Double Robust Estimation: Introduction to AIPW and TMLE

Ashley I Naimi

June 2022

Contents

1 Doubly Robust Methods: Some Basic Intuition

To gain some very basic intuition about what double-robust methods do, we'll use a simple simulation. Before outlining the contours of this simulation, it's important to understand how we might be **misled** by this simple illustration. Think of the following simulation as a oversimplified illustration of the property of double robustness.

However, in realistic settings, we would not rely on the logic of the following simulation to justify using double robust methods. In the simulation below, we generate data compatible with the causal diagram presented in Figure 1:

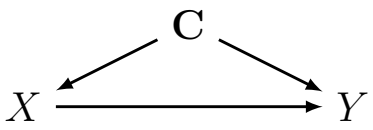


Figure 1: Causal diagram representing the structure from which the simple simulated data were generated.

```
# the inverse logit function, to
# simulate from logistic model (NB: the
# inverse logit function is the softmax
# function)
expit <- function(x) {
    (1/(1 + exp(-x)))
}
# sample size
n <- 2500

# simulation loop
sim_dat <- function(index) {
    set.seed(index)
    # confounders
    c1 <- rbinom(n, 1, 0.5)
    c2 <- rbinom(n, 1, 0.5)

    # propensity score model
    pi_x <- expit(-1.5 + log(2.5) * c1 +
        log(2.5) * c2)
    x <- rbinom(n, 1, pi_x)

    # outcome model
    mu_y <- expit(-1.5 + log(1.75) * x +
```

```
    log(2.5) * c1 + log(2.5) * c2)
y <- rbinom(n, 1, mu_y)

# simulated data
a <- data.frame(x, y, c1, c2)

# regression 1: correctly specified
mod_true <- glm(y ~ x + c1 + c2, data = a,
    family = binomial("logit"))
mu1 <- mean(predict(mod_true, newdata = transform(a,
    x = 1), type = "response"))
mu0 <- mean(predict(mod_true, newdata = transform(a,
    x = 0), type = "response"))
ate_true <- mu1 - mu0
ate_true

# regression 2: misspecified, one
# confounder left out
mod_noconf <- glm(y ~ x + c1, data = a,
    family = binomial("logit"))
mu1 <- mean(predict(mod_noconf, newdata = transform(a,
    x = 1), type = "response"))
mu0 <- mean(predict(mod_noconf, newdata = transform(a,
    x = 0), type = "response"))
ate_noconf <- mu1 - mu0
ate_noconf

# regression 3: double robust
# estimation correctly specified PS
# model
a$propensity_score <- glm(x ~ c1 + c2,
    data = a, family = binomial("logit"))$fitted.values
# stabilized inverse probability
# weights
```

```r
a$sw <- (mean(a$x)/a$propensity_score) *
    a$x + ((1 - mean(a$x))/(1 - a$propensity_score)) *
    (1 - a$x)

# misspecified outcome model
mod_dr1 <- glm(y ~ x + c1, weights = sw,
    data = a, family = binomial("logit"))
mu1 <- mean(predict(mod_dr1, newdata = transform(a,
    x = 1), type = "response"))
mu0 <- mean(predict(mod_dr1, newdata = transform(a,
    x = 0), type = "response"))
ate_dr1 <- mu1 - mu0
ate_dr1

# regression 5: double robust
# estimation misspecified PS model
a$propensity_score <- glm(x ~ c1, data = a,
    family = binomial("logit"))$fitted.values
# stabilized inverse probability
# weights
a$sw <- (mean(a$x)/a$propensity_score) *
    a$x + ((1 - mean(a$x))/(1 - a$propensity_score)) *
    (1 - a$x)

# correctly specified outcome model
mod_dr2 <- glm(y ~ x + c1 + c2, weights = sw,
    data = a, family = binomial("logit"))
mu1 <- mean(predict(mod_dr1, newdata = transform(a,
    x = 1), type = "response"))
mu0 <- mean(predict(mod_dr1, newdata = transform(a,
    x = 0), type = "response"))
ate_dr2 <- mu1 - mu0
ate_dr2
```

```
    res <- data.frame(rbind(c(Method = "True",
        Estimate = ate_true), c(Method = "Confounded",
        Estimate = ate_noconf), c(Method = "Double Robust 1",
        Estimate = ate_dr1), c(Method = "Double Robust 2",
        Estimate = ate_dr2)))

    return(res)
}

res <- lapply(1:500, function(x) sim_dat(index = x))

res <- do.call(rbind, res)

res[, 2] <- as.numeric(res[, 2])

plot1 <- ggplot(res) + geom_density(aes(x = Estimate,
    group = Method, fill = Method), alpha = 0.2) +
    scale_x_continuous(expand = c(0, 0),
        limits = c(0, 0.3)) + scale_y_continuous(expand = c(0,
    0)) + scale_fill_discrete(name = "") +
    guides(fill = guide_legend(nrow = 2,
        byrow = TRUE))

ggsave(here("figures", "simulation_results.pdf"),
    plot = plot1, width = 10, height = 10,
    units = "cm")
```

In the code above, we generate simulated data from a simple triangle causal structure. To estimate the effect correctly using either IP weighting or g computation, we have to adjust for both confounders `c1` and `c2`. If we leave out one of the confounders from the g compuation outcome model, we end up with a biased estimator. However, if we use a doubly robust approach, *and at least one of the models has both confounders included*, we can still recover an unbaised estimator.

The results from our simulation example are shown in Figure 2. What they

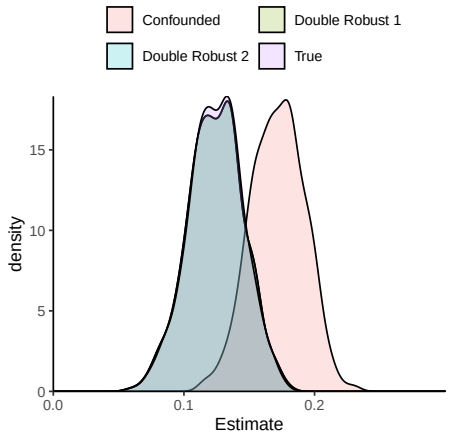


Figure 2: Distribution of 500 Estimates from the Simple Simulation Illustration of DR Estimation

show is that we can still recover the effect of interest, even if the outcome model or the propensity score model is misspecified, as long as long as at least one of them is correctly specified.

This is the basic idea behind double robustness: it gives us "two chances" to get things right (Jonsson-Funk et al., 2011). If we miss one chance, we can rely on the other. But if we miss both, we get things wrong.

The problem with this example is that it is artificial. Quantitative analyses that are used to make important decisions should be conducted carefully and methodically, and are usually conducted in group settings with several people involved. Thus, to justify the use of double robust estimators on the basis of the argument that "if one forgets to include a confounder, we still get the right answer". One would never simply "forget" to include confounders in a model for a double robust estimator.

This reasoning also distorts the main reason that one would use a double robust estimator: under relatively mild conditions, they remain unbiased, with asymptotically nominal confidence interval coverage, even when machine learning methods are used to fit the exposure and outcome models (van der Laan and Rubin, 2006, Kennedy and Balakrishnan (2017)). In effect, doubly robust methods can mitigate or resolve problems caused by the curse of dimensionality.

More accurately, double robust estimators can be $\sqrt{N}$-consistent, asymptotically normal, and optimally efficient even if the estimators used to quantify the outcome model and the propensity score model are converging at slower nonparametric rates. In other words, the doubly robust estimator is less sus-

ceptible to the curse of dimensionality. This is because each singly robust estimator is combined in a way that their combined convergence rates are as fast or faster than the convergence rate of each estimator separately. In particular, if the outcome model and the propensity score models are converging to their targets at least faster than $n^{-1/4}$ rates (technically, in L_2 norm), the doubly robust estimator will behave (asymptotically) just as if both the outcome and propensity score models were estimated with correct parametric models.

Importantly, $n^{-1/4}$ rates can be attained nonparametrically under relatively weak (smoothness, sparsity, or other structural) assumptions (Györfi et al., 2002, Wasserman (2006)).

This improved performance of is the key benefit behind double robust estimators, since it allows us to implement machine learning methods to estimate causal effects without incurring serious performance penalties from the curse of dimensionality.

2 Doubly Robust Methods: AIPW and TMLE

In the sections that follow, let's consider a simple setting with a single binary exposure (X), a set of continuous confounders C, as we did in the simulation. Let's also assume we are interested in the exposure effect on a single continuous outcome (Y) measured at the end of a study follow up follow-up.

In an observational cohort study to estimate the effect of X on Y, C might be assumed a minimally sufficient adjustment set (Greenland et al., 1999), and the outcome and exposure would be assumed generated according to some unknown models, for example:

$$E(Y \mid X, \mathbf{C}) = g(X, C), \qquad \text{(Model 1)}$$

$$P(X = 1 \mid \mathbf{C}) = f(C). \qquad \text{(Model 2)}$$

In a previous section, we assumed $f(C)$ and $g(X, C)$ could be represented with parametric regression models. Here, we make no such assumptions.

To estimate the average treatment effect $\psi = E(Y^{x=1} - Y^{x=0})$ under the relevant identifiability conditions, we can use double robust methods, where predictions from both the exposure and outcome models are combined into a single estimator to quantify the effect of interest.

For example, using predictions from both Models Model 2 and Model 1, ψ can be estimated as:

$$\hat{\psi}_{aipw} = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{(2X_i-1)[Y_i-\hat{g}_i(X,\mathbf{C})]}{(2X_i-1)\hat{f}_i(\mathbf{C})+(1-X_i)}+\hat{g}_i(X=1,\mathbf{C})-\hat{g}_i(X=0,\mathbf{C})\right\}. \tag{1}$$

Equation 1 is an augmented inverse probability weighted estimator, and will converge to the true value as the sample size grows if either $f(\mathbf{C})$ or $g(X,\mathbf{C})$, but not necessarily both, are consistently estimated. The estimator 1 can be viewed as either a bias-corrected version of the g computation estimator (where the correction is the term incorporating the propensity score defined in Model 2), or an efficiency enhanced version of the IPW estimator (where the enhancement is the term incorporating the outcome model defined in Model 1) (Daniel, 2018).

Alternatively, Model 2 can be used to "update'' Model 1 via targeted minimum loss-based estimation[1]: (Rose and van der Laan, 2011)$^{(p72-3)}$

[1] Equivalently, targeted maximum likelihood estimation

$$\hat{\psi}_{tmle} = \frac{1}{N}\sum_{i=1}^{N}\left\{\hat{g}_i^u(X=1,\mathbf{C})-\hat{g}_i^u(X=0,\mathbf{C})\right\}, \tag{2}$$

where $\hat{g}_i^u(X=x,\mathbf{C})$ are predictions from an "updated'' outcome model. For the average treatment effect, this outcome model is updated by first generating a modified inverse probability weight, defined as:

$$H(X,\mathbf{C}) = \begin{cases} \frac{1}{\hat{f}_i(\mathbf{C})} & \text{if } X=1 \\ -\frac{1}{1-\hat{f}_i(\mathbf{C})} & \text{otherwise} \end{cases}$$

and then including this inverse probability weight in a no-intercept logistic regression model for the outcome that includes the previous outcome predictions $\hat{g}_i(X,\mathbf{C})$ as an offset. The $\hat{g}_i^u(X=x,\mathbf{C})$ predictions are then generated from this model by setting X to 1 and then to 0 for all individuals in the sample. TMLE is asymptotically equivalent to equation 1 but can have better performance in finite-samples (Gruber and van der Laan, 2012).

There are many different variations of these doubly robust estimators (Bang and Robins, 2005, Tan (2010),Kennedy (2022)). However, the ones presented above represent the most commonly used (and studied) varieties.

3 Software for Double Robust Estimation

There are many software options available to implement double robust methods. Even in R, there are a handful of packages that can be used to implement AIPW and TMLE. Figure 3 provides a comparison of many of these packages.

Packages	Version Evaluated	Doubly Robust Estimator	Available Model	Cross-fitting	Different Covariate Sets	Exposure Type	Propensity Score Truncation	Outcome Type	Missing Data Support	ATE Estimate Scale	SE Type	Parallel Processing
AIPW	0.6.3.1	AIPW	Super Learner	Yes	Yes	Binary[c]	Yes	Binary & Continuous	Missing Outcome	RD, RR, OR	Asymptotic	Yes
CausalGAM	0.1-4	AIPW	GAMs	No	Yes	Binary	Yes	Binary & Continuous	No	RD	Asymptotic, Sandwich, Bootstrap	No
npcausal	0.1.0	AIPW	Super Learner	Yes	Yes[a]	Binary, Categorical, Continuous	No	Binary & Continuous	Missing Outcome	RD	Asymptotic	No
tmle	1.4.0.1	TMLE	Super Learner	Yes	Yes[b]	Binary[c]	Yes	Binary & Continuous	Missing Outcome	RD, RR, OR	Asymptotic	No
tmle3	0.1.7	TMLE	Super Learner	Yes	Yes	Binary, Categorical, Continuous	Yes	Binary & Continuous	Missing Outcome	RD, RR, OR	Asymptotic	Yes

AIPW, Augmented Inverse Probability Weighting; ATE, Average Treatment Effect; GAM, Generalized Additive Model; TMLE, Targeted Maximum Likelihood Estimation; RD, Risk Difference; RR, Risk Ratio; OR, Odds Ratio; SE, Standard Errors
[a] Users need to manually input propensity scores for different covariate sets.
[b] When the support of different covariate sets is enabled, tmle only uses generalized linear model for estimation.
[c] Continuous and categorical exposures can be used but need to be dichotomized (17).

Figure 3: Comparisons of R Packages for Implementing Doubly Robust Estimators.

Overall, only the `CausalGAM` package is limited to using generalized additive models for the propensity score and outcome models. All other packages employ the use of the Super Learner, which is an ensemble algorithm that can combine a whole host of parametric and machine learning methods into a single estimation algorithm. Additionally, `CausalGAM` does not use cross-fitting, which is required to obtain valid standard errors when highly flexible methods are used to estimate the propensity score and outcome models.

Here, we will use R and be focusing on the `tmle3` package, which implements targeted minimum loss based estimation, and the `AIPW` package, which implements augmented inverse probability weighting.

4 TMLE3 in R

The `tmle3` package is a part of the `tlverse`, which consists of a set of packages and functions with a common underlying language structure devoted to **targeted learning** (Rose and van der Laan, 2011).

Information on the `tlverse` is available here: `https://tlverse.org/`. Currently, the not all of the packages in the `tlverse` are available on CRAN. Thus, one has to install directly from GitHub using either the `devtools` or the

`remotes` (preferred) package:

```
remotes::install_github("tlverse/tlverse")
library(tmle3)
library(sl3)
```

Two key packages within the `tlverse` are the `tmle3` and `sl3` packages.[2] The former runs targeted minimum loss-based estimation and the latter runs the super learner. Here is a quick overview of how `tmle3` and `sl3` functions operate:

[2] It is common to encounter errors in the installation process, especially if you've never used `install_github` before. In my experience, re-running the installation code will sometimes work. Otherwise, the best approach is to web search the specific error code you received to troubleshoot.

```
library(tidyverse)

remotes::install_github("tlverse/tlverse")
library(tmle3)
library(sl3)

# Import NHEFS data
nhefs <- read_csv(here("data", "nhefs.csv")) %>%
    mutate(wt_delta = as.numeric(wt82_71 >
        median(wt82_71)))

# Quick view of data
dim(nhefs)
```

```
## [1] 1394   12
```

```
names(nhefs)
```

```
##  [1] "seqn"     "qsmk"     "sex"      "age"      "income"   "sbp"
##  [7] "dbp"      "price71"  "tax71"    "race"     "wt82_71"  "wt_delta"
```

```
# using the sl3 function create the
# super learner, that contains a simple
# average estimator and a glm
```

```
# estimator.
lrnr_glm <- make_learner(Lrnr_glm)
lrnr_mean <- make_learner(Lrnr_mean)
sl <- Lrnr_sl$new(learners = list(lrnr_glm,
    lrnr_mean))

learner_list <- list(Y = sl, A = sl)

#############################################################################

# PREPARE THE THINGS WE WANT TO FEED IN
# TO TMLE3 ate_spec defines the effect
# we want to target, and the exposure
# levels we want to compare
ate_spec <- tmle_ATE(treatment_level = 1,
    control_level = 0)
# nodes define the variables we want to
# adjsut for, and the exposure and the
# outcome
nodes <- list(W = c("sex", "age", "income",
    "sbp", "dbp", "price71", "tax71", "race"),
    A = "qsmk", Y = "wt_delta")

# Run the tmle3 function using the
# pieces defined above
tmle_fit <- tmle3(ate_spec, nhefs, nodes,
    learner_list)

tmle_fit$summary$psi_transformed
```

```
## [1] 0.1375596
```

```
tmle_fit$summary$lower_transformed
```

```
## [1] 0.07769168
```

```
tmle_fit$summary$upper_transformed
```

```
## [1] 0.1974276
```

Over the next several days, we will revisit and unpack the above code to enable you to better understand the decision points when conducting an analysis using TMLE.

5 AIPW in R

The `AIPW` package is a recently developed package to fit the augmented IPW estimator. It can be implemented with the super learner via `sl3` or the older `SuperLearner` package (Polley et al., 2016)[3].

[3] Importantly, if you install the CRAN version of AIPW, you must use the older `SuperLearner` package, which we demonstrate here.

Information on the `AIPW` packages is available here: `https://yqzhong7.github.io/AIPW/`, and general aspects of the program were recently published in the American Journal of Epidemiology (Zhong et al., 2021). In contrast to the `tlverse`, the `AIPW` package is available on CRAN, and can thus be installed using standard code:

```
install.packages("AIPW", repos = "http://lib.stat.cmu.edu/R/CRAN",
    dependencies = T)

library(AIPW)

install.packages("SuperLearner", repos = "http://lib.stat.cmu.edu/R/CRAN",
    dependencies = T)

library(SuperLearner)
```

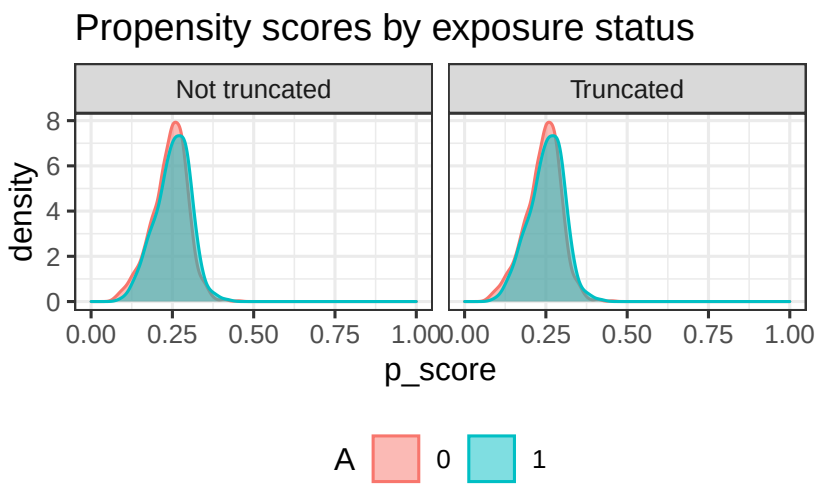
```
## Loading required package: nnls

## Loading required package: gam

## Loading required package: splines

## Loading required package: foreach
```

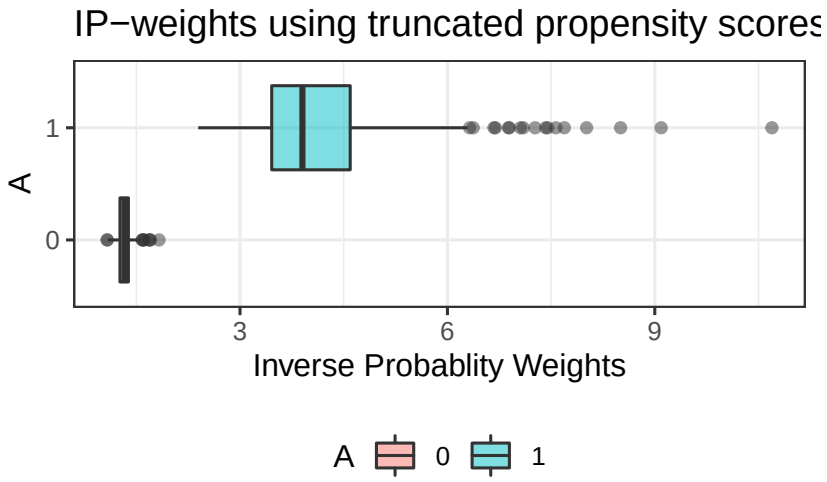
```
##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded gam 1.20

## Super Learner

## Version: 2.0-28

## Package created on 2021-05-04
```

Once installed, the `AIPW` function can be implemented basically as:

```
# we will see, the AIPW function
# includes important visual diagnostics
# which can be implemented with the
# ggplot library
library(ggplot2)

# Import NHEFS data, again
nhefs <- read_csv(here("data", "nhefs.csv")) %>%
    mutate(wt_delta = as.numeric(wt82_71 >
        median(wt82_71)))

# using the SuperLearner function with
# simple average estimator and a glm
# estimator.

sl <- c("SL.mean", "SL.glm")

outcome <- nhefs$wt_delta
exposure <- nhefs$qsmk
covariates <- nhefs[, c("sex", "age", "income",
    "sbp", "dbp", "price71", "tax71", "race")]
```

```
AIPW_SL <- AIPW$new(Y = outcome, A = exposure,
    W = covariates, Q.SL.library = sl, g.SL.library = sl,
    k_split = 3, verbose = FALSE)$fit()$summary(g.bound = 0.025)$plot.p_score()$plot.ip_weights()
```






```
print(AIPW_SL$result[3, ], digits = 2)
```

```
## Estimate       SE  95% LCL  95% UCL        N
##  1.4e-01  3.1e-02  7.9e-02  2.0e-01  1.4e+03
```

This quick introduction to the functions will be expanded upon in subsequent sections. At this point, it is important to note a couple of key aspects of what we just did with these functions, and how we should interpret the results:

1) TMLE and AIPW both returned risk differences of 0.14 (95% Confidence Interval: 0.08, 0.2). However, this estimate should only be interpreted as an average treatment effect $E(Y^1 - Y^0)$ in the context of the identifiability assumptions covered earlier. If these (unverifiable) assumptions don't hold this estimate may not align with the underlying (true) average treatment effect.

2) To obtain these TMLE and AIPW estimates, we used a stacking algorithm that combined a simple mean estimator and a generalized linear model. This was to illustrate basics in a simple setting. In realistic settings, the ideal approach would be to include many different algorithms into the super learner to more flexibly model the exposure and the outcome.

References

Heejung Bang and James M. Robins. Doubly robust estimation in missing data and causal inference models. *Biometrics*, 61(4):962—973, 2005.

Rhian M. Daniel. *Double Robustness*. John Wiley & Sons, Ltd, 2018.

Sander Greenland, Judea Pearl, and JM Robins. Causal diagrams for epidemiological research. *Epidemiol*, 10(1):37–48, 1999.

Susan Gruber and Mark J van der Laan. tmle: An r package for targeted maximum likelihood estimation. *Journal of Statistical Software*, 51(13):1–35, 2012.

L. Györfi, M. Kohler, A. Krzyzak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer, New York, NY, 2002.

Michele Jonsson-Funk, Daniel Westreich, Chris Wiesen, Til Stürmer, M. Alan Brookhart, and Marie Davidian. Doubly robust estimation of causal effects. *Am J Epidemiol*, 173(7):761–767, 2011.

Edward H. Kennedy. Semiparametric doubly robust targeted double machine learning: a review. *arXiv:2203.06469 [stat]*, 2022.

Edward H Kennedy and Sivaraman Balakrishnan. Discussion of "Data-driven confounder selection via Markov and Bayesian networks" by Jenny Häggström. *Biometrics*, In Press, 2017.

Eric Polley, Erin LeDell, Chris Kennedy, and Mark van der Laan. *SuperLearner: Super Learner Prediction*, 2016. URL https://github.com/ecpolley/SuperLearner. R package version 2.0-22.

Sherri Rose and Mark J van der Laan. *Targeted learning: causal inference for observational and experimental data*. Springer, New York, NY, 2011.

Zhiqiang Tan. Bounded, efficient and doubly robust estimation with inverse weighting. *Biometrika*, 97(3):661–682, 2010.

Mark J. van der Laan and Daniel Rubin. Targeted maximum likelihood learning. *Int J Biostat*, 2(1):Article 11, 2006.

Larry Wasserman. *All of nonparametric statistics*. Springer, New York; London, 2006.

Yongqi Zhong, Edward H Kennedy, Lisa M Bodnar, and Ashley I Naimi. Aipw: An r package for augmented inverse probability weighted estimation of average causal effects. *American Journal of Epidemiology*, In Press, 2021.