow one to apply multiple ML algorithms simultaneously.

Super Learner is a method that chooses the optimal regression algorithm from a given set of candidate algorithms that can include both parametric and data-driven algorithms (Dudoit &van der Laan 2005; van der Laan el al. 2007). The selection of algorithms depends on cross-validation and the choice of a loss function. Then a weighted linear combination of the candidate algorithms will be used as a new estimator, which is called SL estimator. This combination of methods has been demonstrated to perform asymptotically at least as well as the best option in the given set of algorithms, if the set does not contain the true parametric model of the dataset (Dudoit &van der Laan 2005). Despite the potential benefits for PS estimation, applying machine learning methods can be quite time-consuming and may have higher requirements for computing power (van der Laan el al. 2007).

Common ways of utilizing propensity scores include stratifying or subclassifying data based on PS, matching treated subjects with control subjects based on PS, reweighting the subjects with weights derived using the PS, and adjusting the regression model with PS (Stuart 2010). When the outcome data is already available, a drawback of PS matching methods is that

not all data are utilized. Some control subjects are discarded and not used in the analysis even though they might be in the range of the treatment groups' scores. Weighting methods, such as inverse probability of treatment weighting (IPTW), matching weights, and overlap weights, instead can use all subjects in the data (Franklin et al. 2017; Li et al. 2019). Contrasting with the nearest neighbor matching method which assigns each individual a weight of either 0 or 1, weighting methods assign weights between 0 and 1 to individuals (Stuart 2010).

Time to an event of interest in many types of studies, including in pharmaco-epidemiological studies, use EHR data. A challenge in the time-to-event setting is that subjects might be censored before their actual survival status are recorded. Unlike in cross-sectional data, excluding these patients from time-to-event data or simply assuming them alive or dead may seriously bias the treatment effect estimation. Moreover, survival times in time-to-event data are usually skewed, which limits the effectiveness of analysis methods that assume a normal data distribution. Thus, the conclusions for cross-sectional data might not be generalizable to time-to-event data even for studies with settings similar in other conditions. We used the Cox proportional hazard model for data generation and treatment effect estimation to address the features of time-to-event data.

The objective of this study is to implement parametric model (logistic regression models) and data-driven models (machine learning methods) for PS estimation, adjust the effect estimation models with or without weighting based on PS, and evaluate the performances of different combination of PS estimation and application methods on simulated survival data inspired by Right Heart Catheterization (RHC) dataset. We assess each approaches' ability to obtain balanced baseline covariate distributions between treatment and control groups and explore the bias of the treatment effect estimates and the coverage of the corresponding confidence intervals for the time-to-event outcomes. We compare the methods to evaluate when it would be beneficial to utilize more computationally intensive data-driven approaches and when weighted models with logistic-based PS would perform well enough. Some steps further from previous studies on comparing machine learning and the logistic regression method for PS estimation is that we include scenarios with second-order terms in the true PS models and not only first-order terms (Pirracchio 2014). We focus on time-to-event data while previous studies focused on cross-sectional data and applied linear logistic regression models. We use more recent PS weighting methods instead of Inverse Probability Treatment Weighting (IPTW) that

have been typically used as a representative of PS weighting approaches in method comparison studies. We also consider real-world application on a health dataset commonly used for studying PS.

Chapter 2

Methods

2.1 Propensity Score Estimation Methods

2.1.1 Logistic Method

For observational studies, propensity score is commonly estimated through a logistic model: $e(X_i; \beta) = \frac{1}{1+\exp(-X_i\beta)}$. The βs in the model are obtained through logistic regression fitted onto the given dataset so that the model can produce the predicted probability of treatment, $\hat{e}(X_i; \beta)$, as propensity for each subject. In our study, the first element of X_i is assumed to be 1 in the logistic model for notational simplicity.

2.1.2 Machine Learning Method

While propensity score methods have become a standard tool in causal inference, studies showed that minor misspecification of regression models on the PS can lead to substantial bias in the estimates of treatment effect (Kang & Schafer 2007). Traditional approaches to modeling prediction have primarily included parametric models like logistic regression model (Brookhart et al. 2006), which require assumptions that may not be always satisfied in practice. The common application ways of using such methods, like using merely main terms and assuming additivity-only relationship between covariates, also might not provide optimal estimation for PS thus bias the treatment effect estimation. This has led to a growing interest in the use of more adaptive regression techniques to improve the estimation of PS. Machine learning methods, including classification trees, boosting, and random forest, have been developed to overcome the limitations of parametric models by loosening the assumption requirements on pre-specified models (Hastie et al. 2009).

The decision tree method is a supervised machine learning method that can be utilized for regression and classification tasks. It can be defined as a set of rules organized in a hierarchical structure with layers of nodes and branches like a tree, starting from an initial node that represents the whole training dataset. A decision rule is applied to the data at each node to partition the dataset into smaller and more homogenous subsets (Breiman et al. 1984). Observations within each node of the tree, will have similar probabilities of class membership. Each subset can also be split until a convergence criterion is met then the rule stops increasing in

complexity and reach a terminal node (Badillo 2020). Now decision trees are almost never used in machine learning in their original form. Some primary issues with the original decision trees are overfitting and instability (Badillo 2020), which make the rules obtained from the training data not performing well on new data. However, The decision tree method becomes the foundation for two widely used approaches: random decision forests (random forests) and gradient boosting model (GBM) (Badillo 2020). Both random forest method and tree-based gradient boosting method use a set of trained decision trees to predict the outcome. The key difference between the two methods is on how the trees are created. As an approach adapted from the decision tree method, the random forests algorithm constructs many deep decision trees. Although each of those trees is likely overfitted, the overtraining problem can be solved by combining the outputs of multiple trees. The GBM generally creates shallow decision trees and then it can decrease the classification error over time by adding more and more trees (Badillo 2020).

The Random Forest method is a type of machine learning method that constructs a multitude of decision trees at training and can be used for classification in PS estimation. A random vector Θ_k, or the kth, is generated and is independent of the past random vectors Θ_1, …, Θ_{k-1} but all vectors are created using an identical distribution. And a tree is developed using the training set and Θ_k, which leads to a classifier $h(x, \Theta_k)$ as the decision rule for the tree. Input variables for each tree are randomly selected in subsets of fixed size in the training data (Ferri 2020). After a number of trees are generated, the output would be the class selected by most trees or by averaging the results from all trees (Breiman 2001). For the estimation of propensity score based on the training data, random forest algorithm draws multiple random samples from the whole dataset to start many trees (Cham et al. 2016). The data at each node is partitioned into branches below by rules for covariates and this data division process continues until the final node. The propensity score is estimated as the proportion of treated subjects in all subjects retained in a final node. This assigns an estimated score to subjects following one specific path of rules in a tree and each tree will have a whole set of propensity scores assigning to the subjects following each path of rules. The final rules for propensity score estimation will be summarized by averaging the propensity scores sets from all trees generated by the algorithm. To predict a subject's propensity score after setting up the estimation rules, the data for this subject

will go through all trees generated by the algorithm and the propensity score will be estimated by the probability of being voted as a treated subject by all trees.

Gradient boosted modeling (GBM) is a machine learning technique that use gradient boosting to create multiple decision trees (Ferri 2020). Decision trees are referred as classification trees when the predicted outcome is a class or regression trees if the outcome is numerical. Collectively these methods can be referred as Classification and Regression Trees (CART). "Boosting" means combining the performance of many "weak" models in order to produce a more powerful whole model (Friedman et al. 2000). GBM is a prediction model in the form of a collection of rough regression models to improve predictive performances (Freidman 2001). The idea is combining many simple fixed-size models in a forward and stagewise fashion instead of finding one best model for better predictive performances (Elith et al. 2008; Friedman et al. 2000). For propensity score estimation by GBM, an initial regression model is built to roughly fit the whole training data (McCaffrey 2004). Then the algorithm searches for a small adjustment in the form of a small simple model to add to the initial model and improves the whole model. The adjustments do not change previous models and only adds on fitting improvement based on residual points in previous models. The model fitting and adjustment process continues until a chosen loss function is minimized.

Despite the strengths of these machine learning methods compared to parametric methods for PS estimation, these methods have their own limitations so that they can only perform well in rather specific situations and might not fit well across various settings. Super Learner (SL) has been proposed as a method for optimal selection of regression algorithms and selects from a set of candidate algorithms based on cross validation (Dudoit & van der Laan 2005; van der Laan 2006; Sinisi et al. 2007). The selection strategy depends on the choice of a loss function (L2 squared error in this study). Comparison of candidate algorithms' performance relies on V-fold cross validation. SL averages the estimated risks across the validation sets and produces the cross-validated risk for each algorithm. And the weighted linear convex combination of the candidate algorithms will be chosen as the most optimal combination as it has the smallest estimated risk. This combination of candidate algorithms is referred as the SL estimator, which is applied to the whole learning data (van der Laan 2007). In our study, Random Forest (randomForest function in R) (Liaw & Wiener 2002) and GBM (XGBoost function in R) (Chen & Guestrin 2016) were included in the SL library as the candidate algorithms. These two

algorithms were proposed to have better performance in bias reduction and stability in results (Ferri 2020), which fell into our main focus of the performance comparison.

2.2 PS Application Methods: Weighting

The estimated propensity scores can be used in several primary ways to help with estimating the treatment effect, such as stratification or subclassification on PS, PS matching methods, PS adjustment method, and PS weighting method (Austin & Stuart 2015). Many studies have been conducted for the first three types of methods, but performance evaluation needs more discussion on PS weighting methods. We focused on weighting methods where each subject was weighted by w_i, a function of the subject's PS (Stuart 2010). Under weighting methods, a hypothetical population is created with the w_is and ideally should have balanced covariate distributions among the treatment and the control groups. A general class of such weights, w_i, is called balancing weights (Li et al. 2018).

2.2.1 Inverse Probability of Treatment Weighting

Inverse probability of treatment weighting (IPTW) is a weighting scheme widely used for balancing covariates. Using propensity scores that summarize differences in measured sample characteristics, IPTW creates a weighted pseudopopulation in which both treatment group and control group resemble the complete sample (Li et al. 2019). IPTW is defined as $1/\widehat{e_i}$ for treated subjects and $1/(1-\widehat{e_i})$ for control subjects (Austin & Stuart 2015). It inverses the probability of being assigned with treatment of the sample. The target population for this method is the entire study cohort. Some subjects in the nonoverlap area of two groups' PS distribution may receive very large weights in this scheme, resulting in large bias and high variance in treatment effect estimation. Trimming off the nonoverlap regions addresses this issue but may discard subjects with outcome events and increase the variance. Truncating the subjects with weights outside of a certain percentile range (rth to $(1-r)$th percentile) may help reduce the influence of such type of extreme weights while also utilizing the whole sample in the analysis (Cole & Herman 2008).

2.2.2 Matching Weights

Matching weights is an alternative weighting method to limit the impact of extreme weights from nonoverlap in the PS and improve covariate balance by treating the same estimand as pair matching on the PS (Li & Greene 2013). The matching weight is defined as $w_i =$

$\frac{min(e_i,1-e_i)}{e_i}$ for treatment group and $w_i = \frac{min(e_i,1-e_i)}{1-e_i}$ for control group. This method downweights overabundant subjects in the sample, such as treated subjects with high PSs and control subjects with low PS. In contrast, IPTW upweights the underestimated subjects in the sample like treated subjects with low PSs and control subjects with high PSs. So those subjects that would receive very high weights under IPTW approach will receive at most 1 as their weights under matching weights approach. And no patients will be excluded from the sample although they might be downweighted.

2.2.3 Overlap weights

Overlap weight is a newer weighting method developed for better balancing function and addresses some of the issues of IPTW (Li et al. 2019). The overlap weight is defined as $w_i = 1 - \widehat{e_i}$ for treatment group and $w_i = \widehat{e_i}$ for control group, which is the probability of a patient being assigned to the opposite group. Overlap weight upweights subjects having substantial probability of receiving treatment and downweights the subjects in the tails of the PS distribution. Therefore, subjects with PS of 0.5 would make the largest contribution to the effect estimation and those with PS close to 0 or 1 would make smallest contribution. It targets on the population with the most overlap in their observed covariates.

2.2.4 Comparison of different weighting methods

Compared to stratification and subclassification on PS, the baseline covariates of treated and control samples can be easily described and presented under IPTW. With the assumption that all of the important confounders are included in the dataset, IPTW may be a convenient PS estimation for its simplicity and alignment with the ideal scenario where the entire sample, rather than subsets, had been randomized to the intervention of interest. However, IPTW might be more sensitive to misspecification of PS estimation model (Deb 2016). And it may perform poorly when the treatment group and the control group are initially very different or when some patients have extreme PS near 0 or 1, which means a subject might always receive treatment or always in control group in the model. Using IPTW method on sample with such patients might lead to larger bias and variance in the estimated treatment effect. Extreme propensities are common in large datasets where inclusion criteria could be broadly defined. Although trimming methods have been suggested as an improvement approach by excluding patients with extreme

propensities, the decision rule for trimming methods might be unclear and controversial and they can result in substantial sample size reduction (Li et al. 2019).

Matching weight is an alternative weighting method that can limit the influence of subjects with extreme propensity scores (Franklin et al. 2017). It can confer numerical stability compared with IPTW by focusing on treatment effects in subjects with good overlap on the propensity score between treatment and control groups (Yoshida 2017). Compared to IPTW, overlap weighting method is a newer PS weighting method meant for better performance in balance and precision. These weights are bounded in a rational range (0-1) and thus substantially reduce the influence of subjects at tails of the PS distribution without removing them from the sample. Overlap weighting method may also minimize the large-sample variance of treatment effect estimator (Li et al. 2019).

2.3 Overall simulation structure

We performed a set of Monte-Carlo simulation experiments with simulated data inspired by the Right Heart Catheterization (RHC) dataset (Connors et al. 1996) to examine the performance of different propensity score estimation and application methods on estimating the treatment effect for the time-to-event outcomes and improving covariate balance. The data were simulated as a hypothetical cohort study with a binary treatment A, a time-to-event outcome and eleven covariates W_is. We had a similar survival time distribution (until 90 days) and incidence rate as the RHC dataset. Datasets were generated 1,000 times for four simulation scenarios which differed in the true association model between covariates, treatment and outcome. Propensity scores were estimated with the traditional logistic regression model and machine learning methods. We then explore the performance of two recently proposed propensity score weighting approaches, matching weight and overlap weight methods, in estimating the treatment effect in the time-to-event model. In each scenario, the performance will be compared using unadjusted and adjusted models with or without weights.

2.3.1 Simulation Scenarios

In practice, researchers do not know the true structure of the association between treatment and covariates and similarly, the association between the outcome and the covariates. Thus, they might fit a mis-specified model by assuming linear and/or additive relationships and ignoring potential interaction or quadratic terms in the true model. To compare the performance

of different methods when using mis-specified models, we designed four scenarios differing in the complexity of the associations between the treatment/outcome and the covariates, or the "true models". The complexity varied in the degree of non-linearity and/or non-additivity of modeled associations between the covariates across the models. In each scenario, the relationship between covariates in the true treatment model and the true outcome model was the same.

The designed four scenarios had the following properties:

- Scenario A: additivity and linearity (main effects only);
- Scenario B: non-linearity (main effects + 3 quadratic terms);
- Scenario C: non-additivity (main effects + 10 two-way interaction terms);
- Scenario D: non-additivity and nonlinearity (main effects + 10 two-way interaction terms + 3 quadratic terms).

The Cox PH models and the PS estimation models in application were additive and linear only between the covariates across all scenarios. The true model in Scenario A were the same with the applied model for PS estimation and treatment effect estimation model. The true models in Scenario B, C, and D were different from the treatment effect estimation models applied onto the simulated datasets. Therefore, the applied models in Scenario B, C, and D were incorrect and the deviation from the true models increased from Scenario B to D.

2.3.2 Data Simulation

The motivating Right Heart Catheterization (RHC) dataset was obtained from a prospective cohort study called Study to Understand Prognoses and Preferences for Outcomes and Risks of Treatments (SUPPORT) that examined the association between the use of right heart catheterization during the first 24 hours of care in the intensive care unit (ICU) and subsequent survival, intensity of care, length of stay, and cost of care (Connors *et al*. 1996). The study was operated between 1989 and 1994 in five US teaching hospitals. The study sample was 5735 critically ill adult patients receiving ICU care for one of the nine prespecified disease categories. The main outcomes were death status, survival time, intensity of care, cost of care, length of stay in the ICU and hospital and these outcome measurements were determined from the clinical record and from the National Death Index. In Connor's study (1996), propensity scores for this dataset were constructed using multivariable logistic regression. The association of right heart catheterization treatment with specific outcomes were analyzed with case-matching and multivariable regression modeling after adjusting for treatment selection with the propensity

scores. Sensitivity analysis was used to estimate the potential effect of missing covariates on the results.

We used the RHC dataset as the inspiration for generating covariates, treatment variable, and outcome variable in the simulated datasets. The RHC dataset had eight covariates that were relatively impactful based on the Cox PH model we fitted to it, which were PaO2/FIO2 ratio ("pafi1"), age ("age"), heart rate ("hrt1"), respiratory rate ("resp1"), bilirubin level ("bili1"), do-not-resuscitate status ("dnr1"), cardiovascular diagnosis ("card") and medical insurance type ("ninsclas"). The simulated datasets in our study had eleven baseline covariates which were generated to have influence on both treatment selection and the outcome. We simulated five continuous covariates from independent standard normal distributions and six binary covariates were simulated from independent binomial distribution to have similar incidence rates as the selected variables in the RHC dataset. Each binary covariate in the simulated datasets represented a nominal value of selected categorical covariates in the RHC dataset.

For the ith subject in a dataset, the probability of the treatment was estimated from one of the four true PS models which were four logistic models each corresponding to a scenario (Scenario A - D). The coefficients of the models were obtained from fitting the same models to the RHC dataset and then inflating some coefficient values to ensure that each covariate included in the model had an impact on treatment selection. The coefficient for the quadratic term of age variable was set to be 0.01 if its absolute value from model fitting was smaller than 0.01. The coefficients for the interactions term related to cardiovascular diagnosis variable ("card") was set to be 0.1 if the original values from model fitting was smaller than 0.1. All other coefficients that had absolute values smaller than 0.001 were set to 0.001. The formulas used for simulating the probability of treatment are given in Appendix D. And for each subject, treatment status was generated from a Bernoulli distribution with subject-specific parameters p_i: $A_i \sim \mathrm{Be}(p_i)$, where $p_i = \frac{\exp(X^T\beta)}{1+\exp(X^T\beta)}$, X^T is the input matrix containing the generated covariate columns and the columns for the specified interaction or quadratic terms, and β is the coefficients for the treatment model.

We generated an observed time-to-event outcome for each subject using a Cox proportional hazards model with an exponential baseline hazard, the actual treatment assigned and simulated covariates (Morina & Navarro 2014; Bender, Augustin, & Blettner 2005). The coefficients used to simulate the time-to-event outcome were obtained from the Cox proportional

hazards model that were same in relation structure between covariates with the corresponding treatment models in each scenario. There were four sets of coefficients, each for one of the four scenarios. The true treatment effect and the formulas for simulating the time-to-event outcomes are given in Appendix D. When generating the outcome status, the generated survival times were censored at 90 as a representative of 90 days in order to make the simulated time distribution similar to the survival time distrribution of the RHC dataset.

We designed the following factors to vary within each scenario: (1) PS estimation: the propensity scores were generated with the logistic method or the machine learning method (Super Learner); (2) whether adjusting for baseline covariates besides treatment in the treatment effect estimation model, (3) whether using weights in the treatment effect estimation model; (4) PS application: the propensity score were applied as weights through the matching weight approach or the overlap weight approach. Therefore, there were ten cases (Case 1 - 10) for each scenario, five for the adjusted models (no weights, matching weights calculated using logistic regression, matching weights calculated using the Super Learner, overlap weights calculated using logistic regression, and overlap weights calculated using the Super Learner) and five for the unadjusted models (no weights, matching weights calculated using logistic regression, matching weights calculated using the Super Learner, overlap weights using logistic regression, and overlap weights calculated using the Super Learner). In each scenario, we simulated 1000 datasets at three sample sizes: n=100, 500, or 1000 subjects.

2.4 Performance Metrics

We measured and compared the performance of the method combinations in the ten cases under each scenario through the following metrics:

- The performance of point estimate of the treatment effect: bias, mean squared errors (MSE), and rooted mean squared errors (RMSE). The bias was reported as absolute bias.

$$\text{Bias} = \frac{1}{N}\sum_{i=1}^{N}\left(Y - \widehat{Y}_i\right)$$

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(Y - \widehat{Y}_i\right)^2$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}\left(Y - \widehat{Y}_i\right)^2}{N}}$$

Y is the true hazard ratio of treatment from the Cox PH model fitted onto the RHC dataset. $\widehat{Y}_i$ is the estimated hazard ratio of treatment fitted onto the ith simulated dataset. N is the number of simulations.

- The performance of variance estimators: 95% confidence interval coverage and width.
  If the true hazard ratio Y is within $\left[exp\left(log\left(\widehat{Y}_i\right) - t_c s_i/\sqrt{n}\right), exp\left(log\left(\widehat{Y}_i\right) + t_c s_i/\sqrt{n}\right)\right]$, then it is counted as covered for once. $t_c = t_{0.975,n-1}$ is the critical value of the t statistic with the significance of 0.05 and n-1 degrees of freedom. $s_i/\sqrt{(n)}$ is the standard error of $\log\left(\widehat{Y}_i\right)$, where s_i is the sample standard deviation for $\log\left(\widehat{Y}_i\right)$. n is the sample size of that dataset.
  Confidence interval width: $2\ t_c s/\sqrt{n}$
- The balance in the covariates between treated and control subjects: the average standardized absolute mean difference (ASAM). ASAM of 0.1 or more were considered to be of concern (Austin 2009).

$$ASAM = \frac{1}{n_c}\sum_{j=1}^{n_c}\frac{M_1 - M_2}{SD_{pooled}}$$

$M_1$and M_2 are the means of one covariate in the treatment group and in the control group. SD_{pooled} is the pooled standard deviation for the covariate values in the total sample. n_c is the number of covariates in the datasets. Note that this equation is the ASAM for one simulated dataset rather than across the 1,000 simulated datasets.

All performance metrics used for method performance comparison were the average of the metrics across the 1,000 simulated data sets.

2.5 Application on Real-Word dataset

Right Heart Catheterization dataset (introduced in section 2.3.2) has been widely used as a biostatistical dataset for illustrating and testing propensity score methods. It has a relatively high outcome event rate while many studies on propensity score methods use datasets with rare outcomes. We applied the methods of interest to the RHC dataset to evaluate the performance of

PS estimation methods, PS weighting methods, and covariate adjustment on a treatment effect estimation model in terms of 95% CI width and covariate balance. The patterns of method performance in the RHC dataset would be compared with those found in the simulated datasets as a partial examination on the findings from the simulated results.

We applied the same methods used in the ten cases under the simulation setting to the RHC dataset, including covariate adjustment in treatment effect estimation model or using weights in the model, PS estimation methods (logistic- and SL-based), and PS application methods (matching weight and overlap weight).

All analyses were performed in R statistical software version 4.0.2 (R Foundation for Statistical Computing, Vienna, Austria), running on a Windows 10 x64 platform. R codes were provided in the Appendix.

Chapter 3

Results

The simulation results obtained for the four scenarios are presented in Table 1.

3.1 The Cox PH model

Under Scenario A, the true model on the association between treatment and covariates was linear and additive so the adjusted Cox PH model when estimating the treatment effect estimation in the simulated datasets was correctly specified. The true models under Scenario B, C, and D had nonlinear, nonadditive, or both features so the linear additive models used in the propensity score estimation and covariate adjustments were incorrectly specified in these scenarios. Three combinations of adjusting and weighting methods, or three types of models, were compared in performances: unadjusted weighted models, adjusted unweighted models, and adjusted unweighted models. Unadjusted weighted models can estimate marginal treatment effects with covariate distribution balance addressed. Adjusted unweighted can be used for conditional effects estimation. Adjusted weighted models might be helpful if people are interested in utilizing the information in the data in both adjusting and weighting approaches.

Since this study focuses on clinical studies using EHR data often having larger sample sizes, the discussion of the results will be initially on the simulations of sample size 1000 and then makes comparison with the simulations of the smaller sample size (i.e., n=100).

3.2 Performance of Treatment Effect Estimator

The performance of treatment effect estimator was measured by bias, mean squared errors (MSE), and rooted mean squared errors (RMSE). Overall, adjusting appeared to be more beneficial than weighting towards the performance of the treatment effect estimator at sample size of 1000 (Figure 1). The SL-based PS estimation method performed similarly with the logistic-based method. Across the scenarios, misspecification of PS model appeared to have larger influence on bias when the true PS model became more complex, which means the bias difference between the three types of models increased for scenarios with more complex true models (Table 1).

Table 1. Simulation Results Obtained under 4 Scenarios with Different Propensity Score Estimation and Application Methods (n=1000)

True model ("Scenarios")	Outcome Model	PS Estimation Model	PS Weighting method	Bias	MSE	RMSE	% of CI coverage	CI width	ASAM
A. Additivity and linearity (Main effects only)	Adjusted	Unweighted	/	0.0058	0.0117	0.1084	94.4	0.417	0.144
		Logit	Matching	0.0057	0.0128	0.1130	94.3	0.440	0.007
		SL	Matching	0.0049	0.0155	0.1243	94.8	0.481	0.029
		Logit	Overlapping	0.0055	0.0129	0.1134	94.6	0.439	0.000
		SL	Overlapping	0.0063	0.0157	0.1251	94.9	0.483	0.043
	Unadjusted	Unweighted	/	-0.0769	0.0151	0.1229	85.5	0.372	0.144
		Logit	Matching	-0.0734	0.0153	0.1236	89.0	0.394	0.007
		SL	Matching	-0.0661	0.0168	0.1298	90.3	0.441	0.029
		Logit	Overlapping	-0.0826	0.0168	0.1297	85.8	0.391	0.000
		SL	Overlapping	-0.0789	0.0188	0.1370	87.9	0.439	0.043
B. Nonlinearity (Main effects + quadratic terms)	Adjusted	Unweighted	/	0.0122	0.0121	0.1099	95.3	0.426	0.152
		Logit	Matching	0.0152	0.0151	0.1228	94.3	0.466	0.008
		SL	Matching	0.0179	0.0173	0.1314	93.9	0.504	0.028
		Logit	Overlapping	0.0145	0.0149	0.1222	94.3	0.464	0.000
		SL	Overlapping	0.0158	0.0173	0.1315	94.2	0.504	0.042
	Unadjusted	Unweighted	/	-0.0622	0.0130	0.1140	90.2	0.380	0.152
		Logit	Matching	-0.0481	0.0137	0.1171	92.7	0.421	0.008
		SL	Matching	-0.0535	0.0161	0.1270	93.2	0.459	0.028
		Logit	Overlapping	-0.0563	0.0143	0.1197	92.1	0.416	0.000
		SL	Overlapping	-0.0581	0.0167	0.1293	92.8	0.460	0.042
C. Non-additivity (Main effects + interaction terms)	Adjusted	Unweighted	/	0.0096	0.01274	0.1129	95.3	0.439	0.229
		Logit	Matching	0.0114	0.01581	0.1258	95.1	0.479	0.010
		SL	Matching	0.0118	0.01629	0.1276	95.4	0.500	0.042
		Logit	Overlapping	0.0115	0.01624	0.1275	94.3	0.483	0.000
		SL	Overlapping	0.0126	0.01651	0.1285	95.4	0.504	0.044
	Unadjusted	Unweighted	/	0.0783	0.01623	0.1274	88.8	0.399	0.229
		Logit	Matching	0.0579	0.01585	0.1259	92.0	0.441	0.010
		SL	Matching	0.0545	0.01668	0.1292	92.4	0.474	0.042
		Logit	Overlapping	0.0744	0.01899	0.1378	90.2	0.451	0.000
		SL	Overlapping	0.0776	0.02105	0.1451	90.0	0.488	0.044
D. Non-additivity and nonlinearity (Main effects+ quadratic term + interaction term)	Adjusted	Unweighted	/	0.0173	0.0122	0.1106	95.5	0.435	0.188
		Logit	Matching	0.0191	0.0143	0.1195	95.1	0.476	0.008
		SL	Matching	0.0186	0.0159	0.1262	94.7	0.496	0.043
		Logit	Overlapping	0.0191	0.0145	0.1206	95.0	0.478	0.000
		SL	Overlapping	0.0192	0.0160	0.1264	95.0	0.499	0.044
	Unadjusted	Unweighted	/	-0.0936	0.0161	0.1270	81.5	0.345	0.188
		Logit	Matching	-0.0822	0.0151	0.1229	87.7	0.388	0.008
		SL	Matching	-0.0861	0.0179	0.1337	88.0	0.422	0.043
		Logit	Overlapping	-0.0907	0.0172	0.1311	85.6	0.388	0.000
		SL	Overlapping	-0.0900	0.0190	0.1377	86.9	0.425	0.044

3.2.1 Weighting without Covariate Adjustment

The bias increased as the true model became more complex (Table 1). For the PS matching weights method with no covariate adjustment, the bias of logistic-based estimator was slightly larger than SL-based estimator when the applied model was correct. The bias associated with SL-based estimators were similar to logistic-based estimators for the scenarios when the PS model was non-additive or both non-additive and non-linear. The biases of logistic-based estimators did not show substantial difference from SL-based estimators in general when using PS matching weights without covariate adjustment (Table 1).

For the PS overlap weight method, the bias associated with logistic-based was slightly higher than SL-based estimator for the non-linear true model (Table 1). The biases of the logistic-based estimators were similar to those of the SL-based estimators across all scenarios when using PS overlap weights without covariate adjustment.

The matching weight method tended to have smaller biases than the overlap weight method. The biases for both nonlinear and nonadditive true models (Scenario D) were generally higher than those for the true models under other scenarios (Scenario A, B, C).

3.2.2 Covariate Adjustment without Weighting

When using covariate adjustment but not weighting, the bias was relatively smaller than in unadjusted weighted models even for the scenarios when the PS model was not correct (Table 1). The differences across the scenarios were negligible.

3.2.3 Covariate Adjustment with Weighting

For the PS matching weight method, the bias associated with logistic-based estimators were lower when the applied PS model was correctly specified than the bias in other scenarios (Table 1). Similar patterns were observed with SL-based estimators (Table 1). The biases of logistic-based estimators were slightly higher than those of SL-based estimators in general when adjusting covariates and using PS matching weights.

For the PS overlap weight method, the bias associated with both logistic-based and SL-based estimators were lower when the applied model was correctly specified (Scenario A) than other scenarios (Table 1). The logistic-based estimators produced bias slightly lower with the SL-based estimators across all cases when adjusting covariates and using PS overlap weights.

The matching weight method tended to have similar scales of biases than the overlap weight method. The biases of correct applied models (Scenario A) might be lower than Scenario B, C, and D.

3.2.4 Comparison between the Three Adjusting/Weighting Situations

When the sample size was 1000, adjusted unweighted models were associated with the smallest biases, followed by adjusted weighted models then unadjusted weighted models. Adjusting for covariates with or without PS weights substantially improved the bias reduction compared to not using covariate adjustment (Table 1). Thus, unless the investigator is interested in marginal effects, adjusting for covariates tends to reduce bias better than not adjusting in studies with larger sample sizes.

3.3 Performance of variance estimator

The performance of variance estimator was evaluated by 95% confidence interval coverage and CI width. At sample size of 1000, models with covariate adjustment had higher 95% CI coverage rates than unadjusted models. SL-based estimators have better 95% CI coverage rates and wider CI than logistic-based estimators and adjusting seemed more beneficial than weighting (Table 1).

3.3.1 Weighting without covariate adjustment

For the PS matching weight method, the coverage rates of 95% confidence intervals (CI) associated with logistic-based estimators when the true PS model was both nonlinear and nonadditive were lower than the nominal level for other scenarios (Table 1). A similar pattern was observed for SL-based estimators (Table 1). The CI coverage rates and CI width for SL-based estimators were slightly better than logistic-based estimators when using PS matching weights without covariate adjustment (Table 1).

For the PS overlap weight method, the coverage rates of 95% CI associated with both logistic-based and SL-based estimators were lower when the true model was both nonlinear and nonadditive than the nominal level of other scenarios (Table 1). The CI coverage rates of SL-based estimators performed similar with those of logistic-based across the scenarios when using PS overlap weights without covariate adjustment.

The CI coverage rates for weighted outcome models without covariate adjustment were generally not ideal (around 90% or lower). The matching weight method produced better CI

coverage rates than the overlap weight method for both logistic-based and SL-based estimator. The CI width did not clearly differ between the two PS weighting methods. Weighting did not provide nominal CI coverage rates based on the results of unadjusted weighted models, even when the PS model was correct and the covariate balance was good.

3.3.2 Covariate adjustment without weighting

The 95% CI coverage rates were similar across different scenarios. The CI coverage rate for correct PS model (Scenario A) was slightly lower than 95% and the nominal rate for other scenarios were slightly higher than 95% (Table 1).

3.3.3 Covariate adjustment with weighting

For both matching and overlap weight methods, the 95% CI coverage rates did not clearly differ between logistic-based and SL-based estimators and were around 95%. There was also no clear difference between the rates from matching and overlap weight methods.

3.3.4 Comparison between the three adjusting/weighting situations

Among the three cases at sample size of 1000, adjusted unweighted models gave the highest rates of CI coverage and slightly narrower CI than adjusted weighted models. Adjusted unweighted models' and adjusted weighted models' performances were similar to each other compared and both obtained better coverage than the unadjusted weighted. The CI coverage rates with covariate adjustment but no weighting were close to 95%. The CI coverage rates for models with neither covariate adjusting nor weighting were generally lower than 90%.

3.4 Balance diagnosis

Propensity score weighting aims to reduce bias in estimates by obtaining more balanced treatment groups, in terms of observed covariates. The balance of covariate distribution was diagnosed with the average standardized absolute mean difference (ASAM). An ASAM of 0.1 or more indicates that the two groups have poor balance (Austin 2009). The ASAM was solely determined by the weights used in the applied models since it was related to how balanced the variable distributions were in the sample and not related to what models would be applied to the sample. So, the comparison was made between weighted and unweighted models and the three combinations of weighting/adjusting were not applicable here.

3.4.1 When the applied model was weighted

For the PS matching weight method, the ASAM values associated with logistic-based PS were smaller than the SL-based PS. The ASAM values were similar across the four scenarios for each PS estimation methods (Table 1). Although SL-based PS had slightly larger ASAM, the difference was not substantial considering that ASAM were not of concern when less than 0.1.

The PS overlap weight method had similar trends for ASAM with the matching weight method. And the differences between the two types of weighting methods were not considerable since they were both less than 0.1. But overlap weight method did produce nearly complete covariate balance when combined with logistic PS estimation method, as Li (2019) claimed.

The ASAM of adjusted unweighted models were not negligible (Table 1). When the true model was nonadditive, the ASAM values tended to be the biggest among all the scenarios, which were even bigger than those of nonlinear and nonadditive models (Table 1). In general, weighted models had better ASAM than unweighted models.

Table 2. Simulation Results Obtained under 4 Scenarios with Different Propensity Score Estimation and Application Methods (n=100)

True model ("Scenarios")	Outcome Model	PS Estimation Model	PS Weighting method	Bias	MSE	RMSE	% of CI coverage	CI width	ASAM
A. Additivity and linearity (Main effects only)	Adjusted	Unweighted	/	0.1006	0.2117	0.4601	92.8	1.652	0.214
		Logit	Matching	0.1238	0.2677	0.5174	92.3	1.834	0.028
		SL	Matching	0.0992	0.2715	0.5211	91.4	1.836	0.151
		Logit	Overlapping	0.1202	0.2609	0.5108	91.8	1.830	0.000
		SL	Overlapping	0.1113	0.2834	0.5323	91.7	1.859	0.156
	Unadjusted	Unweighted	/	-0.0328	0.1009	0.3177	94.8	1.289	0.214
		Logit	Matching	-0.0136	0.1223	0.3497	94.3	1.426	0.028
		SL	Matching	-0.0178	0.1332	0.3650	94.2	1.453	0.151
		Logit	Overlapping	-0.0264	0.1241	0.3522	94.3	1.421	0.000
		SL	Overlapping	-0.0232	0.1367	0.3698	94.3	1.452	0.156
B. Nonlinearity (Main effects + quadratic terms)	Adjusted	Unweighted	/	0.1179	0.2180	0.4669	93.3	1.705	0.221
		Logit	Matching	0.1413	0.3021	0.5496	92.5	1.988	0.030
		SL	Matching	0.2290	0.4245	0.6516	90.8	2.125	0.158
		Logit	Overlapping	0.1394	0.2906	0.5391	92.4	1.971	0.000
		SL	Overlapping	0.1998	0.3858	0.6211	91.2	2.070	0.163
	Unadjusted	Unweighted	/	-0.0104	0.1115	0.3339	94.7	1.313	0.221
		Logit	Matching	0.0194	0.1497	0.3869	95.1	1.525	0.030
		SL	Matching	0.0065	0.1572	0.3964	94.0	1.506	0.158
		Logit	Overlapping	0.0111	0.1505	0.3879	94.7	1.516	0.000
		SL	Overlapping	0.0090	0.1580	0.3975	94.4	1.512	0.163
C. Non-additivity (Main effects + interaction terms)	Adjusted	Unweighted	/	0.1231	0.2407	0.4907	93.4	1.782	0.298
		Logit	Matching	0.1688	0.4295	0.6553	93.4	2.102	0.035
		SL	Matching	0.1330	0.3476	0.5896	91.4	2.085	0.169
		Logit	Overlapping	0.1656	0.4335	0.6584	92.8	2.127	0.000
		SL	Overlapping	0.1444	0.3559	0.5966	91.4	2.117	0.185
	Unadjusted	Unweighted	/	0.1301	0.1371	0.3702	94.4	1.383	0.298
		Logit	Matching	0.1220	0.1718	0.4145	95.8	1.600	0.035
		SL	Matching	0.1356	0.1823	0.4269	94.8	1.652	0.169
		Logit	Overlapping	0.1363	0.1925	0.4388	95.0	1.641	0.000
		SL	Overlapping	0.1461	0.1923	0.4386	94.5	1.675	0.185
D. Non-additivity and nonlinearity (Main effects+ quadratic term + interaction term)	Adjusted	Unweighted	/	0.1449	0.2651	0.5149	92.0	1.798	0.262
		Logit	Matching	0.1942	0.4269	0.6534	90.3	2.082	0.032
		SL	Matching	0.2077	0.4222	0.6497	90.5	2.172	0.142
		Logit	Overlapping	0.1964	0.4209	0.6488	90.1	2.103	0.000
		SL	Overlapping	0.1928	0.4083	0.6389	91.1	2.164	0.157
	Unadjusted	Unweighted	/	-0.0432	0.0974	0.3121	93.1	1.187	0.262
		Logit	Matching	-0.0058	0.1310	0.3619	94.3	1.405	0.032
		SL	Matching	-0.0136	0.1512	0.3888	93.1	1.457	0.142
		Logit	Overlapping	-0.0127	0.1405	0.3748	93.4	1.416	0.000
		SL	Overlapping	-0.0127	0.1560	0.3949	92.4	1.467	0.157

3.5 **Large and Small Sample Sizes**

Besides simulating for sample sizes of 1000, we also tested the simulations for sample size of 500 and 100 to observe the impact of smaller sample sizes. For simulations at all three sample sizes, the functions for machine learning methods in Super Learner library were used with the default parameters.

The simulations results held a similar general pattern for logistic- or SL-based PS estimation methods and matching or overlap weight methods at sample size of 100 with 1000 (Table 2). SL-based estimator in general had similar performance with the logistic-based estimator in terms of bias (Table 2). Although the SL-based estimator did have smaller or larger biases than the logistic-based estimator, the differences were within the Monte Carlo simulation error. Simulation results at a sample size of 500 were very similar to those at 1000 (Table 3).

However, choosing adjusting or weighting in the treatment effect estimation model seemed to have reverse performance in biases at small sample compared to the bias at large sample. When the sample size was smaller (n=100), adjusted models had larger biases than unadjusted models whether the applied model was correct or not. Unadjusted weighted models performed best and were followed by adjusted unweighted models then adjusted weighted models (Figure 1 and Table 2). In contrast, adjusting obtains estimates with relatively small bias when the sample size is larger (n=500 or n=1000), compared to using PS weights in the treatment effect estimation model. This pattern of adjusting outperforms weighting for bias at larger sample sizes still exist even when adjusting by a model that is not correctly specified. One note is that the bias of unadjusted weighted models substantially increased compared to other scenarios but generally were similar to the bias of adjusted models at scenario C. This observation might be specific to our study setting since we had ten interaction terms for scenarios with non-linear condition and the coefficients for interaction terms related to “card” variable were inflated to 0.1.

When the sample size was smaller, adjusted models had lower 95% CI coverage rates and wider CI than unadjusted models especially when the true model became more complex. The coverage rates for adjusted models at smaller sample size were around 92% and were lower for scenarios of using incorrect PS models than scenarios using correct models (Table 2). And the SL-based estimator generally had slightly lower CI coverage rates and wider CI. For both CI

coverage and CI width, unadjusted weighted models performed better than adjusted unweighted models than adjusted weighted models.

For the sample size of 100, the ASAM values associated with the SL-based PS estimation method indicated poor balance but ASAM values for the logistic method were still not concerning (Table 2). Overlap weight method using logistic-based PS yielded best covariate balance. Unweighted models still performed worse in ASAM than weighted models with SL-based PS.

3.6 **Application on RHC Dataset**

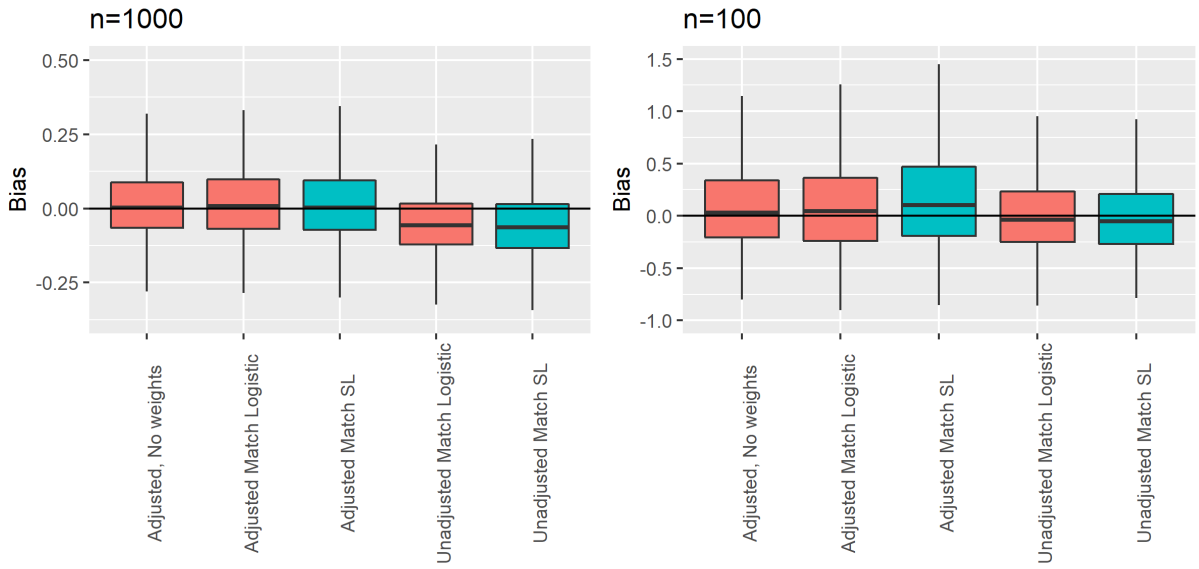
For the method application on the RHC data, the 95% CI width for treatment effect from unadjusted weighted models was the narrowest, followed by adjusted unweighted models then adjusted weighted models. The ASAM from weighted models were smaller than that from unweighted models. Logistic-based PS used in weighting produced slightly smaller ASAM than SL-based PS. But the difference was not concerning as all ASAMs from weighted models were smaller than 0.1 (Table 4). Since the RHC dataset is a real-world dataset, the true PS model is unknown. The CI width and ASAM patterns in the RHC dataset were consistent to the patterns found in the results from the simulated datasets at a sample size of 1000. The treatment effect estimates were similar for all methods and both the logistic and super learner propensity score estimates improved balance between the treatment groups. This was expected since the RHC dataset has 5735 subjects which was relatively large in our simulation setting. Regardless of the analysis approach, like previous studies, the estimate of the hazard ratio and the Kaplan Meier plot suggest that right heart catherization increases the risk of death (Figure 4 and Table 4).

Chapter 4

Conclusion

4.1 Our Findings

Figure 1. Bias of SL-based and Logistic-based Estimator in Scenario B at n=1000 and n=100



The primary goal of our study was to explore the performance of machine learning and logistic regression approaches for PS estimation along with two different weighting techniques through bias of treatment effect, 95% CI coverage and CI width, and ASAM for health studies. Some comparisons of interest include adjusting for covariates only, weighting without covariate adjustment and weighting with covariate adjustment in the treatment effect estimation. At a sample size of 1000, adjusting performed better than weighting in terms of reducing bias for the treatment effect estimator, even if the covariate adjustment model was not correctly specified. For n =100, weighting reduced bias better than adjusting by either correctly or incorrectly specified covariate models. Unadjusted weighted models worked best for treatment effect estimation, followed by adjusted weighted and unadjusted weighted models. SL-based PS had similar performance with logistic-based PS in terms of bias, even when the logistic model was not correctly specified. For variance estimation, SL performed better than the logistic method in a large sample but worse in a small sample. The matching weight approach had slightly better CI performance than overlap weight method for both SL- and logistic-based PS in a large sample but these two methods had similar performance in a small sample. At n=1000 or 500, adjusted

weighted models performed better in CI coverage at the cost of wider CI than adjusted unweighted, followed by unadjusted weighted models. At n=100, unadjusted weighted models had best performance in CI coverage and CI width than adjusted weighted models than adjusted unweighted models. For covariate balance, weighted models had smaller ASAM values. Using SL or logistic method for PS estimation did not have an obvious impact as long as weighting was used in the applied model, although the values of ASAM were smaller when estimating PS with logistic method.

Our study showed that covariate adjustment reduced bias in larger samples and weighting reduced bias in smaller samples, which were consistent with previous studies. Hirano & Imbens (2001) used the RHC dataset to compare the treatment effect estimation performance of using covariate adjustment and IPTW with logistic-based propensity score in linear regression model (they were using the dataset as a cross-sectional dataset). Their methods were similar to our cases in terms of comparing covariate adjustment with logistic-based PS weighting, although we were using matching and overlap weights as PS weighting methods and Cox PH model for treatment effect estimation. Our findings were consistent with their conclusion that using both covariate adjustment and PS weighting might help with reducing bias. Lee (2010), along with Setoguchi (2008), suggested that using machine learning algorithms instead of logistic regression models for PS estimation can largely reduce bias across sample sizes, true model scenarios, and PS application methods. Freedman (2008) claimed that PS weighting would more often bring in bias than help reduce the bias when compared with model adjusting. But our study did not find substantial differences in bias reduction between SL-based and logistic-based PS in general. We also find that PS weighting could be beneficial for bias reduction for smaller sample sizes. However, a major difference to note is that the simulated data in our study was in time-to-event structure while the previous studies were conducted on simulated cross-sectional data. Lee (2010) and Pirracchio (2014) also used a number of machine learning algorithms for PS estimation, either separately or through SL, while we only used two that were mentioned as most strong candidate for this purpose. The covariate setting in the previous studies based on data with linear and binary outcomes were also different from our study. There were three types of covariates in their generated data: exposure predictors that were covariates only associated with generated exposure, outcome predictors, and confounders that were associated with both

exposure and outcome. Our study used the same covariate model when generating treatment and outcome.

4.2 **Limitations**

Due to computational feasibility, we did not test on simulation setting with additional sample sizes. Thus, the conclusions might not apply to situations that were not represented by our simulated data. Some specific simulation setting, such as lower outcome prevalence and the presence of covariates with relationships with outcome but not treatment, would require investigation. Also, the association between the covariates and the treatment was moderate to small in size as the coefficients from the fitted models were relatively small. Stronger association could contribute to more distinct patterns in the results.

The Super Learner method relies on cross-validation which may become infeasible at the larger sample sizes in some EHR studies. Our simulation study used the default settings for parameters in the machine learning algorithms. This mode of parameter settings might not be the optimal setting to utilize the ML algorithms of interest, despite that we chose the algorithms that were claimed to have better performance in propensity score estimation (Lee et al. 2010). The versions of the machine learning algorithms we applied to the data were those available in the Super Learner algorithm library. The performance of machine learning methods might differ across algorithm packages and lead to different observations in results.

The use of machine learning methods to estimate propensity scores is of interest because the true relationship between treatment allocation and observed covariates is generally unknown. In our study, machine learning methods do not seem to have better performance than the logistic method in terms of bias reduction but might be helpful for variance reduction. Adjusting treatment effect models with baseline covariates is more beneficial in a larger sample but less beneficial in a smaller sample in terms of bias reduction for bigger sample sizes than having the models weighted with PS. The overlap weight method could yield slightly better covariate balance if matching weight method could not provide satisfying ASAM values. Future work can be done in the outcome setting of rare outcomes, competing risks, or additional PS utilization methods such as adjusting for the propensity scores in the outcome model.

References

Austin, P. C. (2009). Balance diagnostics for comparing the distribution of baseline covariates between treatment groups in propensity score matched samples. *Stat Med*, *28*(25), 3083–3197. *https://doi.org/10.1002/sim.3697*

Austin, P. C. (2011). An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate Behav Res*, *46*(3), 399–424. *https://doi.org/10.1080/00273171.2011.568786*

Austin, P. C., & Stuart, E. A. (2015). Moving towards best practice when using inverse probability of treatment weighting (IPTW) using the propensity score to estimate causal treatment effects in observational studies. *Stat Med*, *34*(28), 3661–3679.

Austin, P., & Schuster, T. (2016). The performance of different propensity score methods for estimating absolute effects of treatments on survival outcomes: A simulation study. *Statistical Methods in Medical Research*, *25*(5), 2214–2237. *https://doi.org/10.1177/0962280213519716*

Badillo, Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L., Kam-Thong, T., Siebourg-Polster, J., Steiert, B., & Zhang, J. D. (2020). An Introduction to Machine Learning. *Clinical Pharmacology and Therapeutics*, *107*(4), 871–885. https://doi.org/10.1002/cpt.1796

Bender, R., Augustin, T., & Blettner, M. (2005). Generating survival times to simulate Cox proportional hazards models. *Stat Med*, *24*(11), 1713–1723. *https://doi.org/10.1002/sim.2059*

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32. *https://doi.org/10.1023/A:1010933404324*

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and Regression Trees* (1st ed.).

Brookhart, M. A., Schneeweiss, S., Rothman, K., Glynn, R. J., Avorn, J., & Stürmer, T. (2006). Variable selection for propensity score models. *American Journal of Epidemiology*, *163*(12), 1149–1156. *https://doi.org/10.1093/aje/kwj149*

Cham, H., Hurley, L., & Teng, Y. (2016, May). Optimizing random forests propensity scores [Conference presentation]. 2016 Modern Modeling Methods Conference, Storrs, CT, United States. https://modeling.uconn.edu/wp-content/uploads/sites/1188/2016/05/Optimizing-Random-Forests-propensity-score.pdf

Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Cole, S. R., & Hernan, M. A. (2008). Constructing inverse probability weights for marginal structural models. *American Journal of Epidemiology*, *168*(6), 656–664. *https://doi.org/10.1093/aje/kwn164*

Connors, A. F., Speroff, T., Dawson, N. V., Thomas, C., & Harrell, F. E. (1996). The effectiveness of right heart catheterization in the initial care of critically ill patients. SUPPORT Investigators. *Journal of the American Statistical Association*, *276*(11), 889–897. *https://doi.org/10.1001/jama.276.11.889*

D'Agostino, R. B. (2007). Propensity scores in cardiovascular research. *Circulation*, *115*(17), 2340–2343. *https://doi.org/10.1161/CIRCULATIONAHA.105.594952*

Deb, Austin, P. C., Tu, J. V., Ko, D. T., Mazer, C. D., Kiss, A., & Fremes, S. E. (2016). A Review of Propensity-Score Methods and Their Use in Cardiovascular Research. *Canadian Journal of Cardiology*, *32*(2), 259–265. https://doi.org/10.1016/j.cjca.2015.05.015Dudoit, S., & van der Laan, M. (2005). Asymptotics of cross-validated risk estimation in estimator selection and performance assessment. *Stat Methodol*, *2*(2), 131–154. *https://doi.org/10.1016/j.stamet.2005.02.003*

Elith, J., Leathwick, J. R., & Hastie, T. (2008). A working guide to boosted regression trees. *J Anim Ecol*, *77*(4), 802–813. *https://doi.org/10.1111/j.1365-2656.2008.01390.x*

Ferri-Garcia, R., & del Mar Rueda, M. (2020). Propensity score adjustment using machine learning classification algorithms to control selection bias in online surveys. *PLoS ONE*, *15*(4), e0231500. *https://doi.org/10.1371/journal.pone.0231500*

Franklin, J. M., Eddings, W., Austin, P. C., Stuart, E. A., & Schneeweiss, S. (2017). Comparing the performance of propensity score methods in healthcare database studies with rare outcomes. *Statist Med*, *36*(12), 1946–1963. *https://doi.org/10.1002/sim.7250*

Freedman, D. A., & Berk, R. A. (2008). Weighting regressions by propensity scores. *Evaluation Review*, *32*(4), 392–409. *https://doi.org/10.1177/0193841X08317586*

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232. *https://doi.org/10.1214/aos/1013203451*

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive Logistic Regression: A Statistical View of Boosting. *The Annals of Statistics*, *28*(2), 337–407. *https://doi.org/10.1214/aos/1016218222*

Griffin, B. A., McCaffery, D. F., Almirall, D., Burgette, L. F., & Setodji, C. M. (2017). Chasing balance and other recommendations for improving nonparametric propensity score models. *Journal of Causal Inference*, *5*(2), 169–188. *https://doi.org/10.1515/jci-2015-0026*

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning* (2nd ed.). Springer.

Hirano, K., & Imbens, G. (2001). Estimation of causal effects using propensity score weighting: An application to data on right heart catheterization. *Health Services and Outcome Research Methodology*, *2*, 259–278. *https://doi.org/10.1023/A:1020371312283*

Joffe, & Rosenbaum, P. R. (1999). Invited Commentary: Propensity Scores. *American Journal of Epidemiology*, *150*(4), 327–333. *https://doi.org/10.1093/oxfordjournals.aje.a010011*

Ju, C., Combs, M., Lendle, S. D., Frankin, J. M., Wyss, R., Schneeweiss, S., & van der Laan, M. J. (2019). Propensity score prediction for electronic healthcare databases using Super Learner and High-dimensional Propensity Score Methods. *J Appl Stat*, *46*(12), 2216–2236. *https://doi.org/10.1080/02664763.2019.1582614*

Kang, J. D. Y., & Schafer, J. L. (2007). Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data. *Statistical Science*, *22*(4), 523–539. *https://doi.org/10.1214/07-STS227*

Lee, B. K., Lessler, J., & Stuart, E. A. (2010). Improving propensity score weighting using machine learning. *Stat Med*, *29*(3), 337–346. *https://doi.org/10.1002/sim.3782*

Li, F., Morgan, K. L., & Zaslavsky, A. M. (2018). Balancing covariates via propensity score weighting. *Journal of the American Statistical Association*, *113*(521), 390–400. *https://doi.org/10.1080/01621459.2016.1260466*

Li, F., Thomas, L. E., & Li, F. (2019). Addressing extreme propensity scores via the overlap weights. *American Journal of Epidemiology*, *188*(1), 250–257. *https://doi.org/10.1093/aje/kwy201*

Li, L., & Greene, T. (2013). A weighting analogue to pair matching in propensity score analysis. *The International Journal of Biostatistics*, *9*(2), 215–234. *https://doi.org/10.1515/ijb-2012-0030.*

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News 2*(3), 18-22.

McCaffrey, D. F., Ridgeway, G., & Morral, A. R. (2004). Propensity score estimation with boosted regression for evaluating causal effects in observational studies. *Psychological methods, 9*(4), 403–425. https://doi.org/10.1037/1082-989X.9.4.403

Morina, D., & Navarro, A. (2014). The R package survsim for the simulation of simple and complex survival data. *Journal of Statistical Software*, *59*(2), 1–20. *https://doi.org/10.18637/jss.v059.i02*

Pirracchio, R., Petersen, M. L., & van der Laan, M. (2014). Improving propensity score estimators' robustness to model misspecification using Super Learner. *American Journal of Epidemiology*, *181*(2), 108–119. *https://doi.org/10.1093/aje/kwu253*

Polsky, D., Eremina, D., Hess, G., Hill, J., Hulnick, S., Roumm, A., Whyte, J. L., & Kallich, J. (2009). The importance of clinical variables in comparative analyses using propensity-score matching: The case of ESA costs for the treatment of chemotherapy-induced anaemia. *Pharmacoeconomics*, *27*(9), 755–765.

Robins, J. M., Hernan, M. A., & Brumback, B. (2000). Marginal structural models and causal inference in Epidemiology. *Epidemiology*, *11*(5), 550–560.

Rosenbaum, P. R., & Rubin, D. B. (1983). The central role of the propensity score in observational stdies for causal effects. *Biometrika*, *70*(1), 41–55.

Setoguchi, S., Schneeweiss, S., Brookhart, M. A., Glynn, R. J., & Cook, E. F. (2008). Evaluating uses of data mining techniques in propensity score estimation: A simulation study. *Pharmacoepidemiol Drug Saf.*, *17*(6), 546–555. *https://doi.org/10.1002/pds.1555*

Shadish, W. R., Clark, M. H., & Steiner, P. M. (2008). Can nonrandomized experiments yield accurate answers? A randomized experiment comparing random and nonrandom assignments. *Journal of the American Statistical Association*, *103*(484), 1334–1344.

Sinisi, S. E., Polley, E. C., Petersen, M. L., Rhee, S. Y., & van der Laan, M. J. (2007). Super learning: An application to the prediction of HIV-1 drug resistance. *Stat Appl Genet Mol Biol*. *https://doi.org/10.2202/1544-6115.1240*

Stuart, E. A. (2010). Matching methods for causal inference: A review and a look forward. *Statistical Science*, *25*(1), 1–21. *https://doi.org/10.1214/09-STS313*

Thomas, L. E., Li, F., & Pencina, M. J. (2020). Overlap weighting: A propensity score method that mimics attributes of a randomized clinical trial. *Journal of the American Statistical Association*, *323*(23), 2417–2418. *https://doi.org/10.1001/jama.2020.7819*

van der Laan, M., Dudoit, S., & van der Vaart, A. W. (2006). The cross-validated adaptive epsilon-net estimator. *Stat Dec.*, *24*(3), 373–395.

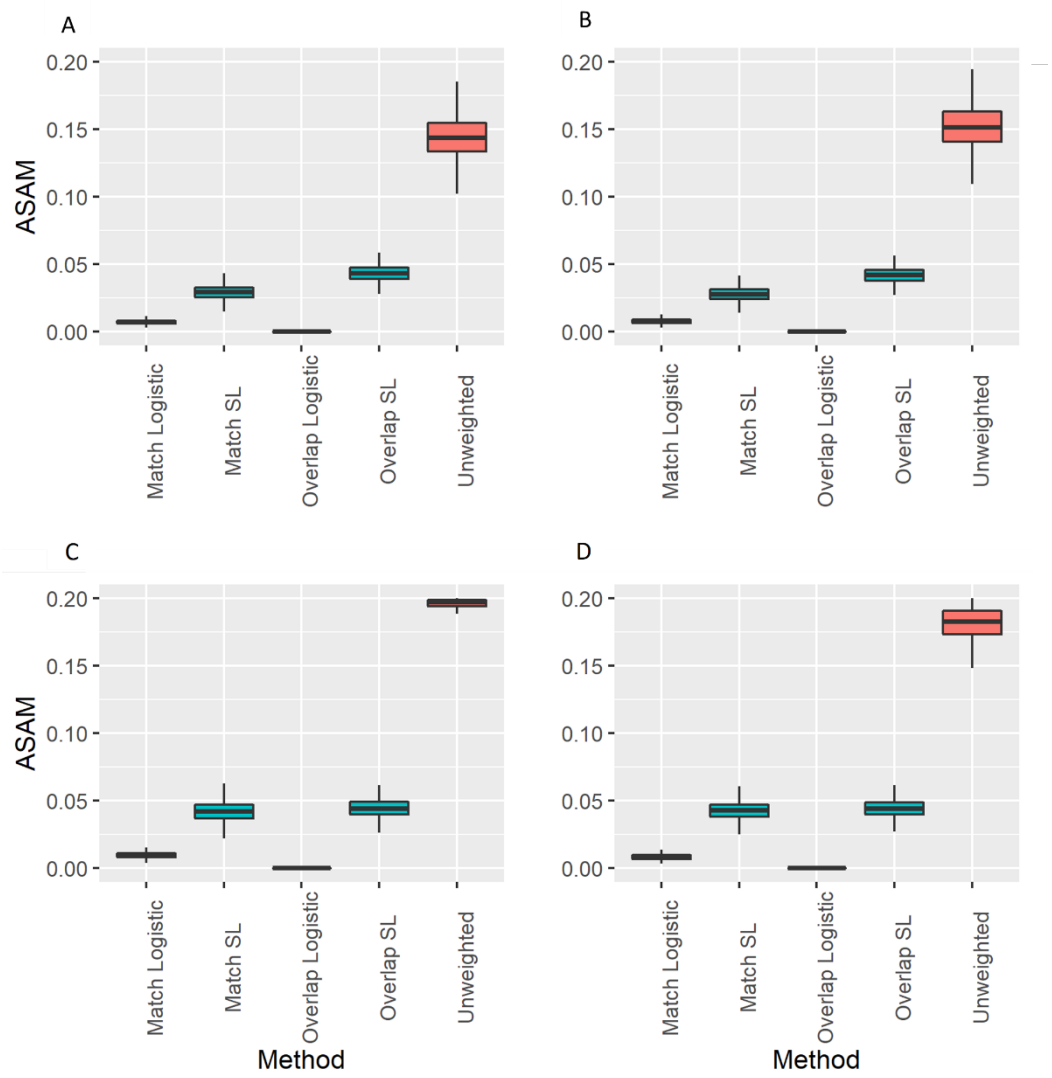
van der Laan, M., Polley, E. C., & Hubbard, A. E. (2007). Super Learner. *Stat Appl Genet Mol Biol*.

Yoshida, K., Hernandez-Diaz, S., Solomon, D. H., Jackson, J. W., Gagne, J. J., Glynn, R. J., & Franklin, J. M. (2017). Matching weights to simultaneously compare three treatment groups: Comparison to three-way matching. *Epidemiology*, *28*(3), 387–395. *https://doi.org/10.1097/EDE.0000000000000627.*

Appendices

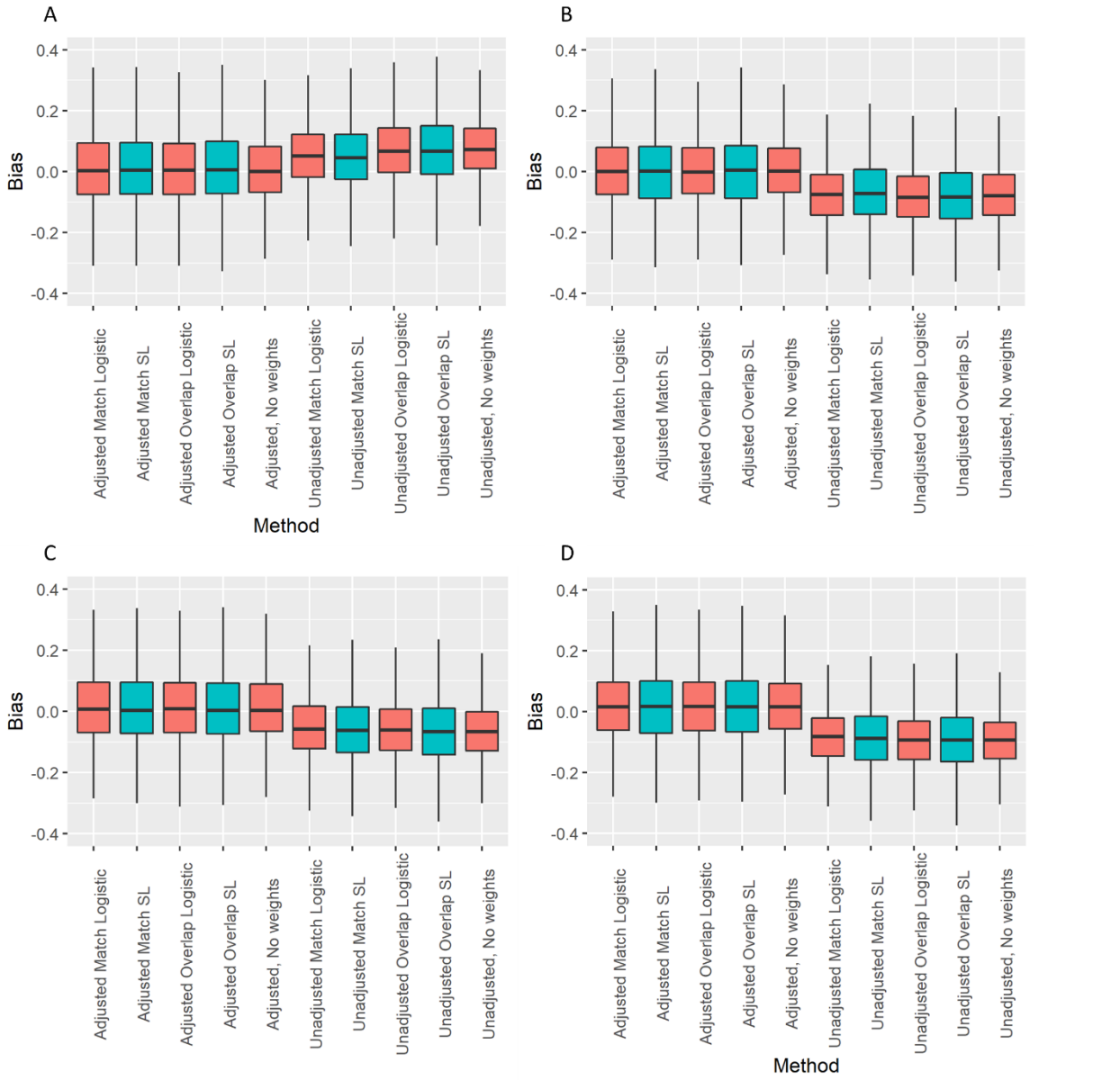
6.1 Appendix A. Tables and Figures

Figure 2. Distribution of ASAM for Each Scenario with Logistic or SL Method at n=1000



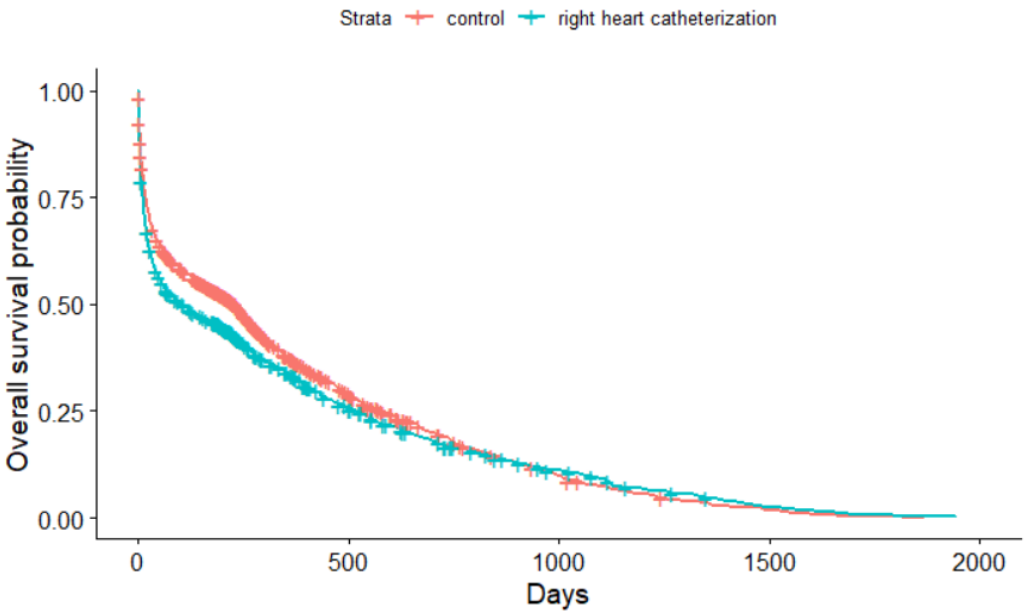
A) Scenario A (i.e., additivity and linearity); B) scenario B (i.e., nonlinearity C) scenario C (i.e., nonadditivity); D) scenario D (i.e., nonadditivity and nonlinearity). The midline represents the mean value, and the vertical lines show the 2.5% and 97.5% quantiles.

Figure 3. Bias for Different Method Combinations between Logistic/SL and Matching/Overlap Weight at n=1000



A) Scenario A (i.e., additivity and linearity); B) scenario B (i.e., nonlinearity C) scenario C (i.e., nonadditivity); D) scenario D (i.e., nonadditivity and nonlinearity). The midline represents the mean value, and the vertical lines show the 2.5% and 97.5% quantiles.

Figure 4. Kaplan-Meier Plot for Right Heart Catheterization Data



	Days	# of risk	# of event
Treatment	90	1108	68
Control	90	2077	120

Table 3. Simulation Results Obtained under 4 Scenarios with Different Propensity Score Estimation and Application Methods (n=500)

True model ("Scenarios")	Outcome Model	PS Estimation Model	PS Weighting method	Bias	MSE	RMSE	% of CI coverage	CI width	ASAM
A. Additivity and linearity (Main effects only)	Adjusted	Unweighted	/	0.0092	0.0248	0.1575	94.5	0.601	0.155
		Logit	Matching	0.0117	0.0279	0.1671	94.0	0.633	0.010
		SL	Matching	0.0150	0.0357	0.1891	94.0	0.701	0.047
		Logit	Overlapping	0.0111	0.0282	0.1678	94.1	0.633	0.000
		SL	Overlapping	0.0172	0.0359	0.1896	94.2	0.704	0.063
	Unadjusted	Unweighted	/	-0.0771	0.0246	0.1568	89.8	0.529	0.155
		Logit	Matching	-0.0686	0.0258	0.1607	90.9	0.563	0.010
		SL	Matching	-0.0629	0.0311	0.1765	92.5	0.632	0.047
		Logit	Overlapping	-0.0780	0.0274	0.1655	89.8	0.560	0.000
		SL	Overlapping	-0.0726	0.0325	0.1803	91.7	0.629	0.063
B. Nonlinearity (Main effects + quadratic terms)	Adjusted	Unweighted	/	0.0114	0.0248	0.1575	94.9	0.611	0.160
		Logit	Matching	0.0191	0.0309	0.1759	94.0	0.667	0.011
		SL	Matching	0.0227	0.0373	0.1930	93.9	0.723	0.045
		Logit	Overlapping	0.0187	0.0308	0.1756	94.0	0.666	0.000
		SL	Overlapping	0.0193	0.0371	0.1926	93.7	0.724	0.062
	Unadjusted	Unweighted	/	-0.0625	0.0226	0.1504	93.4	0.541	0.160
		Logit	Matching	-0.0462	0.0260	0.1613	94.3	0.598	0.011
		SL	Matching	-0.0512	0.0306	0.1749	92.6	0.650	0.045
		Logit	Overlapping	-0.0553	0.0263	0.1621	92.9	0.593	0.000
		SL	Overlapping	-0.0538	0.0312	0.1766	92.5	0.652	0.062
C. Non-additivity (Main effects + interaction terms)	Adjusted	Unweighted	/	0.0180	0.0253	0.1591	95.8	0.637	0.240
		Logit	Matching	0.0196	0.0327	0.1807	93.7	0.694	0.014
		SL	Matching	0.0234	0.0352	0.1875	95.1	0.731	0.052
		Logit	Overlapping	0.0199	0.0329	0.1813	94.6	0.700	0.000
		SL	Overlapping	0.0264	0.0358	0.1892	95.4	0.738	0.078
	Unadjusted	Unweighted	/	0.0826	0.0283	0.1681	91.8	0.571	0.240
		Logit	Matching	0.0578	0.0287	0.1694	94.5	0.632	0.014
		SL	Matching	0.0636	0.0333	0.1824	94.5	0.682	0.052
		Logit	Overlapping	0.0734	0.0334	0.1828	93.5	0.646	0.000
		SL	Overlapping	0.0812	0.0383	0.1957	93.6	0.698	0.078
D. Non-additivity and nonlinearity (Main effects+ quadratic term + interaction term)	Adjusted	Unweighted	/	0.0314	0.0280	0.1674	94.1	0.633	0.200
		Logit	Matching	0.0365	0.0344	0.1856	93.6	0.692	0.012
		SL	Matching	0.0409	0.0390	0.1975	93.0	0.729	0.048
		Logit	Overlapping	0.0356	0.0338	0.1837	93.6	0.694	0.000
		SL	Overlapping	0.0398	0.0389	0.1973	93.3	0.732	0.066
	Unadjusted	Unweighted	/	-0.0917	0.0252	0.1589	86.7	0.491	0.200
		Logit	Matching	-0.0757	0.0258	0.1605	90.9	0.556	0.012
		SL	Matching	-0.0793	0.0307	0.1751	91.1	0.605	0.048
		Logit	Overlapping	-0.0852	0.0282	0.1678	89.6	0.556	0.000
		SL	Overlapping	-0.0832	0.0321	0.1792	90.9	0.608	0.066

Table 4. Method Application on the RHC Data

True model	Outcome Model	PS Estimation Model	PS Weighting method	Estimated HR of Treatment	CI width	ASAM
RHC data (True Model Unknown)	Adjusted	Unweighted	/	1.257	0.177	0.158
		Logit	Matching	1.255	0.186	0.004
		SL	Matching	1.196	0.189	0.044
		Logit	Overlapping	1.266	0.187	0.000
		SL	Overlapping	1.205	0.185	0.025
	Unadjusted	Unweighted	/	1.164	0.153	0.158
		Logit	Matching	1.209	0.171	0.004
		SL	Matching	1.160	0.182	0.044
		Logit	Overlapping	1.213	0.170	0.000
		SL	Overlapping	1.158	0.171	0.025

6.2 Appendix B. Formulas and Coefficients of the Treatment Models and the Outcome Models Fitted on the RHC Dataset for Data Generation and Corresponding True Hazard Ratio of Treatment for Each Scenario in Simulations

Scenario A

Treatment model (logistic):

trt ~ -0.187 - 0.004 pafi1 – 0.004 age + 0.006 hrt1 – 0.018 resp1 + 0.042 bili1 – 0.670 dnr1 + 0.787 card + 0.581 private + 0.364 medicare + 0.522 priNcare + 0.457 other

Outcome model (Cox PH model):

Surv (survtime, death) ~ 0.229 trt + 0.001 pafi1 + 0.01 age
+ 0.002 hrt1 - 0.001 resp1 + 0.039 bili1 + 0.842 dnr1 - 0.151 card + 0.081 private + 0.015 medicare + 0.084 priNcare + 0.009 other

True hazard ratio of treatment: 1.257

Scenario B

Treatment model (logistic):

trt ~ -1.449 - 0.004 pafi1 – 0.054 age + 0.001 hrt1 – 0.021 resp1 + 0.042 bili1 – 0.617 dnr1 + 0.778 card + 0.554 private + 0.439 medicare + 0.569 priNcare + 0.486 other + 0.01 I(age^2) + 0.001 I(hrt1^2) + 0.001 I(resp1^2)

Outcome model (Cox PH model):

Surv (survtime, death) ~ 0.219 trt + 0.001 pafi1 + 0.03 age
- 0.006 hrt1 - 0.015 resp1 + 0.041 bili1 + 0.856 dnr1 - 0.162 card + 0.071 private + 0.031 medicare + 0.081 priNcare + 0.011 other + 0.01 I(age^2) + 0.001 I(hrt1^2) + 0.001 I(resp1^2)

True hazard ratio of treatment: 1.245

Scenario C

Treatment model (logistic):

trt ~ -1.048 - 0.005 pafi1 + 0.01 age + 0.007 hrt1 – 0.015 resp1 + 0.053 bili1 - 0.185 dnr1 + 2.269 card + 0.527 private + 1.578 medicare + 0.898 priNcare + 0.965 other - 0.006 age*dnr1 + 0.1 age*card + 0.001 age*private – 0.019 age*medicare -0.007 age*priNcare - 0.01 age*other + 0.1 pafi1*card + 0.1 hrt1*card + 0.1 card*resp1 + 0.1 card*bili1

Outcome model (Cox PH model):

Surv (survtime, death) ~ 0.215 trt + 0.001 pafi1 + 0.015 age
+ 0.002 hrt1 + 0.001 resp1 + 0.038 bili1 + 2.38 dnr1 + 0.125 card + 0.024 private + 0.09 medicare + 0.622 priNcare – 0.109 other - 0.021 age*dnr1 + 0.1 age*card + 0.001 age*private – 0.001 age*medicare -0.008 age*priNcare + 0.002 age*other + 0.1 pafi1*card + 0.1 hrt1*card + 0.1 card*resp1 + 0.1 card*bili1

True hazard ratio of treatment: 1.24

Scenario D

Treatment model (logistic):

trt ~ -1.572 - 0.005 pafi1 + 0.045 age + 0.003 hrt1 – 0.017 resp1 + 0.053 bili1 – 0.374 dnr1 + 2.032 card + 0.405 private + 0.868 medicare + 0.093 priNcare + 0.667 other + 0.01 I(age^2) + 0.001 I(hrt1^2) + 0.001 I(resp1^2) – 0.003 age*dnr1 + 0.1 age*card + 0.003 age*private – 0.006 age*medicare + 0.007 age*priNcare – 0.003 age*other + 0.1 pafi1*card + 0.1 hrt1*card + 0.1 card*resp1 + 0.1 card*bili1

Outcome model (Cox PH model):

Surv (survtime, death) ~ 0.215 trt + 0.001 pafi1 + 0.017 age – 0.007 hrt1 – 0.014 resp1 + 0.04 bili1 + 2.328 dnr1 - 0.081 card – 0.003 private + 0.082 medicare + 0.624 priNcare – 0.161 other + 0.01 I(age^2) + 0.001 I(hrt1^2) + 0.001 I(resp1^2) – 0.021 age*dnr1 + 0.1 age*card + 0.002 age*private + 0.001 age*medicare – 0.008 age*priNcare + 0.003 age*other + 0.1 pafi1*card + 0.1 hrt1*card + 0.1 card*resp1 + 0.1 card*bili1

True hazard ratio of treatment: 1.24

6.3 Appendix C. R Codes for Simulation

```
knitr::opts_chunk$set(echo = TRUE)

## ======================
## packages used
## ======================
library(survival) ## coxph
library(survey)  ## svydesign and svycoxph
library(simsurv) ## simsurv

#packages for SuperLearner function
library(SuperLearner)
library(caret)
library(glmnet)
library(randomForest)
library(RhpcBLASctl)
library(xgboost)

#package for ASAM
library(PSweight)
start.time = Sys.time()
start.time

## ======================
## simulation parameters
## ======================

seed=1026
set.seed(seed)
n= 100 ## size of data set
nsim = 100## number of simulated data sets

lambda_out = .01
censortime = 90
## ======================
## Create object to save results
## ======================
p_trt = rep(NA,nsim)
event_rate = rep(NA,nsim)
times_c<-vector(mode="list", length=nsim)
times_uc<-vector(mode="list", length=nsim)
status_list<-vector(mode="list", length=nsim)

##simulated datasets
dat_cox<-vector(mode="list", length=nsim)
#load("~/R/summer2020/simulation script 3/sim1D_n1000_datasets.RData") #used when running on saved data

##### for adjusted cox ph model results
## ignoring weights
HR_est = rep(NA,nsim)
```

```
#HR_mse = rep(NA,nsim)
HR_ci = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci) = c("lower","upper")
HR_coverage_ind = rep(NA,nsim)
ci_width_uw=rep(NA,nsim) ##width of CI

## incorporating weights: matching weights
HR_est_wt = rep(NA,nsim)
HR_ci_wt = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt) = c("lower","upper")
HR_coverage_ind_wt = rep(NA,nsim)
ci_width=rep(NA,nsim) #width of CI
#CIF_est_wt = matrix(0,nrow = nsim,ncol = n)
#CIF_mse_wt = matrix(0,nrow = nsim,ncol = n)
#CIV_coverage_ind_wt = matrix(0,nrow = nsim,ncol = n)

## incorporating weights: overlapping weights
HR_est_wt2 = rep(NA,nsim)
HR_ci_wt2 = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt2) = c("lower","upper")
HR_coverage_ind_wt2 = rep(NA,nsim)
ci_width2=rep(NA,nsim) #width of CI


##from superlearner: coefficients for each ML methods
model_coef=matrix(0, ncol=2, nrow=nsim)
colnames(model_coef)=c('RF', 'XGB')



##ASAM
asam= matrix(0,nrow = nsim,ncol = 3)
colnames(asam) = c("unweighted","matching", "overlap")


##### for unadjusted cox ph model results
## ignoring weights
HR_est_ua = rep(NA,nsim)
#HR_mse = rep(NA,nsim)
HR_ci_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_ua) = c("lower","upper")
HR_coverage_ind_ua = rep(NA,nsim)
ci_width_uw_ua=rep(NA,nsim) ##width of CI

## incorporating weights: matching weights
HR_est_wt_ua = rep(NA,nsim)
HR_ci_wt_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt_ua) = c("lower","upper")
HR_coverage_ind_wt_ua = rep(NA,nsim)
ci_width_ua=rep(NA,nsim) #width of CI


## incorporating weights: overlapping weights
```

```
HR_est_wt2_ua = rep(NA,nsim)
HR_ci_wt2_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt2_ua) = c("lower","upper")
HR_coverage_ind_wt2_ua = rep(NA,nsim)
ci_width2_ua=rep(NA,nsim) #width of CI

##from superlearner
model_coef_ua=matrix(0, ncol=2, nrow=nsim)
colnames(model_coef_ua)=c('RF', 'XGB')

##ASAM
asam_ua= matrix(0,nrow = nsim,ncol = 3)
colnames(asam_ua) = c("unweighted","matching", "overlap")



## =====================
## set up: generate X values
## =====================
rhc.dat = read.csv("~/R/summer2020/rhc.csv", stringsAsFactors=T)
dnr1_num<-654
card_num<-1931
private<-1698
medicare<-1458
priNcare<-1236
other<-1343

simulate_cov<-function(n){
  #continuous: Pafi1, age, hrt1, resp1, bili1
  pafi1<-rnorm(n)
  age<-rnorm(n)
  hrt1<-rnorm(n)
  resp1<-rnorm(n)
  bili1<-rnorm(n)

  #binary: dnr1, card
  dnr1<-rbinom(n, 1, dnr1_num/nrow(rhc.dat))
  card<-rbinom(n, 1, card_num/nrow(rhc.dat))
  #categorical:  nineclas
  ninclas.private<-rbinom(n,1, private/nrow(rhc.dat))
  ninclas.medicare<-rbinom(n,1, medicare/nrow(rhc.dat))
  ninclas.priNcare<-rbinom(n,1, priNcare/nrow(rhc.dat))
  ninclas.other<-rbinom(n,1, other/nrow(rhc.dat))

  #simulated data frame
  sim.dat<-cbind(pafi1, age, hrt1, resp1, bili1, dnr1, card,
                 ninclas.private, ninclas.medicare, ninclas.priNcare, ninclas.other)
  colnames(sim.dat)<-c("pafi1", "age", "hrt1", "resp1", "bili1", "dnr1", "card",
                       "private", "medicare", "priNcare", "other")

  sim.dat
}
```

```
## ======================
###set up: generate treatment values
## ======================

vars_selected<-c("pafi1", "age", "hrt1", "resp1", "bili1", "dnr1", "card", "ninsclas")
vars_converted<-c("pafi1", "age", "hrt1", "resp1", "bili1", "dnr1", "card",
                  "private", "medicare", "priNcare", "other")

ps.formula<- trt~pafi1+age+hrt1+resp1+bili1+dnr1+card+private+medicare+priNcare+other

rhc.X<-cbind(rhc.dat[,vars_selected])

##convert factor variables in rhc.cov into indicator variables
rhc.conv<- model.matrix(~., data= rhc.X)[,-1]
rhc.conv<-cbind(rhc.conv, rhc.dat$swang1) #add trt var into the db
rhc.conv<-data.frame(rhc.conv)
rhc.conv[,14]<-rhc.conv[,11]
rhc.conv[,15]<-rhc.conv[,8]
rhc.conv[,16]<-rhc.conv[,12]
rhc.conv[,17]<-rhc.conv[,9]+rhc.conv[,10]
rhc.conv<-cbind(rhc.conv[,1:7],rhc.conv[,13:17])
colnames(rhc.conv)[6:12]<-c("dnr1", "card", "trt", "private", "medicare", "priNcare", "other")
rhc.conv$trt[which(rhc.conv$trt==1)]<-0
rhc.conv$trt[which(rhc.conv$trt==2)]<-1
#head(rhc.conv)


#true mode: main effects only (Scenario A)
#trt_formula<-as.formula(paste("trt ~", paste(vars_converted, collapse="+")))
#trt_mod<-glm(trt_formula, data=rhc.conv, family=binomial())

#true model with 3 quadratic terms (Scenario B)
#trt_formula<-as.formula(paste("trt ~", paste(vars_converted, collapse="+"),
                    #"+ I(age^2) + I(hrt1^2) +I(resp1^2)"))
#trt_mod<-glm(trt_formula, data=rhc.conv, family=binomial())


#true model with 10 interaction terms (Scenario C)
#trt_formula<-as.formula(paste("trt ~", paste(vars_converted, collapse="+"),
                    #"+ age*dnr1 + age*card + age*private + age*medicare + age*priNcare +
                    #age*other + card*pafi1 + card*hrt1 + card*resp1 + card*bili1"))
#trt_mod<-glm(trt_formula, data=rhc.conv, family=binomial())


#true model with interaction and quadratic terms (Scenario D)
trt_formula<-as.formula(paste("trt ~", paste(vars_converted, collapse="+"),
                    "+ age*dnr1 + age*card + age*private + age*medicare + age*priNcare +
                    age*other + card*pafi1 + card*hrt1 + card*resp1 + card*bili1 +
                    I(age^2) + I(hrt1^2) +I(resp1^2)"))
trt_mod<-glm(trt_formula, data=rhc.conv, family=binomial())

#betas for trt model
beta_trt_mod<-trt_mod$coeff
```

```
##adjust coefficients
  #all nonliner terms: >0.001 and setting the coefficient for pafi1 equal to 0.001
beta_trt_mod[abs(beta_trt_mod)<0.001]=0.001
  #age^2: at least 0.01
if(beta_trt_mod["I(age^2)"]<0.01){beta_trt_mod["I(age^2)"]=0.01}
  #interaction terms with the card variable: at least 0.1
indx=which(grepl(':card', names(beta_trt_mod)))
beta_trt_mod[indx][beta_trt_mod[indx]<0.1]=0.1

round(beta_trt_mod,3)

## =====================
## set up: generate event times using cox ph and exponential baseline
## =====================

##get survival time of RHC
survtime<-ifelse(is.na(rhc.dat$dthdte), rhc.dat$lstctdte-rhc.dat$sadmdte, rhc.dat$dthdte-rhc.dat$sadmdte)
rhc.conv$survtime<-survtime
death<-(rhc.dat$death=="Yes")^2
rhc.conv$death<-death

#surv_form<-as.formula(paste("Surv(survtime, death)~", "trt +", paste(vars_converted, collapse="+")) )
surv_form<-as.formula(paste("Surv(survtime, death)~trt+", paste(vars_converted, collapse="+"),
                            "+ age*dnr1 + age*card + age*private + age*medicare + age*priNcare +
                            age*other + card*pafi1 + card*hrt1 + card*resp1 + card*bili1 +
                            I(age^2) + I(hrt1^2) +I(resp1^2)") )
surv_mod<-coxph(surv_form, data=rhc.conv)
beta_out_mod<-surv_mod$coefficients

##adjust coefficients
  #all nonliner terms: >0.001 and setting the coefficient for pafi1 equal to 0.001
beta_out_mod[abs(beta_out_mod)<0.001]=0.001
  #age^2: at least 0.01
if(beta_out_mod["I(age^2)"]<0.01){beta_out_mod["I(age^2)"]=0.01}
  #interaction terms with the card variable: at least 0.1
indx=which(grepl(':card', names(beta_out_mod)))
beta_out_mod[indx][beta_out_mod[indx]<0.1]=0.1

round(beta_out_mod,3)


#get HR of treatment=hazard of exposed/ hazard of unexposed
out_trt<-beta_out_mod[1]
round(out_trt,3)
round(exp(out_trt),3)

for(sim_num in 1:nsim){


  ## +++++
  ## generate X values
```

```
## +++++
X<-simulate_cov(n)

 ## for main effects model(Scenario A)
 #X_int=X

 ## for with quadratic terms only (Scenario B)
 #X_int=cbind(X,(X["age"])^2, (X["hrt1"])^2, (X["resp1"])^2)

 ## for with interaction terms only (Scenario C)
 #X_int=cbind(X,(X["age"])*(X["dnr1"]), (X["age"])*(X["card"]), (X["age"])*(X["private"]),
          #(X["age"])*(X["medicare"]), (X["age"])*(X["priNcare"]), (X["age"])*(X["other"]),
          #(X["card"])*(X["pafi1"]), (X["card"])*(X["hrt1"]), (X["card"])*(X["resp1"]),
          #(X["card"])*(X["bili1"]))

## for with interaction and quadratic terms (Scenario D)
X_int=cbind(X, (X["age"])^2, (X["hrt1"])^2, (X["resp1"])^2,
         (X["age"])*(X["dnr1"]), (X["age"])*(X["card"]), (X["age"])*(X["private"]),
         (X["age"])*(X["medicare"]), (X["age"])*(X["priNcare"]), (X["age"])*(X["other"]),
         (X["card"])*(X["pafi1"]), (X["card"])*(X["hrt1"]), (X["card"])*(X["resp1"]),
         (X["card"])*(X["bili1"]))


## +++++
## generate trt variable
## +++++
XB_trt<-cbind(1, X_int)%*%beta_trt_mod
p_treat<-exp(XB_trt)/(1+exp(XB_trt))
trt<-rbinom(n, 1, p_treat)
p_trt[sim_num] = round(sum(trt)/n,4)

## +++++
## generate event times using cox ph and exponential baseline
## +++++

X_frame=data.frame(trt, X_int)
beta_vect = c(beta_out_mod)
names(beta_vect) = colnames(X_frame)
t_temp =simsurv(dist = c("exponential"), lambdas = lambda_out,x = X_frame, betas =beta_vect)

t = t_temp$eventtime
times_uc[[sim_num]]<-t

## +++++
## generate status variable
## +++++
time = pmin(t, censortime) #censored times
times_c[[sim_num]]<-time
status = 1-(t > censortime)^2
status_list[[sim_num]]<-status
event_rate[sim_num] = sum(status)/n

dat_cox[[sim_num]] = data.frame(X,trt,time,status) #for censored time
```

```
 #dat_cox[[sim_num]] = data.frame(X,trt,t,status) #for uncensored time
 colnames(dat_cox[[sim_num]]) = c(vars_converted, "trt", "time", "status")


 #print(paste("sim number: ", sim_num, sep = ""))
}

#view of simulated data
head(dat_cox[[1]])
#save data sets
#save(dat_cox, file = "~/R/summer2020/simulation script 3/sim1D_n1000_datasets.RData")
#load("~/R/summer2020/simulation script 3/sim1D_n1000_datasets.RData")

## #Compare simulated data with RHC data
## data_mt=as.data.frame(do.call(rbind, dat_cox))
## head(data_mt)
## dim(data_mt)
##   #survival time
## hist(survtime, xlim=c(0,censortime), ylim=c(0, 600), nclass= 1200, main="RHC survival time") #RHC
## hist(sample(data_mt$time, nrow(rhc.dat)), ylim=c(0, 400), nclass= 100, main="simulated survival time") #simulated
##   #event rates
## sum(death)/nrow(rhc.dat) #RHC
## sum(data_mt$status)/nrow(data_mt) #simulated data
##
##
##



for(sim_num in 1:nsim){



## ++++++++++++++++++++++
 ## Fit cox proportional hazard model ignoring weights


 surv_form<-as.formula(paste("Surv(time, status)~", "trt +", paste(vars_converted, collapse="+") ))
 cox_mod <- coxph(surv_form, data =  dat_cox[[sim_num]])
 cox_mod_sum = summary(cox_mod)
 cox_mod_ci = exp(confint(cox_mod))
 HR_est[sim_num] = cox_mod_sum$coefficients[1,2]
 HR_ci[sim_num,] = cox_mod_ci[1,]
 HR_coverage_ind[sim_num] = (exp(out_trt) >= HR_ci[sim_num,1] &
                             exp(out_trt) <= HR_ci[sim_num,2])^2
 ci_width_uw[sim_num]=HR_ci[sim_num,2]- HR_ci[sim_num,1]


#------------------------

 ## fit model with weights
 log_mod = glm(as.formula(paste("trt~", paste(vars_converted, collapse="+")) ),
```

```
        data = dat_cox[[sim_num]], family = "binomial")
logit_ps = predict(log_mod)
prob_ps =  exp(logit_ps)/(1+exp(logit_ps))
prob_ps[which(prob_ps==1)]=0.999


#matching weights and evaluation
w_ps = pmin(prob_ps,1-prob_ps)/(trt*prob_ps + (1-trt)*(1-prob_ps))
w_ps[which(w_ps==min(w_ps))]=0.0001 #for bug fixing
svyDes = svydesign(id=~0, weights=w_ps, data=dat_cox[[sim_num]])
cox_weighted_mod = svycoxph(surv_form, design = svyDes)
HR_est_wt[sim_num] = exp(cox_weighted_mod$coefficients[1])
HR_ci_wt[sim_num,] =  exp(confint(cox_weighted_mod))[1,]
HR_coverage_ind_wt[sim_num] = (exp(out_trt) >= HR_ci_wt[sim_num,1] &
                          exp(out_trt) <= HR_ci_wt[sim_num,2])^2
ci_width[sim_num]=HR_ci_wt[sim_num,2]- HR_ci_wt[sim_num,1]

#overlap weights and evaluation
overlap_w_ps= trt + (-1)^trt * prob_ps
overlap_w_ps[which(overlap_w_ps==min(overlap_w_ps))]=0.0001 #for bug fixing
svyDes2 = svydesign(id=~0, weights=overlap_w_ps, data=dat_cox[[sim_num]])
cox_weighted_mod2 = svycoxph(surv_form, design = svyDes2)
HR_est_wt2[sim_num] = exp(cox_weighted_mod2$coefficients[1])
HR_ci_wt2[sim_num,] =  exp(confint(cox_weighted_mod2))[1,]
HR_coverage_ind_wt2[sim_num] = (exp(out_trt) >= HR_ci_wt2[sim_num,1] &
                          exp(out_trt) <= HR_ci_wt2[sim_num,2])^2
ci_width2[sim_num]=HR_ci_wt2[sim_num,2]- HR_ci_wt2[sim_num,1]

#ASAM
# using SumStat to estimate propensity scores
  #use PSweight package to generate PS
  #msstat1 <- SumStat(ps.formula, trtgrp="1", data=dat_cox[[sim_num]], weight="matching")
  #msstat2 <- SumStat(ps.formula, trtgrp="1", data=dat_cox[[sim_num]], weight="overlap")
msstat1 <- SumStat(zname="trt", xname=vars_converted, data=dat_cox[[sim_num]],
                   ps.estimate=prob_ps,
                   trtgrp="1",  weight="matching")
msstat2 <- SumStat(zname="trt", xname=vars_converted, data=dat_cox[[sim_num]],
                   ps.estimate=prob_ps,
                   trtgrp="1",  weight="overlap")

#ASAM for unweighted dataset
asam[sim_num,1]= mean(summary(msstat1)$unweighted[,"SMD"])
#ASAM for matching weight dataset
asam[sim_num,2]= mean(summary(msstat1)$matching[,"SMD"])
#ASAM for overlap weight dataset
asam[sim_num,3]= mean(summary(msstat2)$overlap[,"SMD"])


## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
####----------fit model with unadjusted cox ph mode #####
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
## Fit cox proportional hazard model ignoring weights

surv_form_ua<-as.formula(paste("Surv(time, status)~trt" ))
cox_mod <- coxph(surv_form_ua, data =  dat_cox[[sim_num]])
cox_mod_sum = summary(cox_mod)
cox_mod_ci = exp(confint(cox_mod))
HR_est_ua[sim_num] = cox_mod_sum$coefficients[1,2]
HR_ci_ua[sim_num,] = cox_mod_ci[1,]
HR_coverage_ind_ua[sim_num] = (exp(out_trt) >= HR_ci_ua[sim_num,1] &
                        exp(out_trt) <= HR_ci_ua[sim_num,2])^2
ci_width_uw_ua[sim_num]=HR_ci_ua[sim_num,2]- HR_ci_ua[sim_num,1]



## fit model with weights
#matching weights and evaluation

svyDes = svydesign(id=~0, weights=w_ps, data=dat_cox[[sim_num]])
cox_weighted_mod_ua = svycoxph(surv_form_ua, design = svyDes)
HR_est_wt_ua[sim_num] = exp(cox_weighted_mod_ua$coefficients[1])
HR_ci_wt_ua[sim_num,] =  exp(confint(cox_weighted_mod_ua))[1,]
HR_coverage_ind_wt_ua[sim_num] = (exp(out_trt) >= HR_ci_wt_ua[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt_ua[sim_num,2])^2
ci_width_ua[sim_num]=HR_ci_wt_ua[sim_num,2]- HR_ci_wt_ua[sim_num,1]

#overlap weights and evaluation
svyDes2 = svydesign(id=~0, weights=overlap_w_ps, data=dat_cox[[sim_num]])
cox_weighted_mod2_ua = svycoxph(surv_form_ua, design = svyDes2)
HR_est_wt2_ua[sim_num] = exp(cox_weighted_mod2_ua$coefficients[1])
HR_ci_wt2_ua[sim_num,] =  exp(confint(cox_weighted_mod2_ua))[1,]
HR_coverage_ind_wt2_ua[sim_num] = (exp(out_trt) >= HR_ci_wt2_ua[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt2_ua[sim_num,2])^2
ci_width2_ua[sim_num]=HR_ci_wt2_ua[sim_num,2]- HR_ci_wt2_ua[sim_num,1]

#ASAM
# using SumStat to estimate propensity scores
  #use PSweight package to generate PS
  #msstat1 <- SumStat(ps.formula, trtgrp="1", data=dat_cox[[sim_num]], weight="matching")
  #msstat2 <- SumStat(ps.formula, trtgrp="1", data=dat_cox[[sim_num]], weight="overlap")
msstat1_ua <- SumStat(zname="trt", xname=vars_converted, data=dat_cox[[sim_num]],
              ps.estimate=prob_ps,
              trtgrp="1",  weight="matching")
msstat2_ua<- SumStat(zname="trt", xname=vars_converted, data=dat_cox[[sim_num]],
              ps.estimate=prob_ps,
              trtgrp="1",  weight="overlap")

#ASAM for unweighted dataset
asam_ua[sim_num,1]= mean(summary(msstat1_ua)$unweighted[,"SMD"])
#ASAM for matching weight dataset
asam_ua[sim_num,2]= mean(summary(msstat1_ua)$matching[,"SMD"])
#ASAM for overlap weight dataset
asam_ua[sim_num,3]= mean(summary(msstat2_ua)$overlap[,"SMD"])
```

```
}


## ======================
## Summarize Simulation Results
## ======================
results_frame = data.frame(matrix(0,nrow = 6,ncol = 6))
colnames(results_frame) = c("Bias", "MSE","RMSE","% CI coverage", "CI width","ASAM")
rownames(results_frame) = c("no weights","matching weights", "overlap weights",
                            "no weights (unadj)","matching weights (unadj)", "overlap weights (unadj)")
results_frame[1,] = c(mean((HR_est -exp(out_trt))), #mean of bias
                      mean((HR_est -exp(out_trt))^2),
                      sqrt(mean((HR_est -exp(out_trt))^2)),
                      round(sum( HR_coverage_ind)/nsim*100,3),
                      mean(ci_width_uw),
                      mean(asam[,1]))

results_frame[2,] = c(mean((HR_est_wt -exp(out_trt))), #mean of bias
                      mean((HR_est_wt -exp(out_trt))^2),
                      sqrt(mean((HR_est_wt -exp(out_trt))^2)),
                      round(sum(HR_coverage_ind_wt)/nsim*100,3),
                      mean(ci_width),
                      mean(asam[,2]) )

results_frame[3,] = c(mean((HR_est_wt2 -exp(out_trt))), #mean of bias
                      mean((HR_est_wt2 -exp(out_trt))^2),
                      sqrt(mean((HR_est_wt2 -exp(out_trt))^2)),
                      round(sum(HR_coverage_ind_wt2)/nsim*100,3),
                      mean(ci_width2),
                      mean(asam[,3]) )

results_frame[4,] = c(mean((HR_est_ua -exp(out_trt))), #mean of bias
                      mean((HR_est_ua -exp(out_trt))^2),
                      sqrt(mean((HR_est_ua -exp(out_trt))^2)),
                      round(sum( HR_coverage_ind_ua)/nsim*100,3),
                      mean(ci_width_uw_ua),
                      mean(asam_ua[,1]))

results_frame[5,] = c(mean((HR_est_wt_ua -exp(out_trt))), #mean of bias
                      mean((HR_est_wt_ua -exp(out_trt))^2),
                      sqrt(mean((HR_est_wt_ua -exp(out_trt))^2)),
                      round(sum(HR_coverage_ind_wt_ua)/nsim*100,3),
                      mean(ci_width_ua),
                      mean(asam_ua[,2]) )

results_frame[6,] = c(mean((HR_est_wt2_ua -exp(out_trt))), #mean of bias
                      mean((HR_est_wt2_ua -exp(out_trt))^2),
                      sqrt(mean((HR_est_wt2_ua -exp(out_trt))^2)),
                      round(sum(HR_coverage_ind_wt2_ua)/nsim*100,3),
                      mean(ci_width2_ua),
```

```
            mean(asam_ua[,3]) )

round(results_frame,5)
save(results_frame, file = "~/R/summer2020/simulation script 3/sim1D_n1000_results_frame_logistic.RData")
save(asam, file="~/R/summer2020/simulation script 3/sim1D_n1000_asam_logistic.RData")
biases=cbind(HR_est -exp(out_trt), #adjusted models
      HR_est_wt -exp(out_trt),
      HR_est_wt2 -exp(out_trt),
      HR_est_ua -exp(out_trt), #unadjusted models
      HR_est_wt_ua -exp(out_trt),
      HR_est_wt2_ua -exp(out_trt))
save(biases, file="~/R/summer2020/simulation script 3/sim1D_n1000_biases_logistic.RData")

##### for adjusted cox ph model results
#####---------------------------------
## ignoring weights
HR_est = rep(NA,nsim)
HR_ci = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci) = c("lower","upper")
HR_coverage_ind = rep(NA,nsim)
ci_width_uw=rep(NA,nsim) ##width of CI

## incorporating weights: matching weights
HR_est_wt = rep(NA,nsim)
HR_ci_wt = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt) = c("lower","upper")
HR_coverage_ind_wt = rep(NA,nsim)
ci_width=rep(NA,nsim) #width of CI

## incorporating weights: overlapping weights
HR_est_wt2 = rep(NA,nsim)
HR_ci_wt2 = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt2) = c("lower","upper")
HR_coverage_ind_wt2 = rep(NA,nsim)
ci_width2=rep(NA,nsim) #width of CI

##ASAM
asam= matrix(0,nrow = nsim,ncol = 3)
colnames(asam) = c("unweighted","matching", "overlap")

##### for un-adjusted cox ph model results
#####---------------------------------
## ignoring weights
HR_est_ua = rep(NA,nsim)
#HR_mse = rep(NA,nsim)
HR_ci_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_ua) = c("lower","upper")
HR_coverage_ind_ua = rep(NA,nsim)
ci_width_uw_ua=rep(NA,nsim) ##width of CI

## incorporating weights: matching weights
HR_est_wt_ua = rep(NA,nsim)
HR_ci_wt_ua = matrix(0,nrow = nsim,ncol = 2)
```

```
colnames(HR_ci_wt_ua) = c("lower","upper")
HR_coverage_ind_wt_ua = rep(NA,nsim)
ci_width_ua=rep(NA,nsim) #width of CI


## incorporating weights: overlapping weights
HR_est_wt2_ua = rep(NA,nsim)
HR_ci_wt2_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt2_ua) = c("lower","upper")
HR_coverage_ind_wt2_ua = rep(NA,nsim)
ci_width2_ua=rep(NA,nsim) #width of CI


##from superlearner: coefficients for ML methods
model_coef=matrix(0, ncol=2, nrow=nsim)
colnames(model_coef)=c('RF', 'XGB')



##ASAM
asam_ua= matrix(0,nrow = nsim,ncol = 3)
colnames(asam_ua) = c("unweighted","matching", "overlap")



## =====================
## Simulation of PS and evaluation
## =====================



for(sim_num in 1:nsim){
  #print(paste("sim number: ",sim_num,sep = ""))



  ## ++++++++++++++++++++++++++++++++
  ## Fit cox proportional hazard model ignoring weights
  ## +++++

  surv_form<-as.formula(paste("Surv(time, status)~", "trt +", paste(vars_converted, collapse="+") ))
    ##the following code are not used because adjusted and unweighted results have been calculated at logistic truck above.
    #cox_mod <- coxph(surv_form, data =  dat_cox[[sim_num]])
    #cox_mod_sum = summary(cox_mod)
    #cox_mod_ci = exp(confint(cox_mod))
    #HR_est[sim_num] = cox_mod_sum$coefficients[1,2]
    #HR_ci[sim_num,] = cox_mod_ci[1,]
    #HR_coverage_ind[sim_num] = (exp(out_trt) >= HR_ci[sim_num,1] &exp(out_trt) <= HR_ci[sim_num,2])^2
    #ci_width_uw[sim_num]=HR_ci[sim_num,2]- HR_ci[sim_num,1]

#----------------------------------------
```

```
## fit model with weights (if n=1000)

x_train=dat_cox[[sim_num]][,vars_converted]
sl_multi = SuperLearner::SuperLearner(Y=dat_cox[[sim_num]]$trt, X=x_train,
                        family=binomial(), SL.library=c( "SL.randomForest", "SL.xgboost"))
multi_pred = predict(sl_multi, x_train, onlySL = TRUE)
prob_ps =multi_pred$pred
prob_ps[which(prob_ps==1)]=0.999


## fit model with weights (if n=100 or 500)
#x_train=dat_cox[[sim_num]][,vars_converted]
### create customized random Forest and XGBoost methods
#create_rf = create.Learner("SL.randomForest", params=list(ntree = 1000, nodeside=5, nPerm=2))
#create_xgb=create.Learner("SL.xgboost", params = list(nrounds=5), tune = list(eta=0.5, max_depth =10))
### apply SL function
#sl_multi = SuperLearner(Y = dat_cox[[sim_num]]$trt, X = x_train, family=binomial(),
        #  SL.library = c(create_rf$names, create_xgb$names))

multi_pred = predict(sl_multi, x_train, onlySL = TRUE)
prob_ps =multi_pred$pred
prob_ps[which(prob_ps==1)]=0.999



#matching weights and evaluation
w_ps = pmin(prob_ps,1-prob_ps)/(trt*prob_ps + (1-trt)*(1-prob_ps))
w_ps[which(w_ps==min(w_ps))]=0.0001 #for bug fixing
svyDes = svydesign(id=~0,weights=w_ps, data=dat_cox[[sim_num]])
cox_weighted_mod = svycoxph(surv_form, design = svyDes)

HR_est_wt[sim_num] = exp(cox_weighted_mod$coefficients[1])
HR_ci_wt[sim_num,] =  exp(confint(cox_weighted_mod))[1,]
HR_coverage_ind_wt[sim_num] = (exp(out_trt) >= HR_ci_wt[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt[sim_num,2])^2
ci_width[sim_num]=HR_ci_wt[sim_num,2]- HR_ci_wt[sim_num,1]
model_coef[sim_num, 1:2]= sl_multi$coef[1:2]

#overlap weights and evaluation
overlap_w_ps= trt + (-1)^trt * prob_ps
overlap_w_ps[which(overlap_w_ps==min(overlap_w_ps))]=0.0001 #for bug fixing
svyDes2 = svydesign(id=~0, weights=overlap_w_ps, data=dat_cox[[sim_num]])
cox_weighted_mod2 = svycoxph(surv_form, design = svyDes2)
HR_est_wt2[sim_num] = exp(cox_weighted_mod2$coefficients[1])
HR_ci_wt2[sim_num,] =  exp(confint(cox_weighted_mod2))[1,]
HR_coverage_ind_wt2[sim_num] = (exp(out_trt) >= HR_ci_wt2[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt2[sim_num,2])^2
ci_width2[sim_num]=HR_ci_wt2[sim_num,2]- HR_ci_wt2[sim_num,1]



 #ASAM
#importing user-supplied propensity scores "prob_ps"
```

```
msstat1 <- SumStat(zname="trt", xname=vars_converted, data=dat_cox[[sim_num]],
                   ps.estimate=prob_ps,
                   trtgrp="1",  weight="matching")
msstat2 <- SumStat(zname="trt", xname=vars_converted, data=dat_cox[[sim_num]],
                   ps.estimate=prob_ps,
                   trtgrp="1",  weight="overlap")

#ASAM for unweighted dataset
asam[sim_num,1]= mean(summary(msstat1)$unweighted[,"SMD"])
#ASAM for matching weight dataset
asam[sim_num,2]= mean(summary(msstat1)$matching[,"SMD"])
#ASAM for overlap weight dataset
asam[sim_num,3]= mean(summary(msstat2)$overlap[,"SMD"])


## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
####----------fit model with unadjusted cox ph mode #####
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


## Fit cox proportional hazard model ignoring weights

surv_form_ua<-as.formula(paste("Surv(time, status)~trt" ))
  ##the following code are not used because adjusted and unweighted results have been calculated at logistic truck above.
  #cox_mod <- coxph(surv_form_ua, data =  dat_cox[[sim_num]])
  #cox_mod_sum = summary(cox_mod)
  #cox_mod_ci = exp(confint(cox_mod))
  #HR_est_ua[sim_num] = cox_mod_sum$coefficients[1,2]
  #HR_ci_ua[sim_num,] = cox_mod_ci[1,]
  #HR_coverage_ind_ua[sim_num] = (exp(out_trt) >= HR_ci_ua[sim_num,1] &exp(out_trt) <=
HR_ci_ua[sim_num,2])^2
  #ci_width_uw_ua[sim_num]=HR_ci_ua[sim_num,2]- HR_ci_ua[sim_num,1]



## fit model with weights
#matching weights and evaluation

svyDes = svydesign(id=~0, weights=w_ps, data=dat_cox[[sim_num]])
cox_weighted_mod_ua = svycoxph(surv_form_ua, design = svyDes)
HR_est_wt_ua[sim_num] = exp(cox_weighted_mod_ua$coefficients[1])
HR_ci_wt_ua[sim_num,] =  exp(confint(cox_weighted_mod_ua))[1,]
HR_coverage_ind_wt_ua[sim_num] = (exp(out_trt) >= HR_ci_wt_ua[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt_ua[sim_num,2])^2
ci_width_ua[sim_num]=HR_ci_wt_ua[sim_num,2]- HR_ci_wt_ua[sim_num,1]
model_coef_ua[sim_num, 1:2]= sl_multi$coef[1:2]

#overlap weights and evaluation
svyDes2 = svydesign(id=~0, weights=overlap_w_ps, data=dat_cox[[sim_num]])
cox_weighted_mod2_ua = svycoxph(surv_form_ua, design = svyDes2)
HR_est_wt2_ua[sim_num] = exp(cox_weighted_mod2_ua$coefficients[1])
HR_ci_wt2_ua[sim_num,] =  exp(confint(cox_weighted_mod2_ua))[1,]
HR_coverage_ind_wt2_ua[sim_num] = (exp(out_trt) >= HR_ci_wt2_ua[sim_num,1] &
```

```
                        exp(out_trt) <= HR_ci_wt2_ua[sim_num,2])^2
 ci_width2_ua[sim_num]=HR_ci_wt2_ua[sim_num,2]- HR_ci_wt2_ua[sim_num,1]

 #ASAM
 # using SumStat to estimate propensity scores
 msstat1_ua <- SumStat(zname="trt", xname=vars_converted, data=dat_cox[[sim_num]],
              ps.estimate=prob_ps,
              trtgrp="1",  weight="matching")
 msstat2_ua<- SumStat(zname="trt", xname=vars_converted, data=dat_cox[[sim_num]],
              ps.estimate=prob_ps,
              trtgrp="1",  weight="overlap")

 #ASAM for unweighted dataset
 asam_ua[sim_num,1]= mean(summary(msstat1_ua)$unweighted[,"SMD"])
 #ASAM for matching weight dataset
 asam_ua[sim_num,2]= mean(summary(msstat1_ua)$matching[,"SMD"])
 #ASAM for overlap weight dataset
 asam_ua[sim_num,3]= mean(summary(msstat2_ua)$overlap[,"SMD"])

}




## =====================
## Summarize Simulation Results
## =====================
results_frame = data.frame(matrix(0,nrow = 6,ncol = 6))
colnames(results_frame) = c("Bias", "MSE","RMSE","% CI coverage", "CI width","ASAM")
rownames(results_frame) = c("no weights","matching weights", "overlap weights",
                  "no weights (unadj)","matching weights (unadj)", "overlap weights (unadj)")
results_frame[1,] = c(mean((HR_est -exp(out_trt))), #mean of bias
              mean((HR_est -exp(out_trt))^2),
              sqrt(mean((HR_est -exp(out_trt))^2)),
              round(sum( HR_coverage_ind)/nsim*100,3),
              mean(ci_width_uw),
              mean(asam[,1]))

results_frame[2,] = c(mean((HR_est_wt -exp(out_trt))), #mean of bias
              mean((HR_est_wt -exp(out_trt))^2),
              sqrt(mean((HR_est_wt -exp(out_trt))^2)),
              round(sum(HR_coverage_ind_wt)/nsim*100,3),
              mean(ci_width),
              mean(asam[,2]) )
results_frame[3,] = c(mean((HR_est_wt2 -exp(out_trt))), #mean of bias
              mean((HR_est_wt2 -exp(out_trt))^2),
              sqrt(mean((HR_est_wt2 -exp(out_trt))^2)),
              round(sum(HR_coverage_ind_wt2)/nsim*100,3),
              mean(ci_width2),
              mean(asam[,3]) )

results_frame[4,] = c(mean((HR_est_ua -exp(out_trt))), #mean of bias
              mean((HR_est_ua -exp(out_trt))^2),
```

```
                sqrt(mean((HR_est_ua -exp(out_trt))^2)),
                round(sum( HR_coverage_ind_ua)/nsim*100,3),
                mean(ci_width_uw_ua),
                mean(asam_ua[,1]))

results_frame[5,] = c(mean((HR_est_wt_ua -exp(out_trt))), #mean of bias
                mean((HR_est_wt_ua -exp(out_trt))^2),
                sqrt(mean((HR_est_wt_ua -exp(out_trt))^2)),
                round(sum(HR_coverage_ind_wt_ua)/nsim*100,3),
                mean(ci_width_ua),
                mean(asam_ua[,2]) )

results_frame[6,] = c(mean((HR_est_wt2_ua -exp(out_trt))), #mean of bias
                mean((HR_est_wt2_ua -exp(out_trt))^2),
                sqrt(mean((HR_est_wt2_ua -exp(out_trt))^2)),
                round(sum(HR_coverage_ind_wt2_ua)/nsim*100,3),
                mean(ci_width2_ua),
                mean(asam_ua[,3]) )

round(results_frame,5)
save(results_frame, file = "~/R/summer2020/simulation script 3/sim1D_n1000_results_frame_SL.RData")
save(asam, file="~/R/summer2020/simulation script 3/sim1D_n1000_asam_SL.RData")
biases=cbind(HR_est -exp(out_trt), #adjusted models
        HR_est_wt -exp(out_trt),
        HR_est_wt2 -exp(out_trt),
        HR_est_ua -exp(out_trt), #unadjusted models
        HR_est_wt_ua -exp(out_trt),
        HR_est_wt2_ua -exp(out_trt))
save(biases, file="~/R/summer2020/simulation script 3/sim1D_n1000_biases_SL.RData")

colMeans(model_coef) #the frequency of ML method chosen for each time
colMeans(model_coef_ua) #the frequency of ML method chosen for each time

end.time = Sys.time()
run_time = end.time - start.time
start.time; end.time; run_time
```

6.4 Appendix D. R Codes for Method Application on the RHC Dataset Code

```
knitr::opts_chunk$set(echo = TRUE)

## ======================
## packages used
## ======================
library(survival) ## coxph
library(survey)  ## svydesign and svycoxph
library(simsurv) ## simsurv

#packages for SuperLearner function
library(SuperLearner)
library(caret)
library(glmnet)
library(randomForest)
library(RhpcBLASctl)
library(xgboost)

#package for ASAM
library(PSweight)

## ======================
## simulation parameters
## ======================
start.time = Sys.time()
seed=0123
#seed=345 #original seeddd
set.seed(seed)
rhc.dat = read.csv("~/R/summer2020/rhc.csv", stringsAsFactors=T)
n= nrow(rhc.dat)## size of data set
nsim = 10 ## number of simulated data sets


## ======================
## Create object to save results
## ======================
p_trt = rep(NA,nsim)
event_rate = rep(NA,nsim)
times_c<-vector(mode="list", length=nsim)
times_uc<-vector(mode="list", length=nsim)
status_list<-vector(mode="list", length=nsim)

##simulated datasets
dat_cox<-vector(mode="list", length=nsim)

##### for adjusted cox ph model results
## ignoring weights
HR_est = rep(NA,nsim)
#HR_mse = rep(NA,nsim)
HR_ci = matrix(0,nrow = nsim,ncol = 2)
```

```
colnames(HR_ci) = c("lower","upper")
HR_coverage_ind = rep(NA,nsim)
ci_width_uw=rep(NA,nsim) ##width of CI

## incorporating weights: matching weights
HR_est_wt = rep(NA,nsim)
HR_ci_wt = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt) = c("lower","upper")
HR_coverage_ind_wt = rep(NA,nsim)
ci_width=rep(NA,nsim) #width of CI

## incorporating weights: overlapping weights
HR_est_wt2 = rep(NA,nsim)
HR_ci_wt2 = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt2) = c("lower","upper")
HR_coverage_ind_wt2 = rep(NA,nsim)
ci_width2=rep(NA,nsim) #width of CI


##from superlearner: coefficients for each ML methods
model_coef=matrix(0, ncol=2, nrow=nsim)
colnames(model_coef)=c('RF', 'XGB')



##ASAM
asam= matrix(0,nrow = nsim,ncol = 3)
colnames(asam) = c("unweighted","matching", "overlap")


##### for unadjusted cox ph model results
## ignoring weights
HR_est_ua = rep(NA,nsim)
#HR_mse = rep(NA,nsim)
HR_ci_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_ua) = c("lower","upper")
HR_coverage_ind_ua = rep(NA,nsim)
ci_width_uw_ua=rep(NA,nsim) ##width of CI

## incorporating weights: matching weights
HR_est_wt_ua = rep(NA,nsim)
HR_ci_wt_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt_ua) = c("lower","upper")
HR_coverage_ind_wt_ua = rep(NA,nsim)
ci_width_ua=rep(NA,nsim) #width of CI


## incorporating weights: overlapping weights
HR_est_wt2_ua = rep(NA,nsim)
HR_ci_wt2_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt2_ua) = c("lower","upper")
HR_coverage_ind_wt2_ua = rep(NA,nsim)
ci_width2_ua=rep(NA,nsim) #width of CI
```

```
##from superlearner
model_coef_ua=matrix(0, ncol=2, nrow=nsim)
colnames(model_coef_ua)=c('RF', 'XGB')

##ASAM
asam_ua= matrix(0,nrow = nsim,ncol = 3)
colnames(asam_ua) = c("unweighted","matching", "overlap")



## =====================
###set up: manage RHC dataset
## =====================

vars_selected<-c("pafi1", "age", "hrt1", "resp1", "bili1", "dnr1", "card", "ninsclas")
vars_converted<-c("pafi1", "age", "hrt1", "resp1", "bili1", "dnr1", "card",
                  "private", "medicare", "priNcare", "other")

ps.formula<- trt~pafi1+age+hrt1+resp1+bili1+dnr1+card+private+medicare+priNcare+other

rhc.X<-cbind(rhc.dat[,vars_selected])

##convert factor variables in rhc.cov into indicator variables
rhc.conv<- model.matrix(~., data= rhc.X)[,-1]
rhc.conv<-cbind(rhc.conv, rhc.dat$swang1) #add trt var into the db
rhc.conv<-data.frame(rhc.conv)
rhc.conv[,14]<-rhc.conv[,11]
rhc.conv[,15]<-rhc.conv[,8]
rhc.conv[,16]<-rhc.conv[,12]
rhc.conv[,17]<-rhc.conv[,9]+rhc.conv[,10]
rhc.conv<-cbind(rhc.conv[,1:7],rhc.conv[,13:17])
colnames(rhc.conv)[6:12]<-c("dnr1", "card", "trt", "private", "medicare", "priNcare", "other")
rhc.conv$trt[which(rhc.conv$trt==1)]<-0
rhc.conv$trt[which(rhc.conv$trt==2)]<-1
#head(rhc.conv)

## =====================
## set up: get outcome model coefficients ("true" model)
## =====================

##get survival time of RHC
survtime<-ifelse(is.na(rhc.dat$dthdte), rhc.dat$lstctdte-rhc.dat$sadmdte, rhc.dat$dthdte-rhc.dat$sadmdte)
rhc.conv$survtime<-survtime
death<-(rhc.dat$death=="Yes")^2
rhc.conv$death<-death

surv_form<-as.formula(paste("Surv(survtime, death)~", "trt +", paste(vars_converted, collapse="+")) )
surv_mod<-coxph(surv_form, data=rhc.conv)
beta_out_mod<-surv_mod$coefficients
beta_out_mod
out_trt<-beta_out_mod[1] #get HR of treatment=hazard of exposed/ hazard of unexposed
out_trt
```

```
exp(out_trt)

dat_cox=rhc.conv
colnames(dat_cox)[c(ncol(rhc.conv)-1,ncol(rhc.conv))]=c("time","status")


for(sim_num in 1:nsim){



## ++++++++++++++++++++
 ## Fit cox proportional hazard model ignoring weights


 surv_form<-as.formula(paste("Surv(time, status)~", "trt +", paste(vars_converted, collapse="+") ))
 cox_mod <- coxph(surv_form, data =  dat_cox)
 cox_mod_sum = summary(cox_mod)
 cox_mod_ci = exp(confint(cox_mod))
 HR_est[sim_num] = cox_mod_sum$coefficients[1,2]
 HR_ci[sim_num,] = cox_mod_ci[1,]
 HR_coverage_ind[sim_num] = (exp(out_trt) >= HR_ci[sim_num,1] &
                        exp(out_trt) <= HR_ci[sim_num,2])^2
 ci_width_uw[sim_num]=HR_ci[sim_num,2]- HR_ci[sim_num,1]


#------------------------

 ## fit model with weights
 log_mod = glm(as.formula(paste("trt~", paste(vars_converted, collapse="+")) ),
           data = dat_cox, family = "binomial")
 logit_ps = predict(log_mod)
 prob_ps =  exp(logit_ps)/(1+exp(logit_ps))
 prob_ps[which(prob_ps==1)]=0.999

 #matching weights and evaluation
 trt=dat_cox$trt
 w_ps = pmin(prob_ps,1-prob_ps)/(trt*prob_ps + (1-trt)*(1-prob_ps))
 svyDes = svydesign(id=~0, weights=w_ps, data=dat_cox)
 cox_weighted_mod = svycoxph(surv_form, design = svyDes)
 HR_est_wt[sim_num] = exp(cox_weighted_mod$coefficients[1])
 HR_ci_wt[sim_num,] =  exp(confint(cox_weighted_mod))[1,]
 HR_coverage_ind_wt[sim_num] = (exp(out_trt) >= HR_ci_wt[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt[sim_num,2])^2
 ci_width[sim_num]=HR_ci_wt[sim_num,2]- HR_ci_wt[sim_num,1]

 #overlap weights and evaluation
 overlap_w_ps= trt + (-1)^trt * prob_ps
 svyDes2 = svydesign(id=~0, weights=overlap_w_ps, data=dat_cox)
 cox_weighted_mod2 = svycoxph(surv_form, design = svyDes2)
 HR_est_wt2[sim_num] = exp(cox_weighted_mod2$coefficients[1])
 HR_ci_wt2[sim_num,] =  exp(confint(cox_weighted_mod2))[1,]
 HR_coverage_ind_wt2[sim_num] = (exp(out_trt) >= HR_ci_wt2[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt2[sim_num,2])^2
```

```
ci_width2[sim_num]=HR_ci_wt2[sim_num,2]- HR_ci_wt2[sim_num,1]

#ASAM
msstat1 <- SumStat(zname="trt", xname=vars_converted, data=dat_cox,
                   ps.estimate=prob_ps,
                   trtgrp="1",  weight="matching")
msstat2 <- SumStat(zname="trt", xname=vars_converted, data=dat_cox,
                   ps.estimate=prob_ps,
                   trtgrp="1",  weight="overlap")

#ASAM for unweighted dataset
asam[sim_num,1]= mean(summary(msstat1)$unweighted[,"SMD"])
#ASAM for matching weight dataset
asam[sim_num,2]= mean(summary(msstat1)$matching[,"SMD"])
#ASAM for overlap weight dataset
asam[sim_num,3]= mean(summary(msstat2)$overlap[,"SMD"])



## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
####----------fit model with unadjusted cox ph mode #####
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


## Fit cox proportional hazard model ignoring weights

surv_form_ua<-as.formula(paste("Surv(time, status)~trt" ))
cox_mod <- coxph(surv_form_ua, data =  dat_cox)
cox_mod_sum = summary(cox_mod)
cox_mod_ci = exp(confint(cox_mod))
HR_est_ua[sim_num] = cox_mod_sum$coefficients[1,2]
HR_ci_ua[sim_num,] = cox_mod_ci[1,]
HR_coverage_ind_ua[sim_num] = (exp(out_trt) >= HR_ci_ua[sim_num,1] &
                        exp(out_trt) <= HR_ci_ua[sim_num,2])^2
ci_width_uw_ua[sim_num]=HR_ci_ua[sim_num,2]- HR_ci_ua[sim_num,1]



## fit model with weights
#matching weights and evaluation

svyDes = svydesign(id=~0, weights=w_ps, data=dat_cox)
cox_weighted_mod_ua = svycoxph(surv_form_ua, design = svyDes)
HR_est_wt_ua[sim_num] = exp(cox_weighted_mod_ua$coefficients[1])
HR_ci_wt_ua[sim_num,] =  exp(confint(cox_weighted_mod_ua))[1,]
HR_coverage_ind_wt_ua[sim_num] = (exp(out_trt) >= HR_ci_wt_ua[sim_num,1] &
                          exp(out_trt) <= HR_ci_wt_ua[sim_num,2])^2
ci_width_ua[sim_num]=HR_ci_wt_ua[sim_num,2]- HR_ci_wt_ua[sim_num,1]

#overlap weights and evaluation
svyDes2 = svydesign(id=~0, weights=overlap_w_ps, data=dat_cox)
cox_weighted_mod2_ua = svycoxph(surv_form_ua, design = svyDes2)
HR_est_wt2_ua[sim_num] = exp(cox_weighted_mod2_ua$coefficients[1])
HR_ci_wt2_ua[sim_num,] =  exp(confint(cox_weighted_mod2_ua))[1,]
```

```
 HR_coverage_ind_wt2_ua[sim_num] = (exp(out_trt) >= HR_ci_wt2_ua[sim_num,1] &
                          exp(out_trt) <= HR_ci_wt2_ua[sim_num,2])^2
 ci_width2_ua[sim_num]=HR_ci_wt2_ua[sim_num,2]- HR_ci_wt2_ua[sim_num,1]

 #ASAM
 msstat1_ua <- SumStat(zname="trt", xname=vars_converted, data=dat_cox,
               ps.estimate=prob_ps,
               trtgrp="1",  weight="matching")
 msstat2_ua<- SumStat(zname="trt", xname=vars_converted, data=dat_cox,
               ps.estimate=prob_ps,
               trtgrp="1",  weight="overlap")

 #ASAM for unweighted dataset
 asam_ua[sim_num,1]= mean(summary(msstat1_ua)$unweighted[,"SMD"])
 #ASAM for matching weight dataset
 asam_ua[sim_num,2]= mean(summary(msstat1_ua)$matching[,"SMD"])
 #ASAM for overlap weight dataset
 asam_ua[sim_num,3]= mean(summary(msstat2_ua)$overlap[,"SMD"])

}




## =====================
## Summarize Simulation Results
## =====================
results_frame = data.frame(matrix(0,nrow = 6,ncol = 6))
colnames(results_frame) = c("Bias", "MSE","RMSE","% CI coverage", "CI width","ASAM")
rownames(results_frame) = c("no weights","matching weights", "overlap weights",
                    "no weights (unadj)","matching weights (unadj)", "overlap weights (unadj)")
results_frame[1,] = c(mean((HR_est -exp(out_trt))), #mean of bias
                mean((HR_est -exp(out_trt))^2),
                sqrt(mean((HR_est -exp(out_trt))^2)),
                round(sum( HR_coverage_ind)/nsim*100,3),
                mean(ci_width_uw),
                mean(asam[,1]))

results_frame[2,] = c(mean((HR_est_wt -exp(out_trt))), #mean of bias
                mean((HR_est_wt -exp(out_trt))^2),
                sqrt(mean((HR_est_wt -exp(out_trt))^2)),
                round(sum(HR_coverage_ind_wt)/nsim*100,3),
                mean(ci_width),
                mean(asam[,2]) )

results_frame[3,] = c(mean((HR_est_wt2 -exp(out_trt))), #mean of bias
                mean((HR_est_wt2 -exp(out_trt))^2),
                sqrt(mean((HR_est_wt2 -exp(out_trt))^2)),
                round(sum(HR_coverage_ind_wt2)/nsim*100,3),
                mean(ci_width2),
                mean(asam[,3]) )
```

```
results_frame[4,] = c(mean((HR_est_ua -exp(out_trt))), #mean of bias
                      mean((HR_est_ua -exp(out_trt))^2),
                      sqrt(mean((HR_est_ua -exp(out_trt))^2)),
                      round(sum( HR_coverage_ind_ua)/nsim*100,3),
                      mean(ci_width_uw_ua),
                      mean(asam_ua[,1]))

results_frame[5,] = c(mean((HR_est_wt_ua -exp(out_trt))), #mean of bias
                      mean((HR_est_wt_ua -exp(out_trt))^2),
                      sqrt(mean((HR_est_wt_ua -exp(out_trt))^2)),
                      round(sum(HR_coverage_ind_wt_ua)/nsim*100,3),
                      mean(ci_width_ua),
                      mean(asam_ua[,2]) )

results_frame[6,] = c(mean((HR_est_wt2_ua -exp(out_trt))), #mean of bias
                      mean((HR_est_wt2_ua -exp(out_trt))^2),
                      sqrt(mean((HR_est_wt2_ua -exp(out_trt))^2)),
                      round(sum(HR_coverage_ind_wt2_ua)/nsim*100,3),
                      mean(ci_width2_ua),
                      mean(asam_ua[,3]) )

round(results_frame,5)
save(results_frame, file = "~/R/summer2020/simulation script 2/sim1A_RHC_results_frame_logistic.RData")

## estimated HRs of treatment from fitted models
round(cbind(mean(HR_est),       #adjusted unweighted
mean(HR_est_wt),     #adjusted, matching weight
mean(HR_est_wt2),    #adjusted, overlap weight
mean(HR_est_ua),     #unadjusted unweighted
mean(HR_est_wt_ua),  #unadjusted matching weight
mean(HR_est_wt2_ua)), 3) #unadjusted overlap weight

##### for adjusted cox ph model results
#####--------------------------------
## ignoring weights
HR_est = rep(NA,nsim)
HR_ci = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci) = c("lower","upper")
HR_coverage_ind = rep(NA,nsim)
ci_width_uw=rep(NA,nsim) ##width of CI

## incorporating weights: matching weights
HR_est_wt = rep(NA,nsim)
HR_ci_wt = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt) = c("lower","upper")
HR_coverage_ind_wt = rep(NA,nsim)
ci_width=rep(NA,nsim) #width of CI

## incorporating weights: overlapping weights
HR_est_wt2 = rep(NA,nsim)
HR_ci_wt2 = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt2) = c("lower","upper")
HR_coverage_ind_wt2 = rep(NA,nsim)
```

```
ci_width2=rep(NA,nsim) #width of CI

##ASAM
asam= matrix(0,nrow = nsim,ncol = 3)
colnames(asam) = c("unweighted","matching", "overlap")

##### for un-adjusted cox ph model results
#####----------------------------------
## ignoring weights
HR_est_ua = rep(NA,nsim)
#HR_mse = rep(NA,nsim)
HR_ci_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_ua) = c("lower","upper")
HR_coverage_ind_ua = rep(NA,nsim)
ci_width_uw_ua=rep(NA,nsim) ##width of CI

## incorporating weights: matching weights
HR_est_wt_ua = rep(NA,nsim)
HR_ci_wt_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt_ua) = c("lower","upper")
HR_coverage_ind_wt_ua = rep(NA,nsim)
ci_width_ua=rep(NA,nsim) #width of CI


## incorporating weights: overlapping weights
HR_est_wt2_ua = rep(NA,nsim)
HR_ci_wt2_ua = matrix(0,nrow = nsim,ncol = 2)
colnames(HR_ci_wt2_ua) = c("lower","upper")
HR_coverage_ind_wt2_ua = rep(NA,nsim)
ci_width2_ua=rep(NA,nsim) #width of CI


##from superlearner: coefficients for ML methods
model_coef=matrix(0, ncol=2, nrow=nsim)
colnames(model_coef)=c('RF', 'XGB')



##ASAM
asam_ua= matrix(0,nrow = nsim,ncol = 3)
colnames(asam_ua) = c("unweighted","matching", "overlap")



## ======================
## Simulation of PS and evaluation
## ======================



for(sim_num in 1:nsim){
```

```
## ++++++++++++++++++++++++++++++++
## Fit cox proportional hazard model ignoring weights
## +++++

surv_form<-as.formula(paste("Surv(time, status)~", "trt +", paste(vars_converted, collapse="+") ))
cox_mod <- coxph(surv_form, data =  dat_cox)
cox_mod_sum = summary(cox_mod)
cox_mod_ci = exp(confint(cox_mod))
HR_est[sim_num] = cox_mod_sum$coefficients[1,2]
HR_ci[sim_num,] = cox_mod_ci[1,]
HR_coverage_ind[sim_num] = (exp(out_trt) >= HR_ci[sim_num,1] &
                        exp(out_trt) <= HR_ci[sim_num,2])^2
ci_width_uw[sim_num]=HR_ci[sim_num,2]- HR_ci[sim_num,1]

#----------------------------------------

## fit model with weights

  x_train=dat_cox[,vars_converted]
sl_multi = SuperLearner::SuperLearner(Y=dat_cox$trt, X=x_train,
                        family=binomial(), SL.library=c( "SL.randomForest", "SL.xgboost"))
multi_pred = predict(sl_multi, x_train, onlySL = TRUE)
prob_ps =multi_pred$pred
prob_ps[which(prob_ps==1)]=0.999

#matching weights and evaluation
w_ps = pmin(prob_ps,1-prob_ps)/(trt*prob_ps + (1-trt)*(1-prob_ps))
svyDes = svydesign(id=~0,weights=w_ps, data=dat_cox)
cox_weighted_mod = svycoxph(surv_form, design = svyDes)

HR_est_wt[sim_num] = exp(cox_weighted_mod$coefficients[1])
HR_ci_wt[sim_num,] =  exp(confint(cox_weighted_mod))[1,]
HR_coverage_ind_wt[sim_num] = (exp(out_trt) >= HR_ci_wt[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt[sim_num,2])^2
ci_width[sim_num]=HR_ci_wt[sim_num,2]- HR_ci_wt[sim_num,1]
model_coef[sim_num, 1:2]= sl_multi$coef[1:2]

#overlap weights and evaluation
overlap_w_ps= trt + (-1)^trt * prob_ps
svyDes2 = svydesign(id=~0, weights=overlap_w_ps, data=dat_cox)
cox_weighted_mod2 = svycoxph(surv_form, design = svyDes2)
HR_est_wt2[sim_num] = exp(cox_weighted_mod2$coefficients[1])
HR_ci_wt2[sim_num,] =  exp(confint(cox_weighted_mod2))[1,]
HR_coverage_ind_wt2[sim_num] = (exp(out_trt) >= HR_ci_wt2[sim_num,1] &
                        exp(out_trt) <= HR_ci_wt2[sim_num,2])^2
ci_width2[sim_num]=HR_ci_wt2[sim_num,2]- HR_ci_wt2[sim_num,1]



 #ASAM
```

```
#importing user-supplied propensity scores "prob_ps"
msstat1 <- SumStat(zname="trt", xname=vars_converted, data=dat_cox,
                   ps.estimate=prob_ps,
                   trtgrp="1",  weight="matching")
msstat2 <- SumStat(zname="trt", xname=vars_converted, data=dat_cox,
                   ps.estimate=prob_ps,
                   trtgrp="1",  weight="overlap")

#ASAM for unweighted dataset
asam[sim_num,1]= mean(summary(msstat1)$unweighted[,"SMD"])
#ASAM for matching weight dataset
asam[sim_num,2]= mean(summary(msstat1)$matching[,"SMD"])
#ASAM for overlap weight dataset
asam[sim_num,3]= mean(summary(msstat2)$overlap[,"SMD"])


## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
####----------fit model with unadjusted cox ph mode #####
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


## Fit cox proportional hazard model ignoring weights

surv_form_ua<-as.formula(paste("Surv(time, status)~trt" ))
cox_mod <- coxph(surv_form_ua, data =  dat_cox)
cox_mod_sum = summary(cox_mod)
cox_mod_ci = exp(confint(cox_mod))
HR_est_ua[sim_num] = cox_mod_sum$coefficients[1,2]
HR_ci_ua[sim_num,] = cox_mod_ci[1,]
HR_coverage_ind_ua[sim_num] = (exp(out_trt) >= HR_ci_ua[sim_num,1] &
                      exp(out_trt) <= HR_ci_ua[sim_num,2])^2
ci_width_uw_ua[sim_num]=HR_ci_ua[sim_num,2]- HR_ci_ua[sim_num,1]



## fit model with weights
#matching weights and evaluation

svyDes = svydesign(id=~0, weights=w_ps, data=dat_cox)
cox_weighted_mod_ua = svycoxph(surv_form_ua, design = svyDes)
HR_est_wt_ua[sim_num] = exp(cox_weighted_mod_ua$coefficients[1])
HR_ci_wt_ua[sim_num,] =  exp(confint(cox_weighted_mod_ua))[1,]
HR_coverage_ind_wt_ua[sim_num] = (exp(out_trt) >= HR_ci_wt_ua[sim_num,1] &
                         exp(out_trt) <= HR_ci_wt_ua[sim_num,2])^2
ci_width_ua[sim_num]=HR_ci_wt_ua[sim_num,2]- HR_ci_wt_ua[sim_num,1]
model_coef_ua[sim_num, 1:2]= sl_multi$coef[1:2]

#overlap weights and evaluation
svyDes2 = svydesign(id=~0, weights=overlap_w_ps, data=dat_cox)
cox_weighted_mod2_ua = svycoxph(surv_form_ua, design = svyDes2)
HR_est_wt2_ua[sim_num] = exp(cox_weighted_mod2_ua$coefficients[1])
HR_ci_wt2_ua[sim_num,] =  exp(confint(cox_weighted_mod2_ua))[1,]
HR_coverage_ind_wt2_ua[sim_num] = (exp(out_trt) >= HR_ci_wt2_ua[sim_num,1] &
```

```
                        exp(out_trt) <= HR_ci_wt2_ua[sim_num,2])^2
 ci_width2_ua[sim_num]=HR_ci_wt2_ua[sim_num,2]- HR_ci_wt2_ua[sim_num,1]

 #ASAM
 # using SumStat to estimate propensity scores
 msstat1_ua <- SumStat(zname="trt", xname=vars_converted, data=dat_cox,
              ps.estimate=prob_ps,
              trtgrp="1",  weight="matching")
 msstat2_ua<- SumStat(zname="trt", xname=vars_converted, data=dat_cox,
              ps.estimate=prob_ps,
              trtgrp="1",  weight="overlap")

 #ASAM for unweighted dataset
 asam_ua[sim_num,1]= mean(summary(msstat1_ua)$unweighted[,"SMD"])
 #ASAM for matching weight dataset
 asam_ua[sim_num,2]= mean(summary(msstat1_ua)$matching[,"SMD"])
 #ASAM for overlap weight dataset
 asam_ua[sim_num,3]= mean(summary(msstat2_ua)$overlap[,"SMD"])

}




## =====================
## Summarize Simulation Results
## =====================
results_frame = data.frame(matrix(0,nrow = 6,ncol = 6))
colnames(results_frame) = c("Bias", "MSE","RMSE","% CI coverage", "CI width","ASAM")
rownames(results_frame) = c("no weights","matching weights", "overlap weights",
                    "no weights (unadj)","matching weights (unadj)", "overlap weights (unadj)")
results_frame[1,] = c(mean((HR_est -exp(out_trt))), #mean of bias
              mean((HR_est -exp(out_trt))^2),
              sqrt(mean((HR_est -exp(out_trt))^2)),
              round(sum( HR_coverage_ind)/nsim*100,3),
              mean(ci_width_uw),
              mean(asam[,1]))

results_frame[2,] = c(mean((HR_est_wt -exp(out_trt))), #mean of bias
              mean((HR_est_wt -exp(out_trt))^2),
              sqrt(mean((HR_est_wt -exp(out_trt))^2)),
              round(sum(HR_coverage_ind_wt)/nsim*100,3),
              mean(ci_width),
              mean(asam[,2]) )
results_frame[3,] = c(mean((HR_est_wt2 -exp(out_trt))), #mean of bias
              mean((HR_est_wt2 -exp(out_trt))^2),
              sqrt(mean((HR_est_wt2 -exp(out_trt))^2)),
              round(sum(HR_coverage_ind_wt2)/nsim*100,3),
              mean(ci_width2),
              mean(asam[,3]) )

results_frame[4,] = c(mean((HR_est_ua -exp(out_trt))), #mean of bias
              mean((HR_est_ua -exp(out_trt))^2),
```

```
                sqrt(mean((HR_est_ua -exp(out_trt))^2)),
                round(sum( HR_coverage_ind_ua)/nsim*100,3),
                mean(ci_width_uw_ua),
                mean(asam_ua[,1]))

results_frame[5,] = c(mean((HR_est_wt_ua -exp(out_trt))), #mean of bias
                mean((HR_est_wt_ua -exp(out_trt))^2),
                sqrt(mean((HR_est_wt_ua -exp(out_trt))^2)),
                round(sum(HR_coverage_ind_wt_ua)/nsim*100,3),
                mean(ci_width_ua),
                mean(asam_ua[,2]) )

results_frame[6,] = c(mean((HR_est_wt2_ua -exp(out_trt))), #mean of bias
                mean((HR_est_wt2_ua -exp(out_trt))^2),
                sqrt(mean((HR_est_wt2_ua -exp(out_trt))^2)),
                round(sum(HR_coverage_ind_wt2_ua)/nsim*100,3),
                mean(ci_width2_ua),
                mean(asam_ua[,3]) )

round(results_frame,5)
save(results_frame,
    file = "~/R/summer2020/simulation script 2/sim1A_n1000_results_frame_SL.RData")

colMeans(model_coef) #the frequency of ML method chosen for each time
colMeans(model_coef_ua) #the frequency of ML method chosen for each time

## estimated HRs of treatment from fitted models
round(cbind(mean(HR_est),       #adjusted unweighted
mean(HR_est_wt),     #adjusted, matching weight
mean(HR_est_wt2),    #adjusted, overlap weight
mean(HR_est_ua),     #unadjusted unweighted
mean(HR_est_wt_ua),  #unadjusted matching weight
mean(HR_est_wt2_ua)), 3) #unadjusted overlap weight

end.time = Sys.time()
run_time = end.time - start.time
start.time; end.time; run_time
```