Metalearners and Confidence Intervals for CATE Estimation
DS-GA 3001 Final Project

Lauren D'Arinzo (lhd258), Kelsey Markey (kcm312),
Tamar Novetsky (trn219), Alene Rhea (akr435)

https://github.com/kelseymarkey/metalearners

In this project, we reproduce the synthetic data experiments and compare results to those described in the paper "Metalearners for estimating heterogeneous treatment effects using machine learning" (Kunzel et al. 2019). The authors propose a meta-algorithm, called the "X-learner," for using machine learning models to estimate treatment effects (TE) as a function of given covariates. We replicate their methods with our own implementation of data generation, metalearner conditional average treatment effect (CATE) estimation, and confidence intervals around predicted CATEs, finding similar results. We subsequently experiment with additional data generating distributions, variations in base learner fitting and hyperparameter tuning, and additional confidence interval methods.

I. Causal Estimation Background and Notation

In the subsequent discussion of our work, we will be consistent with much of the notation from David Rosenberg's lecture notes 'Conditional Average Treatment Effects'. Recall that in the TE setting:
For an individual i,

- $Y_i(1) \in \mathbb{R}$ is the potential outcome if i is given the treatment
- $Y_i(0) \in \mathbb{R}$ is the potential outcome if i is not given the treatment
- $W_i = \mathbb{I}\left[\text{individual } i \text{ received the treatment}\right]$ is the treatment indicator
- $X_i \in X$ is the given covariate vector for individual i
- $\pi(x) = P(W = 1 \mid X = x) \in [0, 1]$ is the propensity score which represents likelihood of being assigned the treatment

The notable challenge of solving TE problems in the real world is that we only ever observe $Y(1)$ or $Y(0)$, never both. We would like to estimate the CATE denoted by τ:

- $\tau(x) = \mu_1(x) - \mu_0(x)$, where
  - $\mu_1(x) = \mathbb{E}[Y(1)|X = x]$
  - $\mu_0(x) = \mathbb{E}[Y(0)|X = x]$

II. Data Simulation

Kunzel et al. generate simulated data to test the performance of their proposed metalearners on different generating distributions. In this synthetic setting, data is simulated with

both $Y(1)$ and $Y(0)$ for each instance in order to compare the predicted CATE to a ground-truth CATE and evaluate the efficacy of each estimator. We selected four of Kunzel's data distributions to replicate (Table I), excluding simulations from the paper which had both unconfounded propensity scores and no treatment effect.

TE Distribution	Our Abbrev.	Paper Abbrev.	Parameters
Unbalanced case with simple CATE	simA	Simulation SI 1	$X_i \sim \mathcal{N}_{20}(0, \sigma)$ where σ is a 20×20 correlation matrix that is created using R's vine method, with 20×20 correlation matrix that is created using R's vine method, with $\alpha = 1$. All X_i are i.i.d. $\pi(x) = 0.01$ $\mu_0(x) = x^T\beta + 5\mathbb{I}(x_1 > 0.5)$, with $\beta \sim Unif\ [-5, 5]^{20}$ $\mu_1(x) = \mu_0(x) + 8\mathbb{I}(x_2 > 0.1)$
Balanced case without confounding and complex linear CATE	simB	Simulation SI 2	$X_i \sim \mathcal{N}_{20}(0, \sigma)$ where σ is a 20×20correlation matrix that is created using R's vine method, with $\alpha = 1$. All X_i are i.i.d. $\pi(x) = 0.5$ $\mu_1(x) = x^T\beta_1$, with $\beta_1 \sim Unif\,[1, 30]^{20}$ $\mu_0(x) = x^T\beta_0$, with $\beta_0 \sim Unif\,[1, 30]^{20}$
Balanced case without confounding and complex non-linear CATE	simC	Simulation SI 3	$X_i \sim \mathcal{N}_{20}(0, \sigma)$ where σ is a 20×20 correlation matrix that is created using R's vine method, with $\alpha = 1$. All X_i are i.i.d. $\pi(x) = 0.5$ $\mu_1(x) = \frac{1}{2}\varsigma(x_1)(x_2)$ $\mu_0(x) = -\frac{1}{2}\varsigma(x_1)(x_2)$ where $\varsigma(x) = \frac{2}{1 + e^{-12(x-\frac{1}{2})}}$
Measured beta-confounding without TE	simD	Simulation SI 6	$x \sim Unif([0, 1]^{n \times 20})$ $\beta_1 = Beta(x_1; 2, 4) = 20x_1(1 - x_1)^3$ $\pi(x) = \frac{1}{4}(1 + \beta_1)$ $\mu_0(x) = 2x_1 - 1$ $\mu_1(x) = \mu_0(x)$

Table I. Description of data generating distributions for synthetic data replication

In their paper Kunzel et al. also introduce the causalToolbox package which contains functions for generating simulated data in R. In early exploration, we observed that some of the parameter options for μ, π, and τ in the `simulate_causal_experiment` function did not align with the distributions listed in the paper. Consequently, we decided to not use the

causalToolbox package directly, and instead implemented our own pipeline for generating synthetic data. Our data generation implementation includes an analogous `simulate_causal_experiment` function. In addition to the simulations listed in Table I, we explore two additional data distributions: 1) measured confounding with TE and 2) unmeasured confounding with TE (Table II).These extensions were motivated by the limited study of confounding in the original paper. Assessing the performance of metalearners in a simulated setting with confounding is a natural next step of this work, as confounding is often a consideration in real-world applications of CATE. Specifically, in simE, we modify the beta-confounding experiment (simD) to include the simple TE from simA. In simF, we consider the case when the ignorability assumption is violated. Ignorability is a strong and untestable assumption, and in practice is often violated when analyzing observational data.

TE Distribution	Our Abbrev.	Description
Measured beta-confounding with simple CATE	simE	$x \sim Unif([0,1]^{n\times 20})$ $\beta_1 = Beta(x_1; 2, 4) = 20x_1(1-x_1)^3$ $\pi(x) = \frac{1}{4}(1+\beta_1)$ $\mu_0(x) = 2x_1 - 1$ $\mu_1(x) = \mu_0(x) + 8\mathbb{I}(x_2 > 0.1)$
Unmeasured confounding with simple CATE	simF	$x \sim Unif([0,1]^{n\times 20})$ $\beta_1 \sim \mathcal{N}(0,1)$ $\pi(x) = max\left(min\left(\frac{2x_1+\beta_1}{2}, 0.95\right), 0.05\right)$ $\mu_0(x) = 2x_1 - 1$ $\mu_1 = \mu_0 + 8\mathbb{I}(x_2 > 0.1) + 2\beta_1$

Table II. Description of data generating distribution for synthetic data extensions

We subsequently performed descriptive analyses of simulated data to verify that CATEs, proportion of instances treated, and covariate relationships in training and test sets were consistent with their definitions.

III. Metalearner Implementation

We implemented the T, S, and X learners in Python3. Recall that for the T-learner, $\hat{\tau^T}(x) = \hat{\mu_1}(x) - \hat{\mu_0}(x)$. Our T-learner implementation uses a base learner to fit $\hat{\mu_1}(x)$ using training data from the instances where $W = 1$, and $\hat{\mu_0}(x)$ from the instances where $W = 0$. For our S-learner implementation, we fit only one model using the base learner, with W and X concatenated to form the input vector. We then predict using $\hat{\mu_1}(x, 1)$ with $W = 1$ filled for all instances and $\hat{\mu_0}(x, 0)$ with $W = 0$ filled for all instances. The S-learner CATE is then given by

$\hat{\tau^S}(x) = \hat{\mu_1}(x,1) - \hat{\mu_0}(x,0)$. Finally, for our X-learner implementation, we first replicate the procedure of the T-learner by fitting $\hat{\mu_1}(x)$ using training data from the instances where $W = 1$, and $\hat{\mu_0}(x)$ from the instances where $W = 0$. Next, we impute $Y(0)$ for the treated group using $\hat{\mu_0}(x)$ and $Y(1)$ for the control group using $\hat{\mu_1}(x)$. We then fit two subsequent models, $\hat{\tau_0}(x)$ and $\hat{\tau_1}(x)$, which are the CATE estimate fit to the control group and the CATE estimate fit to the treatment group, respectively. The X-learner predicted CATE is computed by $\hat{\tau}(x) = g(x)\hat{\tau_0}(x) + (1 - g(x))\hat{\tau_1}(x)$, where $g(x) \in [0,1]$ is a weight function. The authors recommend setting g to be the propensity function, which in the paper they estimate using a random forest classifier (RFC). In our work, we explore the impact of fitting g with logistic regression compared to RFC.

For all experiments (replication and extensions), we implemented a Python script that looped through all data generating simulations and averaged MSE results over 10 samples for each simulation and training set size. The authors used 30 samples in their paper, but we limit this value to 10 due to computational limitations (particularly with the more intensive CausalToolbox hyperparameters and random forest base learner). In initial studies we also found that results averaged over 10 samples were similar to those averaged over 30 samples.

For all experiments but one, we used the same training set sizes documented in the causalToolbox package: [5000, 10,000, 20,000, 100,000, 300,000], and a static test set size of 100,000. The exception was the random forest base learner, with propensity score fit using a RFC and CausalToolbox hyperparameters, which was too computationally expensive to complete with 300,000, so we limited the maximum training set size to 100,000. Each training set size was generated with a stratified sampling method without replacement from each 300,000 sized dataset that was generated by our data simulation script. Experiments were run both locally and on the NYU Greene cluster.

While Kunzel et al. used both random forests (RF) and Bayesian Additive Regression Trees (BART) as base learners in their experiments, we only experimented with RF because we implemented our experiments in Python rather than R, and the implementation of BART that is available in Python (BartPy) had prohibitively slow run-times and was too computationally expensive to be feasible. We also determined that there was not much more to learn from using BART as a base learner that we could not learn from using RF. For our random forest experiments we substituted the honest random forest methods set forth in CausalToolbox with the Python-compatible Regression Forest models from EconML.

Before beginning our experiments, we reached out to the authors to request their MSE results in a tabular format for comparison and to ask about the hyperparameters used to train their metalearners. In response to this request they generously sent us a link to a metalearners-reproduction repository which contained additional code to replicate their original MSE results as well as tabular MSE results for each simulation for training set sizes 5000, 10,000, and 20,000. We compare our replication results to both the original paper and these results from the metalearners-reproduction repository. This repository was recently updated on 5/21/21 to contain additional tabular results; these new results are not used in any of the figures or other contents of this report.

Extension to IW Regression

We extend the methods of the original paper by using linear regression as an alternative base learner. Linear regression has the advantage of being interpretable and very computationally inexpensive, especially if we used logistic regression to estimate the propensities rather than a random forest classifier. In some domains (i.e. regulated finance models), using this type of simple model is actually mandated.

Since the base learners in some of the meta-algorithms are used to estimate distributions that differ from the distributions that the training data are drawn from, we incorporate training weights in an attempt to reduce bias. In most cases we use importance-weighted linear regression (LR-IW). The exception to this rule is that for μ_0 and μ_1 in the X-learner, we instead use inverse-propensity weighting, because we'd like those models be unbiased for the incomplete cases they're used on rather than for the full data. Note that we use "IW" to mean either type of weights.

We chose LR-IW because it has been shown to perform well with missing data, in particular in the Missing At Random (MAR) setting which we experiment with in simD and simE. The specific importance weights used for each base learner are listed in Table III. Note that the displayed weights apply to μ_1 and τ_1; the corresponding weights for μ_0 and τ_0 can be obtained by replacing π with $1-\pi$.

Metalearner	Base learner	Training distribution	Target distribution	Weights**
X	μ_0, μ_1	Complete cases	Incomplete cases	$\frac{(1-\pi)}{\pi}$
X	τ_0, τ_1	Complete cases	Full data	$\frac{1}{\pi}$
T	μ_0, μ_1	Complete cases	Full data	$\frac{1}{\pi}$
S	μ	Full data	Full data	No weights

Table III. Description of importance weights used for IWLR base learners.
** See Importance-weighted regression imputation slides 7 & 10 for derivation.

Bias Analysis Extension

We also extend our analysis to consider not just the MSE of the metalearners, but also the bias of individual predicted components. For the S and T learners we export each row of the test set predictions from $\hat{\mu_0}(x)$ and $\hat{\mu_1}(x)$ and compare them to the true synthetic values, $Y(1)$ and $Y(0)$. In the first stage of the X learner we export the imputed $Y(0)$ for the treated group and $Y(1)$ for the control group and compare them to the true values from the training set. For the X learner we also export the predicted $\hat{\pi}(x)$ and compare it to the true value. For each of the metalearners we compare the predicted $\hat{\tau}(x)$ on the test set to the true $\tau(x)$. We conduct an analysis of the bias from each of these predictions in our export_preds_analysis notebook.

Hyperparameters

In the original paper, the authors used a fixed set of hyperparameters for each metalearner. This set of values was tuned on 20 datasets from the 2016 Atlantic Causal Inference Conference (ACIC) competition to find a set of hyperparameters for each metalearner that would work well in a variety of contexts (Dorie et al. 2019). The authors mention that tuning on their own datasets was too difficult, which is why they used parameters from the ACIC competition. In the paper appendix, they also acknowledge that tuning the hyperparameters for each learner, while difficult, can be beneficial to performance.

It was not immediately apparent from the original paper the exact hyperparameter values that the authors used; however, in the metalearners-reproduction repository that they provided to us, they call the S-, T-, and X-learner functions from the causalToolbox package with the default values. We assume that those are the ones that they used in the paper, which were tuned on datasets from the ACIC16 competition. We reached out to the authors for confirmation but did not hear back from them in time. As such, in our replication experiments we compare to the paper figures and results from the metalearners-reproduction repository using the hyperparameters documented in the causalToolbox package.

We tune the hyperparameters that we used for each base learner, metalearner, and data simulation by adapting the tuning code provided in the causalToolbox package. We first attempted to tune each base learner individually by optimizing the MSE for the given base learner's task (e.g. tuning $\hat{\mu}_0(x)$ in the T-learner on its ability to predict $Y(0)$), but then decided to more closely mirror the authors' implementation by sampling hyperparameters for all base learners and tuning on the MSE of the metalearner. We chose the ranges of hyperparameters for tuning so that they explore the space, making sure that the space also contains the default values used in causalToolbox. We iterate over 200 combinations of hyperparameters, which are sampled from the distributions shown in Table IV.

Hyperparameter	Distribution sampled
n_estimators	In [48, 496]
max_samples	In [0.1, 0.9]
max_features	In [1, 20]
min_samples_leaf	In [1, 30]

Table IV: Distributions used for hyperparameter tuning

Because our work is in Python, the EconML implementation of honest RF that we use does not have all of the same hyperparameter options as causalToolbox provides. In particular, causalToolbox samples with replacement for each tree by default, while EconML always samples without replacement, with no option to sample with replacement. causalToolbox also provides multiple options for customizing the subsets for determining the tree splits and leaf values, and EconML has no such customization options. We chose to tune the parameters that we could and that are tuned in the causalToolbox implementation, and leave the rest at the default values. We additionally tuned the number of trees in each forest, which causalToolbox

does not. A complete summary of the hyperparameters in causalToolbox and in our experiments is shown in Table VI.

IV. CATE Confidence Interval Implementation

Kunzel et al. constructed 95% confidence intervals around their Get Out The Vote ("GOTV") CATE estimates using a method which performed well on an X-learner in the Atlantic Causal Inference Conference (ACIC) challenge. (We refer to this method as "ACIC.") They were unable to achieve 95% coverage on the GOTV dataset, confirming a finding from ACIC that creating confidence intervals which achieve nominal coverage in a wide range of CATE settings is extremely difficult. Kunzel et al. also tested a second confidence interval algorithm (referred to as "smoothed") and found that it yielded similar results.

We test the coverage of the ACIC CI method in the confounded setting, on simE and simF. We use an alpha level of 0.05, to construct 95% confidence intervals. For each simulation, we test coverage on each metalearner's predictions (T, S, and X), alternately using as base learners RF with default parameters and LR with importance weights. These base learners were selected in order to study results for set-ups with high MSE (LR-IW) and with low MSE (RF). In each case the bootstrap samples are drawn from a training set of 20,000 observations, and coverage is measured on a test set of 100,000 observations. We considered testing whether increasing the size of the training set would improve coverage, but determined that this line of inquiry was not worth the increased computational resources, as we found that the MSE results were fairly steady for training sizes larger than 20,000. We note that Kunzel et al. used training sets of 50,000 and test sets of 2000 for their GOTV confidence intervals. We hope to achieve more precise coverage estimates by using the full test set of 100,000 observations.

In addition to the ACIC method of estimating confidence intervals, we also test the percentile method (Puth et al. 2015), the basic (or "reverse percentile") method (Hesterberg 2014), first-order normal approximation (Puth et al. 2015), and a studentized (or "bootstrap-t") method (Tibshirani 2014). The studentized method requires a second-level of bootstrap sampling to estimate standard errors, and is therefore far more computationally demanding than the other methods. Initial tests on the T LR-IW learner revealed that the ACIC intervals significantly outperformed studentized intervals, so we did not extend the study of the studentized method to other learning configurations.

Kunzel et al. used B=10,000 bootstrap samples to estimate their confidence intervals. We attempted to use B=10,000 wherever possible, but this requires significant computational resources. For particularly computation-heavy learning set-ups (i.e., X-RF), we were not able to work with more than 1000 bootstrap samples. To investigate the relationship between B (number of bootstrap samples) and confidence interval quality, we subsetted our bootstrap prediction results to recalculate intervals for B=100, 1000, 5000, and 10,000.

V. Results

The results of our replication analysis for simA-simD can be found in the Appendix (Figure 4). We compare the original paper RF results (leftmost plots) to those from the metalearners-reproduction repository (middle plots) and to those we replicate with EconML default hyperparameters (rightmost plots).

Simulation	**Best Learner Configuration at N=100,000**	**Notes**	**Mean MSE**
simA	X-RF Hyperparameters: Causal ToolBox g: RFC	Based on analysis with default RF hyperparameters, the choice of base learner for g is insignificant . We therefore suspect the simpler g model (logistic regression) would also perform well in the selected set-up, and would be more desirable due to simplicity. This specific set-up was not tested, however.	2.1711
simB	S-RF Hyperparameters: Tuned	No g used because LR was implemented without weights.	2367.77
simC	X-RF Hyperparameters: Causal ToolBox g: RFC	T and S results are both within 0.0002. Based on analysis with default RF hyperparameters, the choice of base learner for g is insignificant . We therefore suspect the simpler g model (logistic regression) would also perform well in the selected set-up, and would be more desirable due to simplicity. This specific set-up was not tested, however.	3.751E-3
simD	S-LR g: Logistic Regression	Performance is identical regardless of choice of the base learner for g. We therefore select logistic regression because it is the simpler model.	1.542E-5
simE	X-RF Hyperparameters: Causal ToolBox g: RFC	Based on analysis with default RF hyperparameters, the choice of base learner for g is insignificant . We therefore suspect the simpler g model (logistic regression) would also perform well in the selected set-up, and would be more desirable due to simplicity. This specific set-up was not tested, however.	1.284E-3
simF	X-RF Hyperparameters: Default g: Logistic Regression	X-learner outperforms the other metalearners, but the choice of base learner is less clear. X with author-tuned RF and RFC g, default-tuned RF and RFC g, and LR-IW with logistic regression all perform within 0.03 of the selected configuration. We expect that X with author-tuned RF and logistic regression g would in fact be the best configuration, as the author-tuned hyperparameters slightly out performs the default parameters when combined with RFC g in this setting. This specific set-up was not tested, however.	5.7688

Table V. Optimal learner configurations for each simulation's experiments and additional observations

Tuning

The results of our initial tuning run can be found in the "tuned" hyperparameter files (e.g. rf_x_tuned.json). Ultimately, our tuned hyperparameters fail to improve upon the default CausalToolbox hyperparameters for most simulations (Figure 5), with the exception of the T and S learner in simB. As such, we decided to start another tuning run in order to more thoroughly explore a refined space with a higher number of estimators, max samples, max features, and lower minimum samples per leaf. Unfortunately, due to computational constraints, this second tuning run was not able to complete before the project deadline.

Confidence Intervals

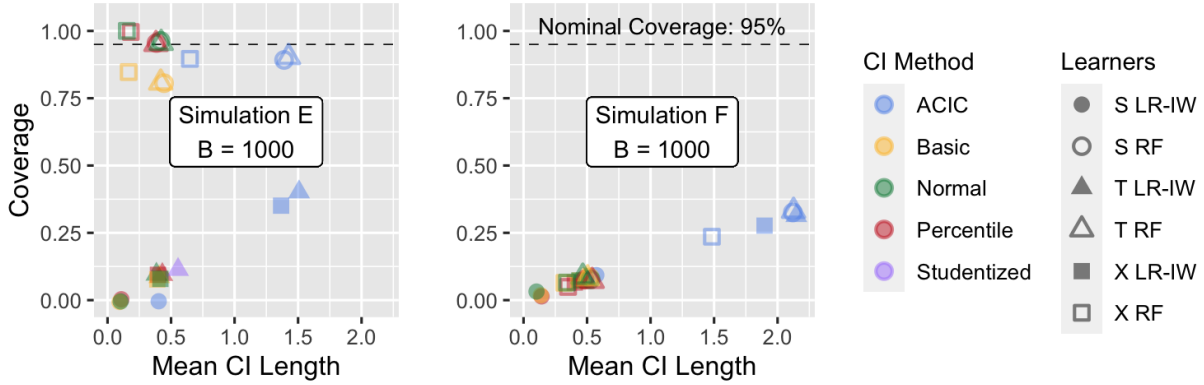


Figure 1: Confidence intervals for Simulations E and F. Coverage is measured by the proportion of 100,000 test set observations for which the true lies in the CI. Mean interval length is calculated across all 100,000. Jitter is added to better reveal overlapped points (jitter radius is set to 1% of the total displayed Y-axis and 2% of the total displayed X-axis.)

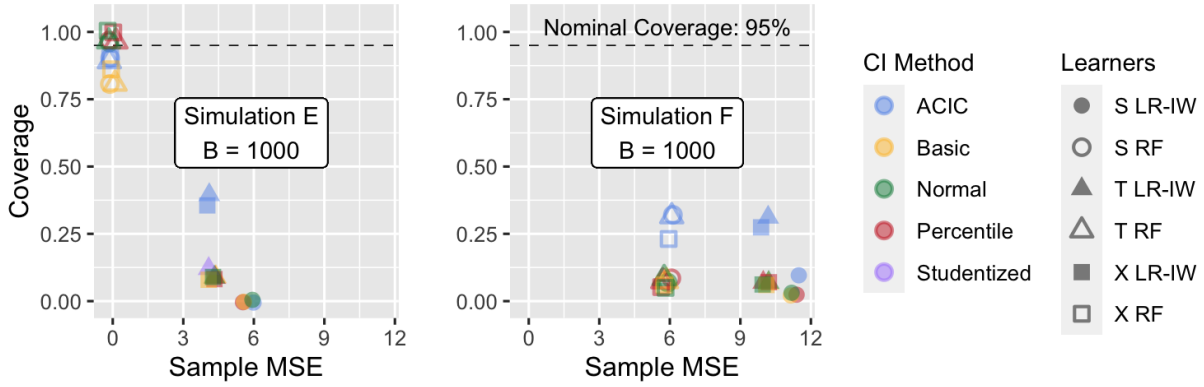


Figure 2: Confidence intervals for Simulations E and F. Coverage is measured by the proportion of 100,000 test set observations for which the true lies in the CI. The X-axis measures the mean square error of the given learning set-up on the sample from which confidence intervals are calculated. Jitter is added to better reveal overlapped points (jitter radius is set to 1% of the total displayed Y-axis and 2% of the total displayed X-axis.)

With the exception of S LR-IW, ACIC consistently outperforms all other methods when MSE is high. This gain in coverage is accompanied by wider intervals. For S LR-IW, the ACIC interval is still slightly larger than the intervals estimated by other methods, yet it does not gain any coverage in simE and gains only a slight amount of coverage in simF.

For simF, the RF-based learners all yield high MSEs. In this setting, ACIC prevails over all other methods; ACIC's gains in these settings are comparable to those it achieves in analogous LR-IW settings.

Normal and percentile methods get better coverage than ACIC for all RF-based metalearners on simE, and the intervals are lower. These are the lowest-MSE settings which we built confidence intervals for, and are the only cases in which we achieve the nominal coverage of 95%.

In all set-ups, we observe only very minor improvements when B is increased above 1000 (See Figure 3 in Appendix.)

VI. Discussion

Tuning

The fact that we were largely unable to improve performance after hyperparameter tuning proves Kunzel et al.'s point that, "tuning the base learners to receive better CATE estimators… is very difficult". Our worsened results after tuning demonstrate that hyperparameter tuning can actually be harmful if the hyperparameter space is not adequately searched. Future efforts on this work should concentrate on further tuning the hyperparameters, using the distributions specified in our most recent update to our tuning script.

Interestingly, we also found that across all simulations the default EconML hyperparameters perform similarly to the default hyperparameters in CausalToolbox. This suggests that the tuning exercise done in Kunzel et al. to create the CausalToolbox hyperparameters was not very successful. The authors acknowledge this limitation in the paper, admitting that tuning each algorithm for each data set separately is very challenging.

Confounding

For the data generating distributions that had confounding (simD, simE, and simF), we conclude that the X-learner with RF base learner is more capable of handling confounding than LR-IW. In particular, in simE, the RF outperforms LR-IW by one order of magnitude. At the same time, however, the best-performing metalearner depends on the simulation. In simE, the X-learner performs better than the S- and T- learners, but in simD, the S- learner performs better than the X or T learner. This highlights the fact that our experiments tested multiple factors against each other simultaneously (propensity, CATE function, and confounding). This is one gap in our simulation methodology that makes it difficult to isolate which characteristics of the data-generating distribution are most attributable to good or bad model performance. It would be advantageous for future work to more exhaustively consider data-distributions for combinations of these factors of interest.

Also of note in these simulations with confounding was the observation that simF was the only experiment yielding different results when the propensity function for estimating IW weights was fit with RFC compared to logistic regression. RFC performs worse, which we suspect is because the propensity model is overfitting due to the ignorability violation.

IW Regression

Our base learner analysis revealed that the method for estimating propensity weights did not generally matter, except in the case of unmeasured confounding. (See discussion in the "Confounding" section above.)

When used with S-learners, performance of LR base learners was on par with RF. Although S-learners are not often the best performing method (with the exception of simD, where there was no TE), it is useful to learn that simpler base learners may be a more appropriate choice for this metalearner.

It is of note that S was the only metalearner for which we performed un-weighted regression; the relative success of regression in this context may indicate that our selected weights do not achieve the intended bias reduction. The weights used for τ_0 and τ_1 in the X-learner, for example, correct only for the covariate distribution on which they are trained, but do not account for the fact that these models are learned from the joint distribution of these covariates with the imputed treatment effects of the opposite treatment groups. A rigorous proof of the appropriate weights is a key area for future work in this vein. Assessing the success of the weights empirically would also require implementing unweighted regression base learners to use as a baseline for comparison.

In all of our experiments, the base learners used within a single model are either all RF or all LR. A key benefit of the X-learner is that it allows for the use of different models for different purposes and of different complexities. For example, as noted by Kunzel, it could be useful in the unbalanced case (simA) to use a simpler model to model to estimate Y(1), where we have less data available. Future work could harness our bias analysis results to identify other places where simpler models, or specific parametric models, would be useful.

Confidence Intervals

Kunzel's experiments with confidence intervals support the ACIC finding that it is difficult to construct confidence intervals that yield any formal guarantees on generalized metalearners without any parametric assumptions on the CATE. Our experiments further bolster this assertion. This constitutes a considerable limitation to the applicability of the X-learner method: it is not appropriate for causal inference, which is often a key goal in CATE estimation. Rather, the X-learner is best suited to uncover latent heterogeneity in the treatment effect which can then be explored with appropriate methods from causal inference.

Our experiments indicate that the performance gains achieved by increasing the number of bootstrap samples (B) above 1000 are quite minimal. We expect that higher B will yield lower-variance intervals. Although we did not see any confidence intervals which were obvious outliers, our experimental design did not allow for explicit variance testing in regards to B, as we only constructed one interval for each B value in a given learning set-up.

In general, we find that Kunzel's choice of ACIC far outperforms all other methods of calculating confidence intervals. These gains are achieved by using wider intervals, rather than improved centering.

Preliminary analysis of bootstrap-t confidence intervals indicated that the increased computation required for this method was not justified, however there is room for future work to

explore the use of a studentized method which relies on a heuristic for estimating standard error, rather than the computationally-expensive second-level bootstrapping.

We find that the percentile and normal methods are the best performers in low-MSE settings, achieving the nominal 95% coverage where ACIC does not, and with far narrower intervals. Our experiments on simE and simF are not enough to make a formal conclusion in this regard; we simply caution against any work which requires formal guarantees from confidence intervals. Further work ought to be devoted to studying whether relaxed confidence guarantees could be derived from normal or percentile methods in low-MSE settings.

Bias Analysis

Our experiments tested a great many factors against each other, and yielded a huge amount of data to be analyzed. There is much more work to be done in analyzing sources of bias for our learning set-ups in various simulations, and our exported CATE predictions offer a rich source of insight which has not yet been fully mined. Some preliminary directions for future analysis based on our initial observations are:

1) In the X-learner setting, there is not a clear relationship between the true and predicted Y values, suggesting that μ_0 and μ_1 are performing poorly. However, this does not seem to be affecting the total error significantly. To what extent do poorly fit intermediate Y predictions ($\hat{Y_0}$, $\hat{Y_1}$) influence the performance of the final CATE estimator?
2) In simA it appears that the S-learner always predicts a single value of tau. Additionally, performance does not improve with additional training data (SI Figure 2-4). Is this an artifact of the unbalanced treatment and control groups?
3) In simE and simF we see very similar shapes across all RF S and T plots. Does this suggest that using separate μ_0 and $\mu_1$learners in the T-learner setting does not improve upon the single learner used in the S-learner setting?

Replication

Overall, we find our replicated results to be generally consistent with the results of Kunzel et al., and that our MSE values are always within a reasonable range of the values published in the paper figures. Considering these similarities it is likely that the default causalToolbox hyperparameter values are what they used to create the figures in the paper.

It is difficult to compare the original paper results to the results from the metalearners-reproduction repository because of the difference in sample sizes. However in the range that overlaps ($N \leq 20000$), we find the two to be reasonably similar across simulations with some exceptions (e.g. SimA Figure 4).

Appendix

causalToolbox Parameter	Definition in casualToolbox	EconML Parameter
ntree	The number of trees to grow in the forest. The default value is 500.	n_estimators
replace	An indicator of whether sampling of training data is with replacement. The default value is TRUE.	No analogue - sampling is always without replacement
sample.fraction	If this is given, then sampsize is ignored and set to be round(length(y) * sample.fraction). It must be a real number between 0 and 1	max_samples
mtry	The number of variables randomly selected at each split point. The default value is set to be one third of total number of features of the training data.	max_features
nodesizeSpl	Minimum observations contained in terminal nodes. The default value is 3.	min_samples_leaf
nodesizeAvg	Minimum size of terminal nodes for averaging dataset. The default value is 3.	No analogue - presumably same as min_samples_leaf
splitRatio	Proportion of the training data used as the splitting dataset. It is a ratio between 0 and 1. If the ratio is 1, then essentially splitting dataset becomes the total entire sampled set and the averaging dataset is empty. If the ratio is 0, then the splitting data set is empty and all the data is used for the averaging data set (This is not a good usage however since there will be no data available for splitting).	No analogue - value is always 0.5 if honest = True, honest = False corresponds to value of 1
middleSplit	Flag to indicate whether the split value takes the average of two feature values. If false, it will take a point based on a uniform distribution between two feature values. The default value is FALSE.	No analogue

Table VI. Description of hyperparameters that are set in the S-, T-, and X-learner functions in causalToolbox[1], and their analogues in EconML.

[1] These are the parameters that were set at the time that we were determining our tuning procedure, but they have since been slightly altered in the causalToolbox repository.

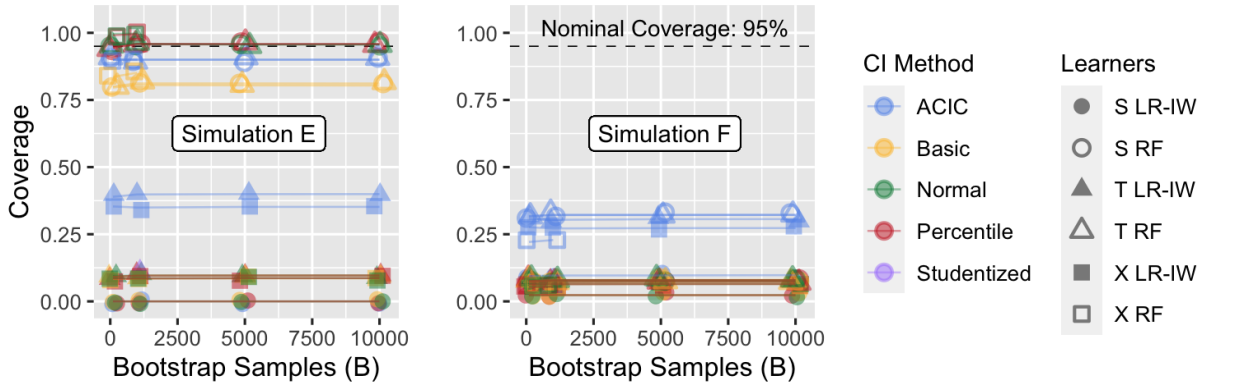
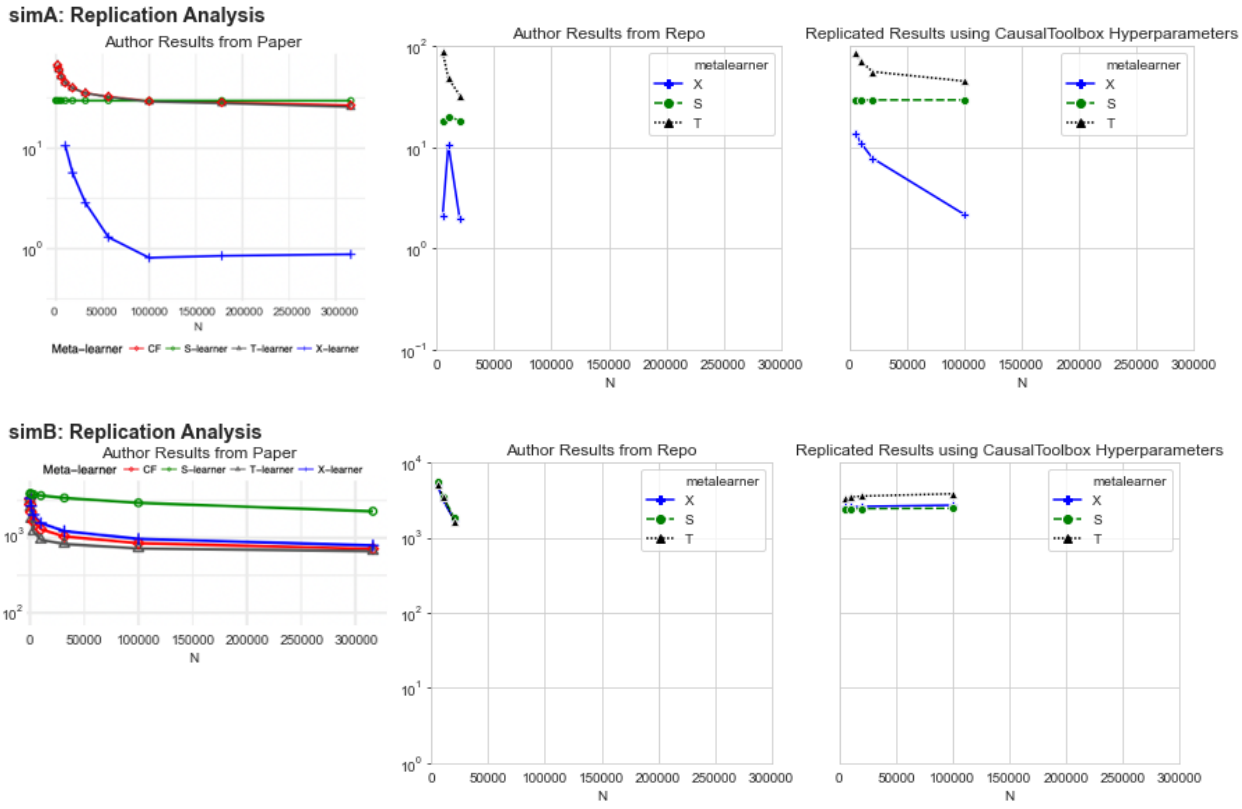


Figure 3: Confidence intervals for Simulations E and F, calculated using varying numbers of bootstrapped predictions. Coverage is measured by the proportion of 100,000 test set observations for which the true lies in the CI. Jitter is added to better reveal overlapped points (jitter radius is set to 1% of the total displayed Y-axis and 2% of the total displayed X-axis.)

Replication Analysis



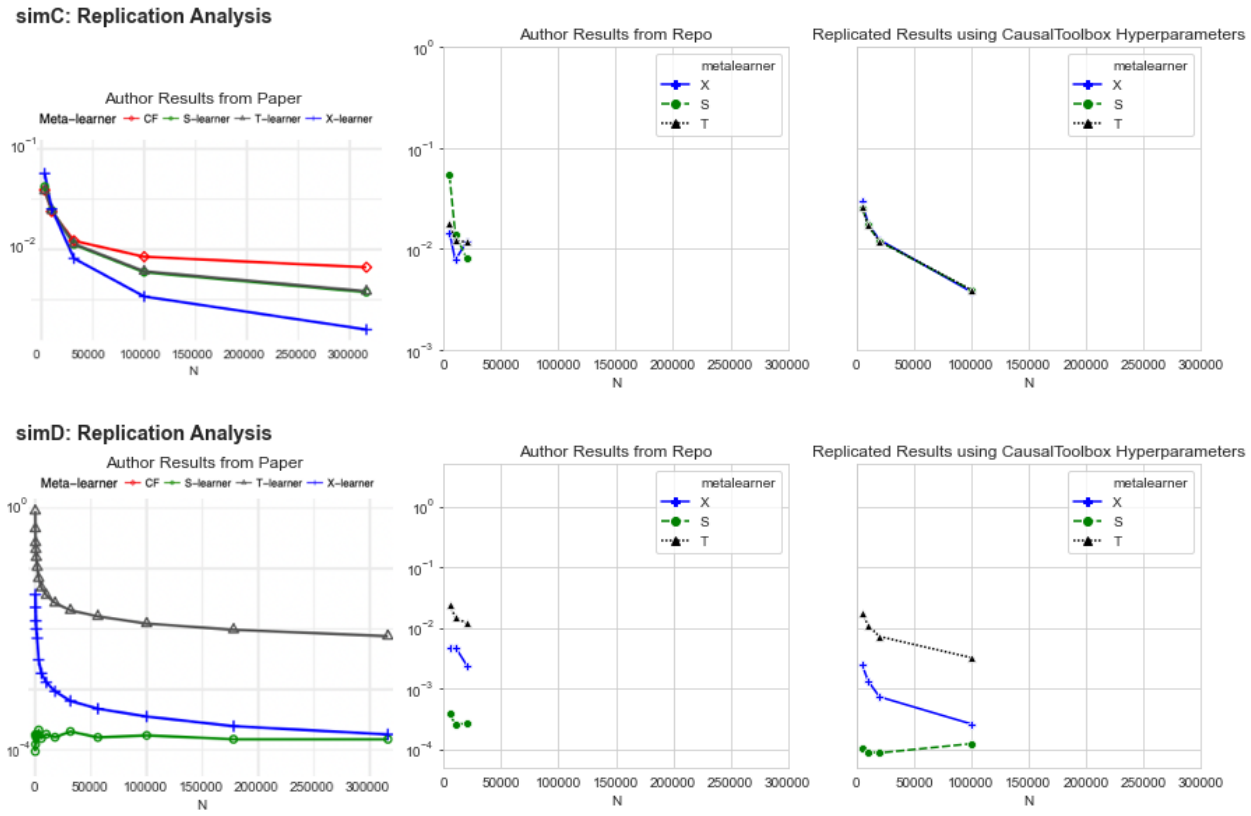
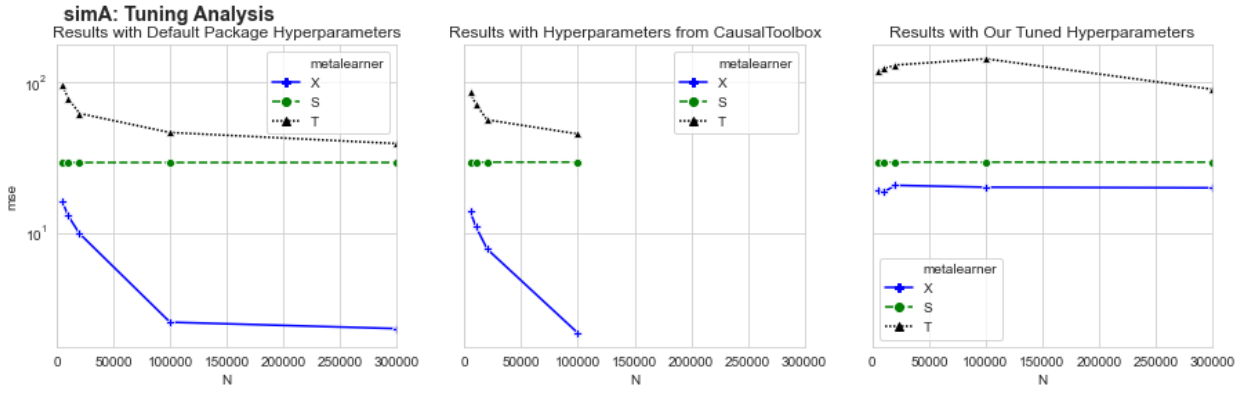
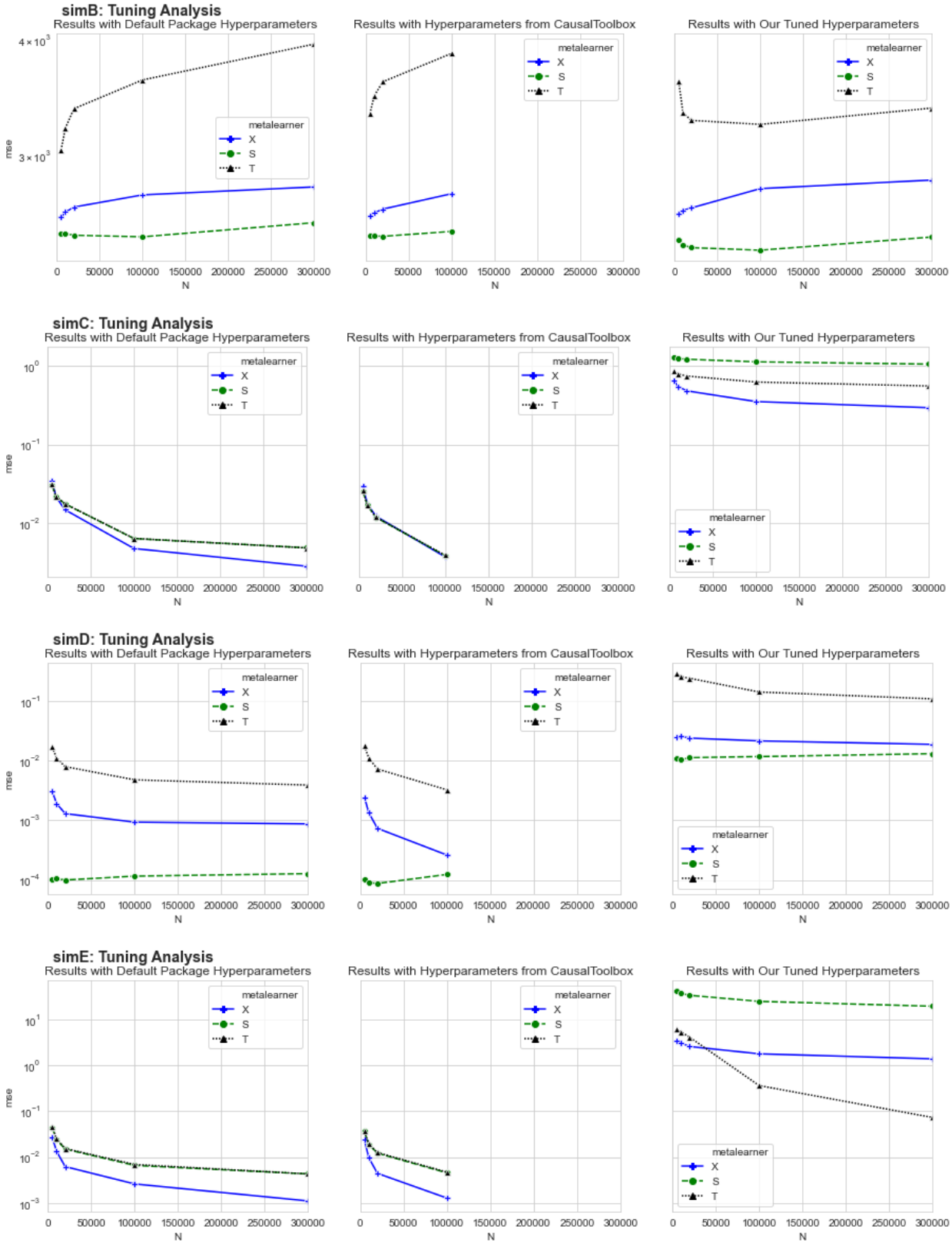


Figure 4. MSE results across simulations from the original paper (left), metalearners-reproduction repository (middle), and our replication with EconML RF and CausalToolbox hyperparameters (right).

Tuning Analysis




simB: Tuning Analysis
Results with Default Package Hyperparameters
Results with Hyperparameters from CausalToolbox
Results with Our Tuned Hyperparameters
simC: Tuning Analysis
Results with Default Package Hyperparameters
Results with Hyperparameters from CausalToolbox
Results with Our Tuned Hyperparameters
simD: Tuning Analysis
Results with Default Package Hyperparameters
Results with Hyperparameters from CausalToolbox
Results with Our Tuned Hyperparameters
simE: Tuning Analysis
Results with Default Package Hyperparameters
Results with Hyperparameters from CausalToolbox
Results with Our Tuned Hyperparameters
mse
N
metalearner
X
S
T

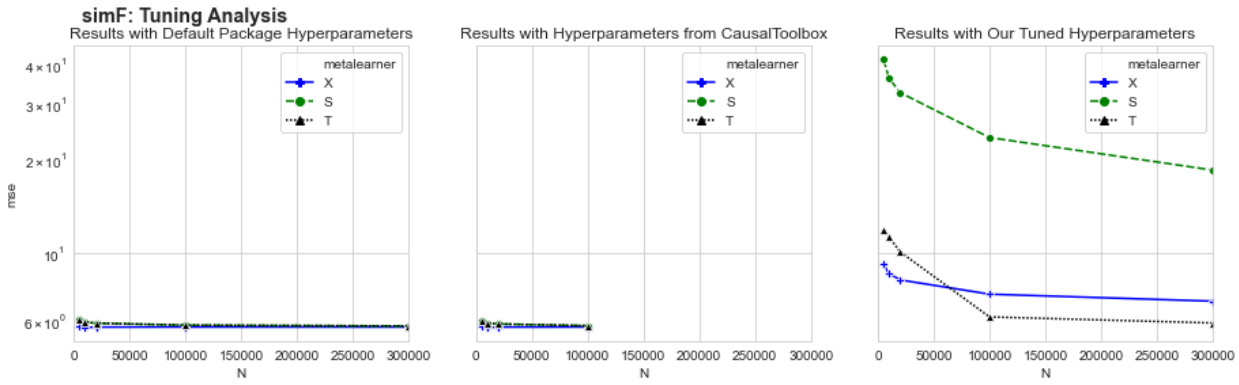
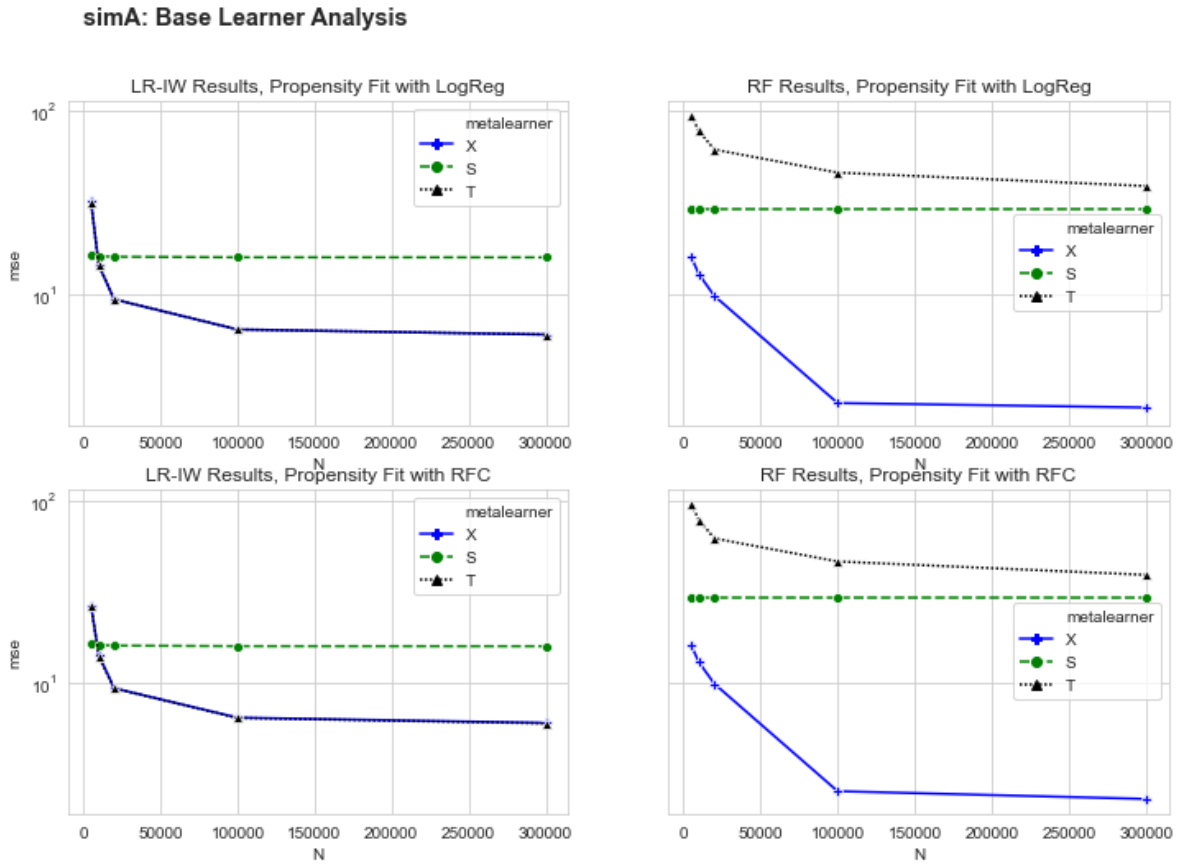
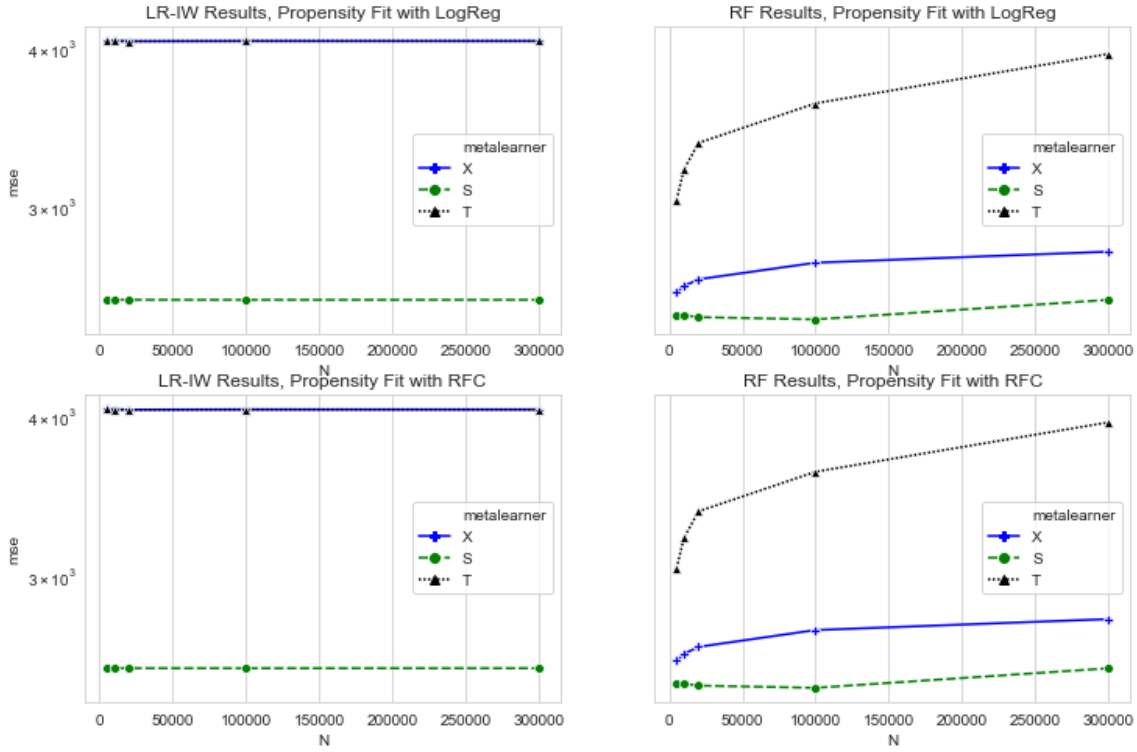


Figure 5. MSE results across simulations using default EconML hyperparameters (left), CausalToolbox default hyperparameters (middle), and with best-performing hyperparameters from our tuning experiment (right). All models trained with EconML RegressionForest and propensities fit with random forest classifier.

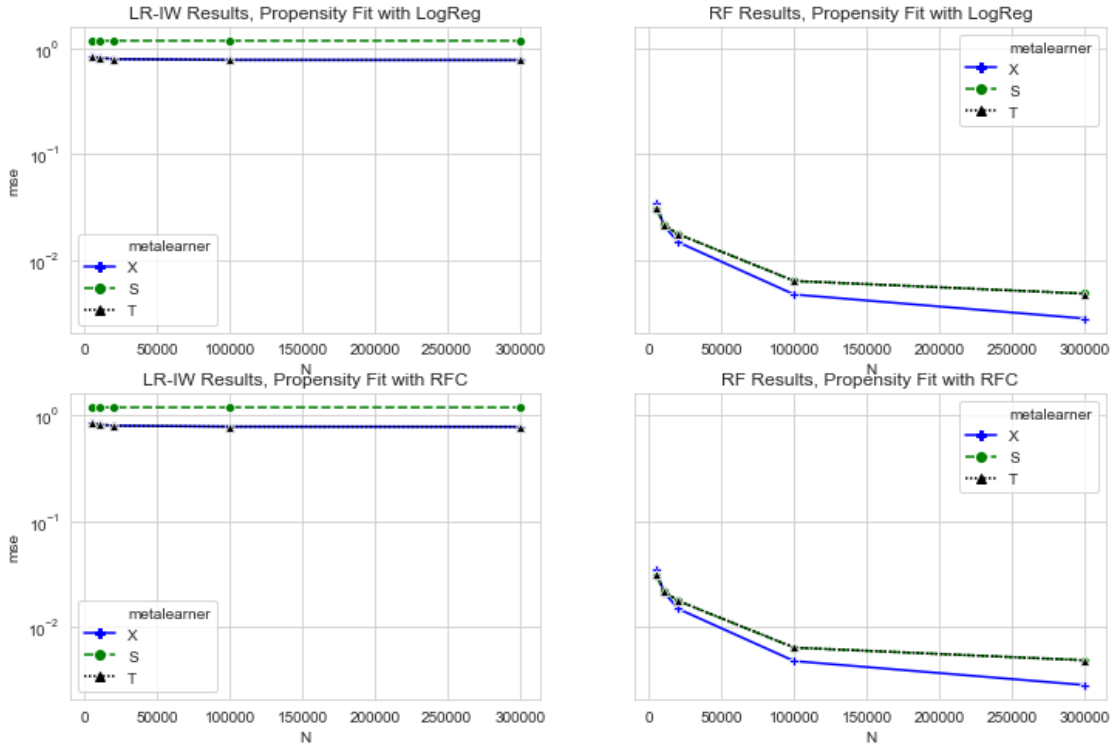
Base Learner Analysis



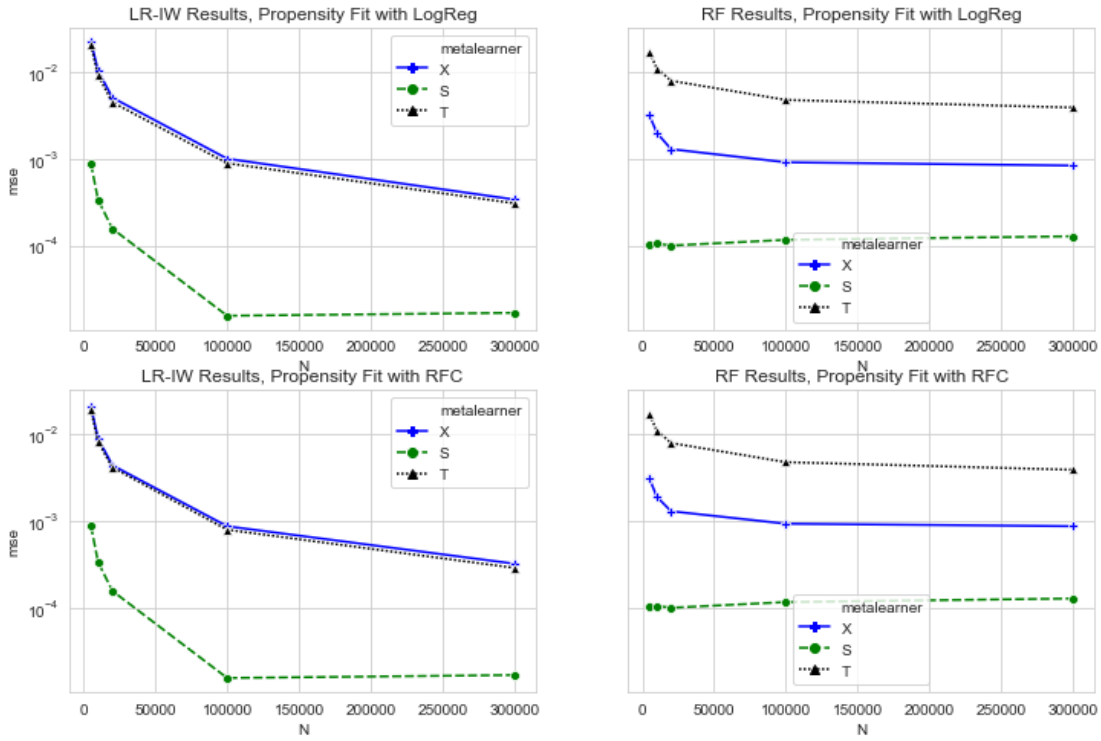
simB: Base Learner Analysis



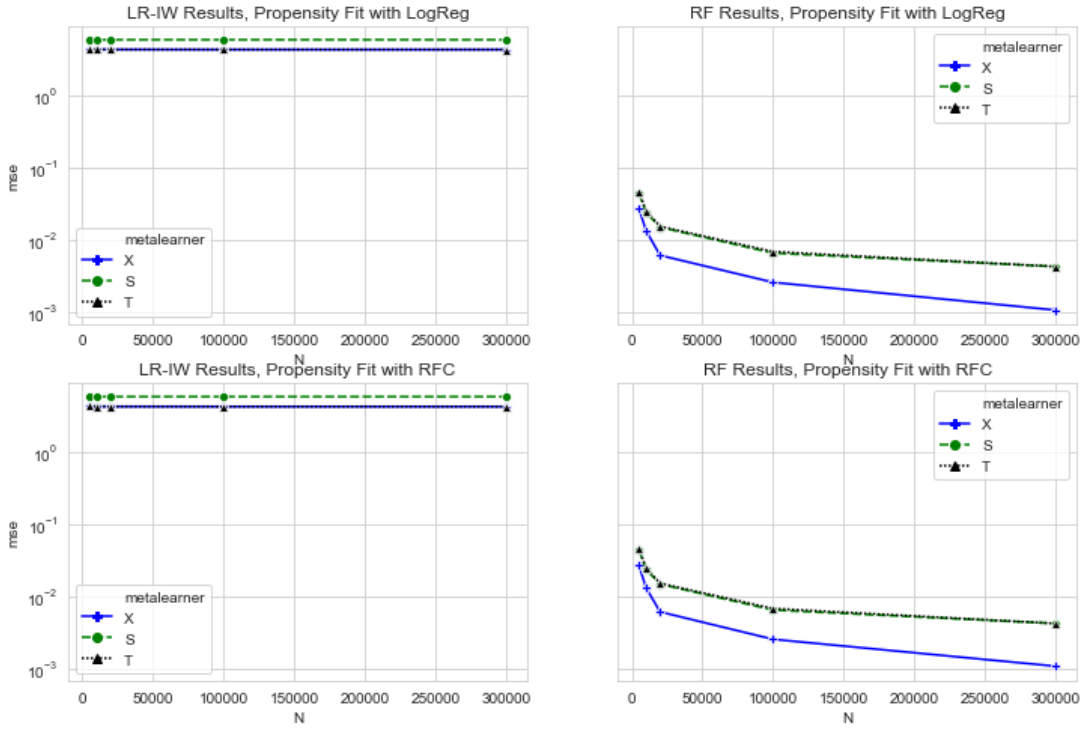
simC: Base Learner Analysis



simD: Base Learner Analysis



simE: Base Learner Analysis



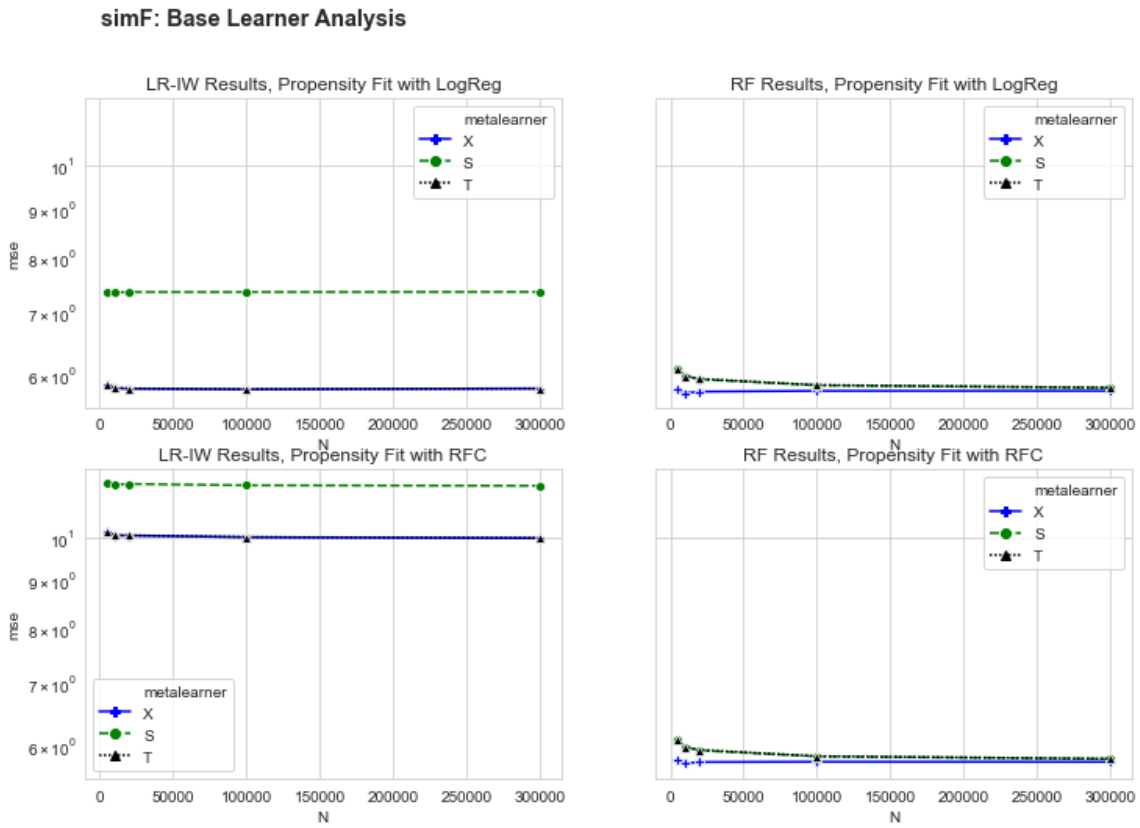


Figure 6. MSE results across simulations using LR-IW base learner with logistic regression propensity model (top left), LR-IW with random forest classifier propensity model (bottom left), RF base learner with logistic regression propensity model (top right), and RF with random forest classifier propensity model (bottom right). All models use the default hyperparameters provided by the package (scikit-learn or EconML).

Works Cited

Dorie, V., Hill, J., Shalit, U., Scott, M., & Cervone, D. (2019). Automated versus do-it-yourself methods for causal inference: Lessons learned from a data analysis competition. *Statistical Science*, *34*(1), 43-68.

Hesterberg, T. C. (2015). What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum. The American Statistician, 69(4), 371-386.

Künzel, S. R., Sekhon, J. S., Bickel, P. J., & Yu, B. (2019). Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the national academy of sciences*, *116*(10), 4156-4165.

Puth, M. T., Neuhäuser, M., & Ruxton, G. D. (2015). On the variety of methods for calculating confidence intervals by bootstrapping. Journal of Animal Ecology, 84(4), 892-897.

Tibshirani, R. (2014). *The Bootstrap*. Retrieved from Carnegie Mellon University Statistics 36-402/36-608 Advanced Methods for Data Analysis course site.