# R Implementation - Introduction to Double Robust Estimation for Causal Inference

**Laura B. Balzer, PhD**
University of Massachusetts-Amherst
lbalzer@umass.edu
Twitter: @LauraBBalzer

August 26, 2018

***During the workshop, we will complete this R exercise together. After the workshop, an answer key will be distributed.***

## 1 Preliminaries

### 1.1 Workshop Goals:

(a.) Introduce the Roadmap for causal inference with an applied example (Petersen and van der Laan, 2014; Petersen and Balzer, 2014; Balzer et al., 2016).

(b.) In R, implement the simple substitution estimator (a.k.a. parametric G-computation) and targeted maximum likelihood estimation (TMLE) with the ensemble algorithm Super Learner for estimation of the effect of a binary exposure.

(c.) Show the ltmle and drtmle packages in R (Lendle et al., 2017; Benkeser, 2018).

(d.) Time-permitting: Explore the performance of the estimators with simulations.

### 1.2 The Observed Data:

Load the data set CausalWorkshop.csv and assign it to object data.

```
> data<- read.csv("CausalWorkshop.csv")
> # Hint 1. Do not type or copy the >. This represents the prompt in an R console.
> # Hint 2. Press enter or return to evaluate the code.
> # Hint 3. Hashtags are used in R for making comments.  #YayTMLE!
```

Use the tail function to view the last 6 observations, and the dim function to obtain the dimensions of the data set:

```
> tail(data)
```

```
    W1         W2          W3         W4 A           Y
95   0 0.43866653  0.62099901  0.6736281 0 0.041108280
```

```
96   0 0.47763995  0.40945733  0.8789507 0 0.087362023
97   1 0.38601726  0.10916835 -0.9633303 0 0.045684975
98   1 0.51518289  1.02451436  1.6171894 1 0.000042700
99   0 0.05015548 -1.62360003 -1.1573370 0 0.005801751
100  1 0.88454275  0.05359586 -0.6562194 0 0.103371039
```

```
> dim(data)
```

```
[1] 100   6
```

The observed data consist of the following variables

- $W1$: Binary baseline covariate (values 0 and 1)
- $W2$: Continuous baseline covariate (range from 0 to 1)
- $W3$: Continuous baseline covariate (centered at 0)
- $W4$: Continuous baseline covariate (centered at 0)
- $A$: The exposure (1 for exposed; 0 for unexposed)
- $Y$: The outcome (range from 0 to 1).

Let $W = (W1, W2, W3, W4)$ represent the vector of baseline covariates (i.e. measured confounders). Then our observed data are

$$O = (W, A, Y)$$

with $A$ as the exposure and $Y$ as the outcome. There are $n = 100$ observations in the data set. We will use the following notation:

- $\mathbb{P}$: the distribution of the observed data $O = (W, A, Y)$
- $\mathbb{E}(Y|A, W)$: the conditional mean of the outcome, given the exposure and covariates
- $\mathbb{P}(A = 1|W)$: the conditional probability of being exposed, given the covariates (i.e. the propensity score)
- $\mathbb{P}(W)$: the marginal distribution of baseline covariates

In this workshop, we will implement three algorithms for estimating the G-Computation identifiability result (Robins, 1986):

$$\begin{aligned}\Psi(\mathbb{P}) &= \sum_w \big[\mathbb{E}(Y|A = 1, W = w) - \mathbb{E}(Y|A = 0, W = w)\big]\mathbb{P}(W = w) \\ &= \mathbb{E}\big[\mathbb{E}(Y|A = 1, W) - \mathbb{E}(Y|A = 0, W)\big]\end{aligned}$$

This is our statistical parameter, which is also called the estimand.

Before doing any estimation, it is useful to "set the seed" in `R`. This ensures that the same values are generated each time each time we run **all** of our code.

```
> set.seed(1)
> #More information about this function can be accessed with
> ?set.seed
```

# 2 The simple substitution estimator (i.e. parametric G-computation)

- **Step 1. Estimate the conditional mean outcome $\mathbb{E}(Y|A, W)$ with parametric regression.**

  First, we run a regression of the outcome $Y$ on the exposure $A$ and covariates $W$. This provides an estimate the conditional mean outcome $\hat{\mathbb{E}}(Y|A, W)$. Here, we are using the "hat" notation to denote an estimate based on our observed data.

Suppose we were willing to assume the conditional expectation of the outcome $Y$ is accurately described by the following function of the exposure $A$ and covariates $W$:

$$logit\big[\mathbb{E}(Y|A,W)\big] = \beta_0 + \beta_1 A + \beta_2 W1 + \beta_3 W2 + \beta_4 W3 + \beta_5 W4$$

where $logit(x) = log(x/1-x)$ is the log-odds.

(a.) Use the `glm` function to fit the conditional mean function $\mathbb{E}(Y|A,W)$ with main terms parametric logistic regression.

```
> outcome.regression<- glm(Y~A+W1+W2+W3+W4, family='binomial', data=data)
```

The argument `family='binomial'` specifies logistic regression, and the argument `data=data` specifies the data set for fitting the regression.

Logistic regression is appropriate for a binary or bounded outcome in [0,1] (e.g. a proportion or a probability). Using logistic regression over linear regression ensures that the bounds on the outcome are respected during the estimation process and provides robustness under strong confounding and/or rare outcomes. Therefore, ignore the following warning message.

```
Warning message:
In eval(expr, envir, enclos) : non-integer #successes in a binomial glm!
```

(b.) We can see a summary of the output of regression fit with

```
> summary(outcome.regression)
>
```

- **Step 2: Using the fitted regression, obtain the predicted outcomes.**

Next, we obtain the expected (predicted) outcome for each observation under the exposure and given the covariates $\hat{\mathbb{E}}(Y|A=1,W)$ as well as the expected (predicted) outcome for each observation under no exposure and given the covariates $\hat{\mathbb{E}}(Y|A=0,W)$.

(a.) Copy the data set `data` into two new data frames `data.txt` and `data.untxt`. Then set all units in `data.txt` to be treated/exposed (`A=1`) and all units in `data.untxt` to be untreated/unexposed (`A=0`). The columns of a data frame can be accessed with the `$` operator.

```
> # copying the observed data
> data.txt<- data.untxt <- data
> # set A=1 in the data.txt and A=0 in data.untxt
> data.txt$A <-1
> data.untxt$A <- 0

> # Verify with the colMeans function (which takes the mean in each data column)
> colMeans(data.txt)
> colMeans(data.untxt)
>
```

(b.) Use the `predict` function to get the expected outcome for each observation $i$ given the exposure and its covariates $\hat{\mathbb{E}}(Y_i|A_i=1,W_i)$.

```
> predict.outcome.txt<- predict(outcome.regression, newdata=data.txt, type='response')
```

The argument `newdata=data.txt` specifies that we want to predict the outcomes using as input the data set `data.txt`, where $A=1$ for all units. The argument `type='response'` specifies that we want the predictions returned on the original scale of the response (not the log-odds).

We have created a vector `predict.outcome.txt`, which contains the predicted outcomes for all units, given the exposure and their measured covariates. View the predicted outcomes for the first 6 units under the exposure with `head` function.

```
> head(predict.outcome.txt)
>
```

(c.) Use the `predict` function to get the expected outcome for each observation $i$ given no exposure and its covariates $\hat{\mathbb{E}}(Y_i|A_i = 0, W_i)$.

```
> predict.outcome.untxt<- predict(outcome.regression, newdata=data.untxt, type='response')
```

The argument `newdata=data.untxt` specifies that we want to predict the outcomes using as input `data.untxt`, where $A = 0$ for all units. As before, we want the predicted outcomes (i.e. `type='response'`).

```
> # Viewing the predicted outcomes for the first 6 units under no exposure:
> head(predict.outcome.untxt)
>
```

- **Step 3. Estimate the statistical parameter by substituting the predicted outcomes into the G-Computation formula.**

Finally, we estimate $\Psi(\mathbb{P})$ by averaging the difference in the predicted outcomes. This step standardizes (marginalizes) our estimates to the empirical distribution of confounders:

$$\begin{aligned}\Psi_{SimpleSubs}(\hat{\mathbb{P}}) &= \frac{1}{n}\sum_{i=1}^{n}\left[\hat{\mathbb{E}}(Y_i|A_i = 1, W_i) - \hat{\mathbb{E}}(Y_i|A_i = 0, W_i)\right] \\ &= \underbrace{\frac{1}{n}\sum_{i=1}^{n}\hat{\mathbb{E}}(Y_i|A_i = 1, W_i)}_{\text{Est. of } \mathbb{E}[\mathbb{E}(Y|A=1,W)]} - \underbrace{\frac{1}{n}\sum_{i=1}^{n}\hat{\mathbb{E}}(Y_i|A_i = 0, W_i)}_{\text{Est. of } \mathbb{E}[\mathbb{E}(Y|A=0,W)]}\end{aligned}$$

where $\hat{\mathbb{P}}$ is the empirical distribution, which places weight $1/n$ on each observation $O_i = (W_i, A_i, Y_i)$. The sample average is the non-parametric maximum likelihood estimator (NPMLE) of the covariate distribution.

```
> # estimate of E[E(Y|A=1,W)]
> mean(predict.outcome.txt)
> # estimate of E[E(Y|A=0,W)]
> mean(predict.outcome.untxt)
> # our point estimate of Psi(P)
> Simple.Subs <- mean(predict.outcome.txt - predict.outcome.untxt)
> Simple.Subs
```

- Consistency of the simple (non-targeted) substitution estimator depends on consistent estimation of outcome regression: $\mathbb{E}(Y|A, W)$. If our parametric regression model is incorrect, our point estimates can be biased and inference misleading.
- An estimator is *consistent* if the point estimates converge (in probability) to the estimand as sample size $n \to \infty$.

# 3 Targeted Maximum Likelihood Estimation (TMLE)

- **Step 1. Estimate the conditional mean outcome $\mathbb{E}(Y|A, W)$ with Super Learner.**

First, we use Super Learner to flexibly estimate the conditional expectation of the outcome $Y$, given the exposure $A$ and covariates $W$.

(a.) Use the `library` function to load the `SuperLearner` package (Polley et al., 2018).

```
> library("SuperLearner")
>
```

(b.) Specify the Super Learner library with the following algorithms: main terms logistic regression (`SL.glm`), main terms logistic regression with all possible pairwise interactions (`SL.glm.interaction`), and generalized additive models (`SL.gam`) (Hastie and Tibshirani, 1990; Hastie, 2016). We are using this simple library for pedagogic purposes. In practice, we would want to use a larger library with a mixture of simple (e.g. parametric) and more aggressive libraries.

```
> SL.library<- c("SL.glm", "SL.glm.interaction", "SL.gam")
> # note these have to be in quotations
>
```

(c.) As input to the `SuperLearner` algorithm, we need to specify the outcome-for-prediction, denoted `Y`, as well as the predictors, denoted `X`. In this case, we are interested in predicting the outcome $Y$ with the measured covariates $W$ and the exposure $A$. Therefore, in the following, we denote the outcome-for-prediction with `Y=data$Y` and the predictors as the baseline covariates and the exposure with `X=subset(data, select=-Y)`. The `subset` function is excluding the outcome $Y$ from `data` and retaining (selecting) the covariates $W$ and exposure $A$.

```
> SL.outcome.regression<- SuperLearner(Y=data$Y, X=subset(data, select=-Y),
                    SL.library=SL.library, family="binomial")
```

We are also specifying the library and the appropriate `family`. The resulting object is called `SL.outcome.regression`. As before, ignore the warning message "In eval(expr, envir, enclos) : non-integer #successes in a binomial glm!"

```
> # We can examine the output by typing
> SL.outcome.regression
>
```

The `Risk` column gives the cross-validated estimates of the risk (i.e. expected loss) for each algorithm averaged across the 10 folds. The `Coef` column gives the weight of each algorithm in the convex combination.

- **Step 2. Using the Super Learner fit, obtain the predicted outcomes.**

Next, we use the Super Learner fit to obtain initial estimates of the expected (predicted) outcome for each observation given the observed exposure and the covariates $\hat{\mathbb{E}}(Y|A,W)$, given the treatment/exposure and covariates $\hat{\mathbb{E}}(Y|A=1,W)$, and given no treatment/exposure and the covariates $\hat{\mathbb{E}}(Y|A=0,W)$.

(a.) Use the `predict` function to obtain initial estimates of the expected outcome, given the observed exposure and covariates $\hat{\mathbb{E}}(Y|A,W)$.

```
> SL.predict.outcome<- predict(SL.outcome.regression, newdata=data)$pred
```

The argument `newdata=data` specifies that we want to predict the outcome using as input the observed exposure and covariates. We have created a vector `SL.predict.outcome`, which contains initial estimates of the expected outcome, given the observed exposure and covariates $\hat{\mathbb{E}}(Y|A,W)$.

```
> head(SL.predict.outcome)
>
```

(b.) Also obtain the initial estimates of the expected outcome for all units under the treatment/exposure $\hat{\mathbb{E}}(Y|A=1,W)$. Now we specify `newdata=data.txt` to predict the outcome using as input `data.txt`, where $A=1$ for all units.

```
> SL.predict.outcome.txt<- predict(SL.outcome.regression, newdata=data.txt)$pred
> head(SL.predict.outcome.txt)
>
```

(c.) Finally, obtain the initial estimates of the expected outcome for all units under no treatment/exposure $\hat{\mathbb{E}}(Y|A=0,W)$. Now we specify `newdata=data.untxt` to predict the outcome using as input `data.untxt`, where $A=0$ for all units.

```
> SL.predict.outcome.untxt<- predict(SL.outcome.regression, newdata=data.untxt)$pred
> head(SL.predict.outcome.untxt)
>
```

- **Step 3: Estimate the propensity score $\mathbb{P}(A = 1|W)$ with Super Learner**.

  Now we want to use Super Learner to flexibly estimate the conditional probability of being exposed, given the measured covariates.

  (a.) As input to the `SuperLearner` algorithm, we need to specify the outcome-for-prediction as well as the predictors. In this case, we are interested in predicting the exposure $A$ with the measured covariates $W$. Therefore, in the following, we denote the "outcome" with `Y=data$A` and the predictors as the baseline covariates with `X=subset(data, select=-c(A,Y))`. Here, the `subset` function is excluding both the exposure and outcome $(A, Y)$ from `data` and retaining (selecting) the covariates $W$.

```
> SL.pscore<- SuperLearner(Y=data$A, X=subset(data, select=-c(A,Y)),
                           SL.library=SL.library, family="binomial")
```

  For simplicity, we are using the same library, although this is not a requirement. We are telling `SuperLearner` that the "outcome" (here, the exposure) is binary (`family="binomial"`). We are calling the resulting object `SL.pscore`.

```
> # We can examine the output by typing
> SL.pscore
>
```

- **Step 4: Using the Super Learner fit, create the clever covariate $\hat{H}(A, W)$:**

$$\hat{H}(A, W) = \frac{\mathbb{I}(A = 1)}{\hat{\mathbb{P}}(A = 1|W)} - \frac{\mathbb{I}(A = 0)}{\hat{\mathbb{P}}(A = 0|W)}.$$

  The clever covariate for exposed units is 1 over the predicted probability of being treated/exposed, and the clever covariate for unexposed units is -1 over the predicted probability of not being treated/exposed.

  (a.) We can access the estimated propensity score $\hat{\mathbb{P}}(A_i = 1|W_i)$ with

```
> SL.predict.prob.txt<- SL.pscore$SL.predict
> # Note this is equivalent to
> # predict(SL.pscore, newdata=data)$pred
> summary(SL.predict.prob.txt)
> # In practice, we may need to bound the predicted propensity scores away from (0,1).
>
```

  (b.) Likewise, we can calculate the predicted probability of not being treated/exposed, given the baseline covariates.

```
> SL.predict.prob.untxt<- 1- SL.predict.prob.txt
```

  (c.) Use these estimates to create the clever covariate:

$$\hat{H}(A, W) = \left( \frac{\mathbb{I}(A = 1)}{\hat{\mathbb{P}}(A = 1|W)} - \frac{\mathbb{I}(A = 0)}{\hat{\mathbb{P}}(A = 0|W)} \right).$$

```
> H.AW<- as.numeric(data$A==1)/SL.predict.prob.txt -
       as.numeric(data$A==0)/SL.predict.prob.untxt
> summary(H.AW)
>
```

  (d.) Also evaluate the clever covariate at $A = 1$ and $A = 0$ for all units:

$$\hat{H}(1, W) = \frac{1}{\hat{\mathbb{P}}(A = 1|W)} \quad \text{and} \quad \hat{H}(0, W) = \frac{-1}{\hat{\mathbb{P}}(A = 0|W)}$$

  Call the resulting values `H.1W` and `H.0W`, respectively.

```
> H.1W<- 1/SL.predict.prob.txt
> H.0W<- -1/SL.predict.prob.untxt
```

```
> # We have created 3 new vectors H.AW, H.1W, H.0W.
> # viewing the last six observations in each vector along with the exposure status:
> tail(data.frame(data$A, H.AW, H.1W, H.0W))
>
```

- **Step 5. Target the initial estimator of the conditional mean outcome $\hat{\mathbb{E}}(Y|A,W)$ with information in the estimated propensity score $\hat{\mathbb{P}}(A=1|W)$.**

(a.) Run a univariate regression of the outcome $Y$ on the clever covariate $\hat{H}(A,W)$ with the initial estimator as offset. Specifically for a binary or bounded continuous outcome, we estimate the coefficient $\epsilon$ by fitting the following logistic regression model

$$logit\big[\hat{\mathbb{E}}^*(Y|A,W)\big] = logit\big[\hat{\mathbb{E}}(Y|A,W)\big] + \epsilon\hat{H}(A,W).$$

Note there is no intercept (i.e. there is no $\beta_0$ term), and the coefficient on the ($logit$) of the initial estimator is set to 1.

```
> logitUpdate<- glm(data$Y ~ -1 +offset(qlogis(SL.predict.outcome)) + H.AW,
                    family='binomial')
```

- We are again calling the `glm` function to fit a generalized linear model.
- On the left hand side of the formula, we have the outcome $Y$.
- On the right hand side of the formula, we suppress the intercept by including -1 and use as `offset` the $logit$ of our initial Super Learner estimates `SL.predict.outcome`. In `R`, $logit(x) = log(x/(1-x))$ function is given by `qlogis(x)`.
- The only main term in the regression is the clever covariate $\hat{H}(A,W)$.
- Including `family='binomial'` runs logistic regression.
- Again ignore any warning message.

```
> # we can examine the output by typing
> summary(logitUpdate)
>
```

(b.) Let `epsilon` denote the resulting maximum likelihood estimate of the coefficient on the clever covariate `H.AW`.

```
> epsilon<- logitUpdate$coef
> epsilon
```

(c.) Plug-in the estimated coefficient $\hat{\epsilon}$ to yield **targeted** estimates of the expected outcome under the exposure $\hat{\mathbb{E}}^*(Y|A=1,W)$ and under no exposure $\hat{\mathbb{E}}^*(Y|A=0,W)$:

$$\hat{\mathbb{E}}^*(Y|A=1,W) = logit^{-1}\Big[logit\big[\hat{\mathbb{E}}(Y|A=1,W)\big] + \hat{\epsilon}\hat{H}(1,W)\Big]$$
$$\hat{\mathbb{E}}^*(Y|A=0,W) = logit^{-1}\Big[logit\big[\hat{\mathbb{E}}(Y|A=0,W)\big] + \hat{\epsilon}\hat{H}(0,W)\Big]$$

where $logit^{-1}$ is the inverse-$logit$, $\hat{H}(1,W)$ is the clever covariate evaluated for all units under the exposure, and $\hat{H}(0,W)$ is the clever covariate evaluated for all units under no exposure.

```
> target.predict.outcome.txt<- plogis( qlogis(SL.predict.outcome.txt)+ epsilon*H.1W)
> target.predict.outcome.untxt<- plogis( qlogis(SL.predict.outcome.untxt)+ epsilon*H.0W)
>
```

In `R`, the inverse-$logit$ function is given by `plogis(x)`.

- **Step 6. Estimate the statistical parameter by substituting the targeted predictions into the G-Computation formula.**

Estimate $\Psi(\mathbb{P})$ by averaging the difference in the targeted predictions:

$$\Psi_{TMLE}(\hat{\mathbb{P}}) = \frac{1}{n}\sum_{i=1}^{n}\Big[\hat{\mathbb{E}}^*(Y_i|A_i=1,W_i) - \hat{\mathbb{E}}^*(Y_i|A_i=0,W_i)\Big]$$

```
> TMLE<- mean(target.predict.outcome.txt- target.predict.outcome.untxt)
> TMLE

> # we can also estimate E[E(Y|A=1,W)]
> mean(target.predict.outcome.txt)
> # and estimate E[E(Y|A=0,W)]
> mean(target.predict.outcome.untxt)
```

- Compare the point estimates from the 3 methods:

  ```
  > c(Simple.Subs, TMLE)
  ```

  The true value is -2.14%.

- TMLE is double robust; it will be consistent if *either* the conditional mean outcome $\mathbb{E}(Y|A,W)$ or the propensity score $\mathbb{P}(A=1|W)$ is estimated consistently.

- If both $\mathbb{E}(Y|A,W)$ and $\mathbb{P}(A=1|W)$ are estimated consistently (at a fast enough rate), the TMLE will be efficient and achieve the lowest possible asymptotic variance over a large class of (semi-parametric) estimators.

- These asymptotic properties describe what happens when sample size goes to infinity and also typically translate into lower bias and variance in finite samples.

- *Note:* If you call `SuperLearner` multiple times, you might get slightly different output. This is due, in part, to working with a limited library of algorithms in which no one algorithm dominates in performance. Furthermore, during the cross-validation step, the observations are randomly allocated to folds and with a relatively small sample size ($n = 100$), this random allocation could impact the weight distribution (`Coef`) for the algorithms.

# 4 The basics of the `ltmle` package

**The `ltmle` package (Lendle et al., 2017) expands the previous `tmle` package (Gruber and van der Laan, 2012). Specifically `ltmle` estimates parameters corresponding to point-treatment exposures, longitudinal exposures, marginal structural working models, dynamic treatment regimes and much more!**

- **Step 1. Load the `SuperLearner` and `ltmle` packages.**

  ```
  > library('SuperLearner')
  > library('ltmle')
  > # we can learn a lot more about the function by reading the help file
  > ?ltmle
  ```

  - The basic input to the function is the data set `data`, the exposure variable(s) `Anodes`, the outcome(s) `Ynodes`, and the exposure levels of interest `abar`.
  - The user can also specify censoring variables `Cnodes`, time-dependent covariates `Lnodes`, weights `observation.weights`, and the independent unit `id`. (See the help file for more information.)
  - Initial estimates of the conditional mean outcome $\mathbb{E}(Y|A,W)$ can be estimated according to a user-specified regression formula `Qform` or estimated with Super Learner `SL.library`.
  - Initial estimates of the propensity score $\mathbb{P}(A=1|W)$ can be estimated according to a user-specified regression formula `gform` or estimated with Super Learner `SL.library`.

- **Step 2. Call the `ltmle` function using Super Learner to estimate the conditional mean outcome $\mathbb{E}(Y|A,W)$ and the propensity score $\mathbb{P}(A=1|W)$.**

```
> ltmle.package<- ltmle(data=data, Anodes='A', Ynodes='Y', abar=list(1,0),
                        SL.library=SL.library)
```

Here, `abar=list(1,0)` specifies the comparison of interest: all exposed ($A = 1$) vs. all unexposed ($A = 0$).

- **Step 3. Use the `summary` function to obtain point estimates and get inference.**

```
> summary(ltmle.package)
```

# 5 Bonus: `drtmle` package

- The `drtmle` package is an alternative and can handle categorical exposures (and provides valid statistical inference for IPW with Super Learner) (Benkeser, 2018).
- For more information and an example with multi-level exposures, check out: https://benkeser.github.io/drtmle/articles/using_drtmle.html#multi-level-treatments

# References


L. Balzer, M. Petersen, and M.J. van der Laan. Tutorial for causal inference. In P. Buhlmann, P. Drineas, M. Kane, and M. van der Laan, editors, *Handbook of Big Data*. Chapman & Hall/CRC, 2016.

David Benkeser. *drtmle: Doubly-Robust Nonparametric Estimation and Inference*, 2018. URL https://CRAN.R-project.org/package=drtmle. R package version 1.0.3.

S. Gruber and M.J. van der Laan. tmle: An R package for targeted maximum likelihood estimation. *Journal of Statistical Software*, 51(13):1–35, 2012. doi: 10.18637/jss.v051.i13.

T. Hastie. *gam: Generalized Additive Models*, 2016. URL http://CRAN.R-project.org/package=gam. R package version 1.14.

T.J. Hastie and R.J. Tibshirani. *Generalized additive models*. Chapman & Hall, Boca Raton, 1990.

S.D. Lendle, J. Schwab, M.L. Petersen, and M.J. van der Laan. ltmle: An R package implementing targeted minimum loss-based estimation for longitudinal data. *Journal of Statistical Software*, 81(1):1–21, 2017.

M.L. Petersen and L.B. Balzer. Introduction to Causal Inference, 2014. URL www.ucbbiostat.com.

M.L. Petersen and M.J. van der Laan. Causal models and learning from data: Integrating causal modeling and statistical estimation. *Epidemiology*, 25(3):418–426, 2014.

M.L. Petersen, K.E. Porter, S. Gruber, Y. Wang, and M.J. van der Laan. Diagnosing and responding to violations in the positivity assumption. *Statistical Methods in Medical Research*, 21(1):31–54, 2012. doi: 10.1177/0962280210386207.

E. Polley, E. LeDell, C. Kennedy, and M. van der Laan. *SuperLearner: Super Learner Prediction*, 2018. URL http://CRAN.R-project.org/package=SuperLearner. R package version 2.0-24.

J.M. Robins. A new approach to causal inference in mortality studies with sustained exposure periods–application to control of the healthy worker survivor effect. *Mathematical Modelling*, 7:1393–1512, 1986. doi: 10.1016/0270-0255(86)90088-6.