Introduction to Causal Machine Learning

Turing Masterclass

Stijn Vansteelandt

Ghent University, Belgium

London School of Hygiene and Tropical Medicine, U.K.

Machine learning-based plug-in estimators

- We have seen that treatment effects can be evaluated by cleverly averaging predictions
  - of outcome, as if all were (un)treated,
  - of treatment (a.k.a. propensity scores).
- It is tempting to obtain such predictions via application of standard machine learning (ML) algorithms.
- But we have seen that this can be problematic...

Outline

1. Why are naïve ML-based plug-in estimators problematic?
2. How to remove plug-in bias?
3. What are the general principles?

One obvious concern (1)

concern 1: no valid uncertainty margins

plug-in estimators based on machine learning
are 'easily' obtained,
but we have no clue how accurate these are...

- Suppose we use machine learning to estimate

$$Q_0(W) \equiv E(Y|A=1, W)$$

as $\bar{Q}_n(W)$.

- Then we can estimate $E(Y^1)$ as

$$\frac{1}{n}\sum_{i=1}^{n}\bar{Q}_n(W_i)$$

One obvious concern (2)

It may be tempting

- to evaluate the standard error as

$$\frac{1}{\sqrt{n}}\mathrm{SD}\left\{\bar{Q}_n(W_i)\right\}$$

but this ignores uncertainty in the predictions.

(estimator of $E(Y^1)$ is also usually non-normally distributed).

- to consider the bootstrap,
but this has no theoretical justification.

(e.g. Samworth, 2011)

- to recur to model-based statistical analyses
but these only provide valid uncertainty margins
when the model is pre-specified,
which is rarely the case in practice.

Why does the bootstrap fail?

- Consider an analysis whereby we first fit model

$$E(Y|A, W) = \psi A + \theta W$$

  using OLS.
- We report the resulting OLS estimator $\hat{\psi}$ of ψ if there is evidence that $\theta \neq 0$ (based on a standard test), and otherwise report the OLS estimator $\hat{\psi}_0$ in model

$$E(Y|A) = \psi_0 A.$$

What should we expect to see?

- Suppose that in truth

$$E(Y|A,W) = 0 \quad \text{and} \quad E(A|W) = 3W.$$

- Since $\theta = 0$, the chance of falsely concluding that $\theta \neq 0$ is 5%.
- We expect to see a mixture of estimates $\hat{\psi}$ (5%) and $\hat{\psi}_0$ (95%).
- Both are normal, centred at zero, but have different variance.

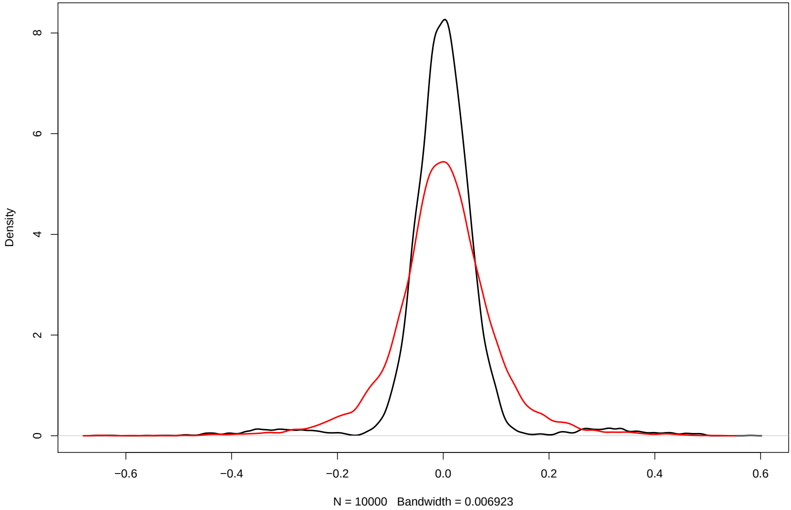
What does bootstrap inference suggest?

- A parametric bootstrap evaluates the procedure on simulated data, using

$$E(Y|A,W) = \hat{\psi}A + \hat{\theta}W$$

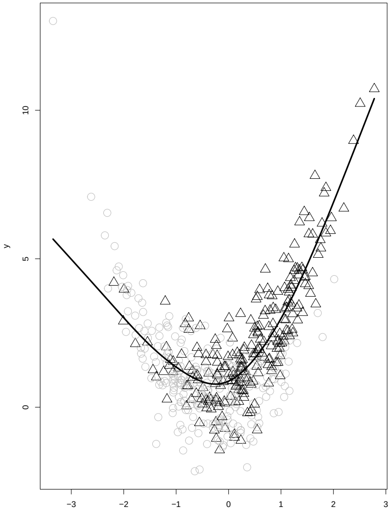
- Because we now generate data with $\hat{\theta} \neq 0$, the chance of falsely concluding $\theta \neq 0$ on the simulated data is larger than 5%.
- In a simulation, we found it to be 16.5%.
- This problem does not disappear in large sample sizes.
- Indeed, $\hat{\theta}$ has a similar order of magnitude as the SE, so that tests remain 'hesitant' as to accept/reject the null hypothesis that $\theta = 0$.

Empirical (black) versus bootstrap (red) distribution of $\hat{\psi}$



One subtle concern (1)

- There is also a more subtle concern that makes naïve ML-based plug-in estimators problematic.
- Machine learning is 'at best' optimally tuned to minimise prediction error.
- But not to deliver low bias in treatment effect estimates.



One subtle concern (2)

concern 2: plug-in bias

plugging machine learning predictions into a statistical analysis, typically induces plug-in bias.

- e.g. oversmoothing is useful
  for prediction over the observed data range,
  but may induce severe bias in causal effect estimates;
- e.g. eliminating strong predictors of treatment is useful
  for prediction over the observed data range,
  but may lead to mistakenly throwing out important confounders.
- Undersmoothing can reduce plug-in bias.
- However, it is not readily clear how this should be done...

Summary so far

- Machine learning offers useful perspectives for better confounding control and more objective analysis.
- But honest uncertainty assessments are lacking because it delivers estimators with
  - relatively large approximation bias, partly as a result of ‘poor tuning’.
  - difficult-to-calculate standard errors,
  - and complex mixture distributions.
- There is no simple remedy...

Outline

1. Why are naïve ML-based plug-in estimators problematic?
2. How to remove plug-in bias?
3. What are the general principles?

How to remove plug-in bias?

- This problem of plug-in bias has no simple remedy.
- When the aim is prediction,
  we can always compare predictions with observed outcomes.
- Even when we use wrong models
  or poorly understood algorithms,
  we can therefore assess some measure of prediction error,
  e.g., based on cross-validation.
- We can even optimise the algorithms
  by using mean squared (prediction) error as a loss function.

How to remove plug-in bias?

- When the aim is effect estimation, we cannot compare predictions with observed outcomes.
- E.g. we cannot measure the bias in $E(Y^1)$ since Y^1 is only measured for a selective subgroup.
- Asymptotic theory from mathematical statistics therefore unavoidably plays a key role.

A bit of history...

- Foundations for a solution have been laid in the 80's - 90's.
- Mathematical statisticians were then studying plug-in estimators, obtained by plugging non-parametric estimators into statistical functionals.

(e.g. Bickel et al., 1982, 1998; Newey, 1990; Pfanzagl, 1982; Robins and Rotnitzky, 1995; van der Vaart, 1991;...)

- van der Laan made use of this theory to construct plug-in estimators based on machine learning, which he called Targeted Maximum Likelihood Estimators or Targeted Minimum Loss-based Estimators.

(van der Laan and Rubin, 2008; van der Laan and Rose, 2014)

- He refers to his approach as targeted learning.
- Related proposals were recently made by Belloni, Chernozhukov, Newey, Robins, ...

(Chernozhukov et al., 2018)

- They refer to their approach as double machine learning.

Causal machine learning for the ATE

- I will talk more broadly about causal machine learning:
  the use of machine learning for the evaluation of causal effects (not individual prediction).
- I will suggest one specific causal ML proposal,
  drawing on your intuition for now.
- We will sketch the more general principles in the next sessions.
- We will then also provide more formal insight why it works.

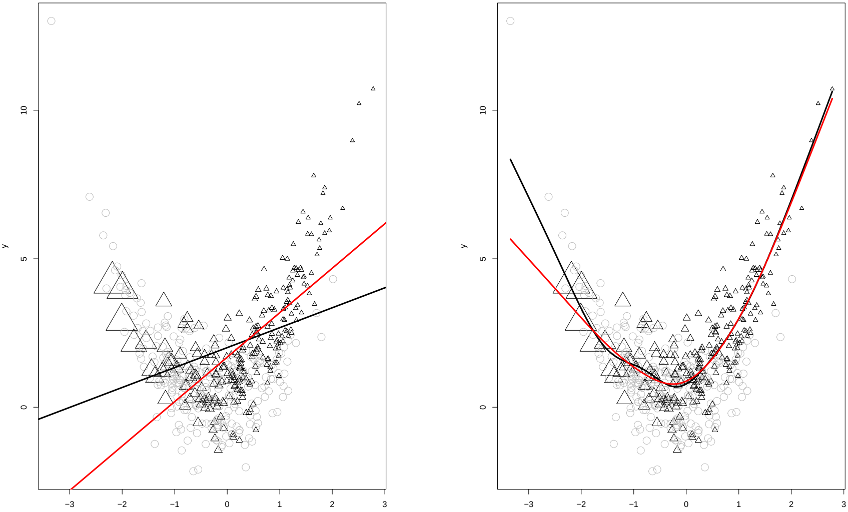
Causal machine learning for $E(Y^1)$

Suppose that we

- evaluate $P(A = 1|W)$ as $g_n(W)$;
- evaluate $E(Y|A = 1, W)$ in the treated as $\bar{Q}_n^{(0)}(W)$, weighting by $1/g_n(W)$;
- use this to predict outcome for all;
- average these predictions.

Then surely we have tuned the predictions better towards the eventual goal.

Causal machine learning for $E(Y^1)$



Causal machine learning for $E(Y^1)$

- It can be shown that the foregoing procedure 'almost' succeeds to remove plug-in bias.
- To make it work, one must additionally calibrate predictions as

$$\bar{Q}_n^{(1)}(W) = \bar{Q}_n^{(0)}(W) + \delta$$

with

$$\delta = \frac{\sum_{i=1}^{n} \{A_i / g_n(W_i)\} \left\{ Y_i - Q_n^{(0)}(W_i) \right\}}{\sum_{i=1}^{n} \{A_i / g_n(W_i)\}}$$

An impression...

- Consider estimating the ATE $\psi = 0.5$ indexing

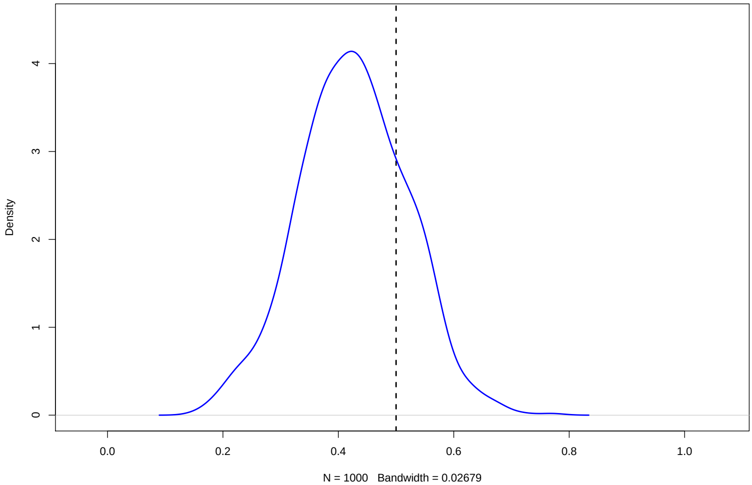
$$E(Y|A, W) = \psi A + 1.25 \cos^2(\gamma' W)$$

with standard normal noise and

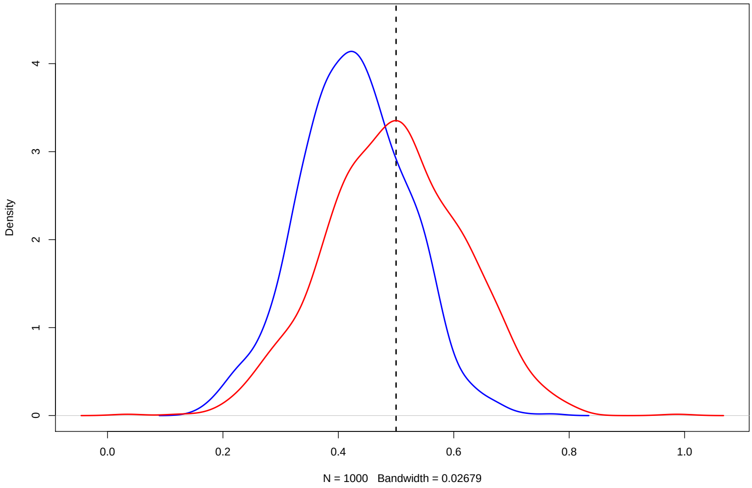
$$P(A = 1|W) = \mathrm{expit}\left\{2 \cos(\gamma' W) + 2 \sin(\gamma' W)\right\}.$$

- $p = 10, n = 500.$

Naïve machine learning using random forests



Causal machine learning using random forests



Summary so far

- Plug-in estimators typically suffer plug-in bias
as a result of ML algorithms aiming for minimal prediction error,
instead of being tuned towards the estimand of interest.
- They also typically have a complex distribution
with difficult-to-calculate variance.
- Causal machine learning aims to remove plug-in bias.
- We will see that it delivers
estimators with easy-to-calculate variance,
even when the uncertainty in the ML predictions
is unknown.

Outline

1. Why are naïve ML-based plug-in estimators problematic?
2. How to remove plug-in bias?
3. What are the general principles?

How accurate is the plug-in estimator?

- Let $\theta(P)$ be the parameter of interest, evaluated at the true data distribution P, e.g.

$$\theta(P) = E(Y^1) = E\left\{Q_0(W)\right\}.$$

- Let $\theta(\hat{P}_n)$ be a plug-in estimator, based on plugging in ML predictions, e.g.

$$\theta(\hat{P}_n) = \hat{E}(Y^1) = \frac{1}{n}\sum_{i=1}^{n} \bar{Q}_n(W_i).$$

- Then we are interested in understanding the difference

$$\theta(\hat{P}_n) - \theta(P).$$

A large sample expansion

- It follows from the mathematical statistics literature that plug-in estimators 'usually' obey the expansion

$$\theta(\hat{P}_n) - \theta(P) \quad = \quad \frac{1}{n}\sum_{i=1}^{n} \phi_P(O_i) - \frac{1}{n}\sum_{i=1}^{n} \phi_{\hat{P}_n}(O_i) + \text{small remainder},$$

where O_i refers to the observed data for subject i.

- Here, $\phi_P(O_i)$ is a mean zero function, called an influence curve.
- E.g. for $\theta(P) = E(Y^1)$, it equals

$$\phi_P(O_i) = \frac{A_i}{g_0(W_i)}\left\{Y_i - Q_0(W_i)\right\} + Q_0(W_i) - \theta(P).$$

- A general theory on how to calculate it for other parameters is beyond the scope of this masterclass.

Zooming in on the first term...

- The first term in the expansion

$$\frac{1}{n}\sum_{i=1}^{n}\phi_P(O_i)$$

  is well understood.
- It is normally distributed in large samples, with mean 0 and variance given by 1 over n times the variance of the influence curve.

Zooming in on the second term...

- The second term
$$-\frac{1}{n}\sum_{i=1}^{n}\phi_{\hat{P}_n}(O_i)$$
is usually not well understood.
- Its randomness originates from the randomness of the data O_i, but also the uncertainty in the machine learning estimators $\hat{P}_n$, which is ill understood.
- Moreover, the complex distribution of such estimators $\hat{P}_n$, may propagate into this term, thereby rendering $\theta(\hat{P}_n)$ biased and non-normal.
- This is the root cause of the previously discussed plug-in bias.

Eliminating plug-in bias

- Since this bias term

$$-\frac{1}{n}\sum_{i=1}^{n}\phi_{\hat{P}_n}(O_i)$$

  is determined by the influence curve,
  we can aim to remove it.
- We discuss 3 strategies to achieve this:
  - one-step plug-in estimators;
  - estimating equations estimators;
  - targeted maximum likelihood.

The one-step plug-in estimator

- The identity

$$\theta(\hat{P}_n) - \theta(P) = \frac{1}{n}\sum_{i=1}^{n}\phi_P(O_i) - \frac{1}{n}\sum_{i=1}^{n}\phi_{\hat{P}_n}(O_i) + \text{small remainder},$$

suggests that we can remove plug-in bias
by adding the bias term to the plug-in estimator:

$$\theta(\hat{P}_n) + \frac{1}{n}\sum_{i=1}^{n}\phi_{\hat{P}_n}(O_i) - \theta(P) = \frac{1}{n}\sum_{i=1}^{n}\phi_P(O_i) + \text{small remainder}.$$

- This delivers the one-step plug-in estimator

$$\theta(\hat{P}_n) + \frac{1}{n}\sum_{i=1}^{n}\phi_{\hat{P}_n}(O_i).$$

A one-step plug-in estimator of $E(Y^1)$

- Starting from a plug-in estimator

$$\hat{E}(Y^1) = \frac{1}{n}\sum_{i=1}^{n} \bar{Q}_n(W_i),$$

we thus calculate a one-step plug-in estimator as

$$\frac{1}{n}\sum_{i=1}^{n} \bar{Q}_n(W_i) + \frac{1}{n}\sum_{i=1}^{n} \frac{A_i}{g_n(W_i)}\left\{Y_i - \bar{Q}_n(W_i)\right\} + \bar{Q}_n(W_i) - \hat{E}(Y^1)$$

$$= \frac{1}{n}\sum_{i=1}^{n} \frac{A_i}{g_n(W_i)}\left\{Y_i - \bar{Q}_n(W_i)\right\} + \bar{Q}_n(W_i).$$

- This is known as a plug-in augmented inverse probability weighted (AIPW) estimator.

A one-step plug-in estimator of $E(Y^1)$

- This result is extremely powerful.
- The one-step plug-in estimator behaves like

$$\theta(P) + \frac{1}{n}\sum_{i=1}^{n} \phi_P(O_i) + \text{small remainder.}$$

- It is thus asymptotically unbiased
  and normal, despite its reliance
  on estimators with a non-standard distribution.

The estimating equations estimator

- Alternatively, we can remove the bias term by calculating the estimator $\hat{\theta}$ as the solution to

$$0 = \frac{1}{n} \sum_{i=1}^{n} \phi_{\hat{P}_n}(O_i),$$

thus effectively setting the bias term to zero.

- This estimator is called the estimating equations estimator.
- It forms the basis of the debiased/double ML literature.

An estimating equations estimator of $E(Y^1)$

- Solving

$$0 = \frac{1}{n}\sum_{i=1}^{n}\frac{A_i}{g_n(W_i)}\left\{Y_i - \bar{Q}_n(W_i)\right\} + \bar{Q}_n(W_i) - \theta(\hat{P}_n),$$

  happens to deliver the one-step plug-in estimator.
- This is not generally the case.

Implementation in R

```
> install.packages("devtools")
> library(devtools)
> install_github("ehkennedy/npcausal")
> library(npcausal)

> set.seed(640)
> n <- 1000
> w1 <- rbinom(n, size=1, prob=0.5)
> w2 <- rbinom(n, size=1, prob=0.65)
> w3 <- round(runif(n, min=0, max=4), digits=3)
> w4 <- round(runif(n, min=0, max=5), digits=3)
> A <- rbinom(n, size=1,prob=
   plogis(-0.4 + 0.2*w2 + 0.15*w3 + 0.2*w4 + 0.15*w2*w4))
> Y <- rbinom(n, size=1,prob=
   plogis(-1 + A -0.1*w1 + 0.3*w2 + 0.25*w3 + 0.2*w4 + 0.15*w2*w4))

#Create data frame with baseline covariates
> W<-data.frame(cbind(w1,w2,w3,w4))
```

Implementation in R

```
##Specify SuperLearner libraries
> SL.library <- c("SL.glm","SL.glm.interaction","SL.ranger")

#AIPW with no split
> aipw <- ate(y=Y, a=A, x=W, nsplits=1, sl.lib=SL.library)
> aipw

$res
    parameter       est         se     ci.ll     ci.ul pval
1      E{Y(0)} 0.6154303 0.02483961 0.5667447 0.6641160    0
2      E{Y(1)} 0.7937511 0.01582655 0.7627311 0.8247712    0
3 E{Y(1)-Y(0)} 0.1783208 0.02898653 0.1215072 0.2351344    0
```

The Targeted Maximum Likelihood estimator

- The Targeted Maximum Likelihood estimator (TMLE) solves the same equation

$$0 = \frac{1}{n}\sum_{i=1}^{n} \phi_{\hat{P}_n}(O_i),$$

but in doing so, ensures that the solution equals the maximum likelihood estimator

$$\hat{E}(Y^1) = \frac{1}{n}\sum_{i=1} \bar{Q}_n(W_i),$$

under a specific parametric submodel,

(the so-called least favourable submodel).

- This results in a substitution estimator, which generally has better performance.

The Targeted Maximum Likelihood estimator of $E(Y^1)$

- Let $\bar{Q}_n^{(0)}(W)$ be the initial ML estimator for $Q_0(W)$.
- We then build a parametric model around $\bar{Q}_n^{(0)}(W)$, with the aim to remove plug-in bias from the plug-in estimator.
- In particular, we will tune the initial estimator $\bar{Q}_n^{(0)}(W)$ to remove plug-in bias in the estimation of $E(Y^1)$.
- For dichotomous Y, fit the logistic regression model

$$\mathrm{logit}E(Y|A, W) = \mathrm{logit}\bar{Q}_n^{(0)}(W) + \delta \frac{A}{g_n(W)}$$

and let the fitted values (based on the MLE for δ) be $\bar{Q}_n^{(1)}(W)$.

The Targeted Maximum Likelihood estimator of $E(Y^1)$

The maximum likelihood score for δ then solves the equation

$$0 = \frac{1}{n}\sum_{i=1}^{n} \frac{A_i}{g_n(W_i)} \left\{ Y_i - \bar{Q}_n^{(1)}(W_i) \right\},$$

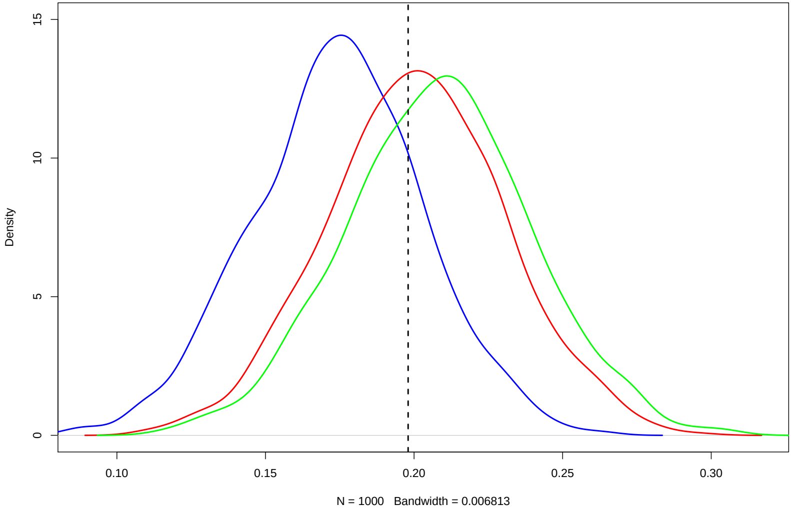
so that the AIPW estimator becomes

$$\begin{aligned} \hat{E}(Y^1) &= \frac{1}{n}\sum_{i=1}^{n} \frac{A_i}{g_n(W_i)} \left\{ Y_i - \bar{Q}_n^{(1)}(W_i) \right\} + \bar{Q}_n^{(1)}(W_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} \bar{Q}_n^{(1)}(W_i). \end{aligned}$$

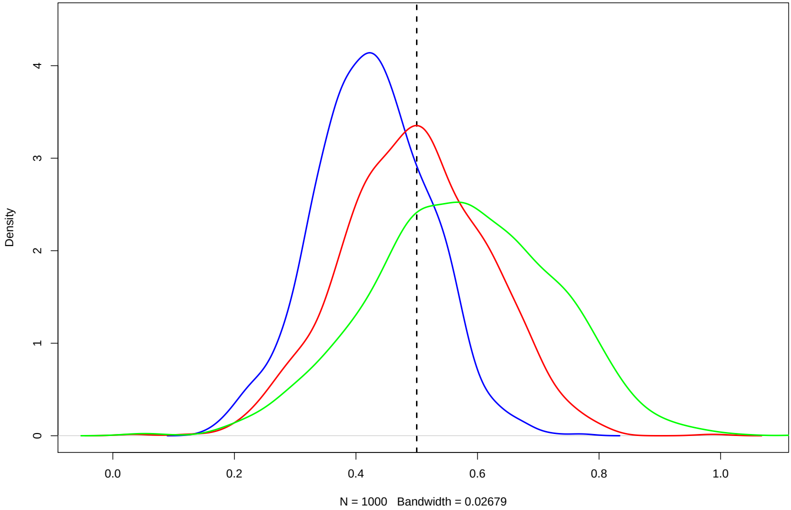
Implementation in R

```
>  install.packages("tmle")
>  library(tmle)

#TMLE with no split
> tmle_est <- tmle(Y=Y,A=A,W=W,family="binomial",
    Q.SL.library=SL.library,g.SL.library=SL.library)
> tmle_est

 Additive Effect
   Parameter Estimate:  0.18212
   Estimated Variance:  0.00098828
              p-value:  6.9064e-09
    95% Conf Interval: (0.1205, 0.24374)

 Additive Effect among the Treated
   Parameter Estimate:  0.17928
   Estimated Variance:  0.00099416
              p-value:  1.3021e-08
    95% Conf Interval: (0.11748, 0.24107)

 Additive Effect among the Controls [...]
```

Naïve (blue), plug-in AIPW (red) and TMLE (green)



Naïve (blue), plug-in AIPW (red) and TMLE (green)



An alternative Targeted Maximum Likelihood estimator

- Alternatively, consider the logistic regression model

$$\operatorname{logit} E(Y|A=1,W) = \operatorname{logit} \bar{Q}_n^{(0)}(W) + \delta,$$

fitted in the treated
using weighted MLE with weights $1/g_n(W)$,
and let the fitted values (based on the MLE for δ) be $\bar{Q}_n^{(1)}(W)$.

- The maximum likelihood score for δ then again solves

$$0 = \frac{1}{n}\sum_{i=1}^{n} \frac{A_i}{g_n(W_i)}\left\{Y_i - \bar{Q}_n^{(1)}(W_i)\right\},$$

so that the AIPW estimator becomes

$$\hat{E}(Y^1) = \frac{1}{n}\sum_{i=1}^{n} \bar{Q}_n^{(1)}(W_i).$$

Summary so far

- Plug-in bias can be removed via one-step plug-in estimators, debiased machine learning or targeted learning.
- This requires knowledge of the influence curve of the parameter of interest.
- For many common parameters, this influence curve has been documented in the literature.

  (Levy, 2019)
- Plug-in bias typically disappears with increasing sample size, but removing it is nonetheless essential.
- In the next part, you will learn why.
- You will also develop a better understanding when this works, and how confidence intervals can be obtained.

Lab session

- Re-load the SuperLearner predictions
  and use them to calculate
  - the plug-in AIPW estimator;
  - a TMLE (do this both ‘by hand’ and using the `tmle` package).
- Start with the case of $n = 500, p = 5$,
  and consider the more challenging settings later.