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.

Solution: This is the answer key.

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:

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

  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 $n \to \infty$.

Solution:

```
> #---------------------------------------------------------------------
> # 1.  Estimate the conditional mean outcome Y given the exposure A and covariates W
> # according to main terms logistic regression
> #---------------------------------------------------------------------
> outcome.regression<- glm(Y~A+W1+W2+W3+W4, family='binomial', data=data)
> # We can see the output of regression fit with the summary command
> summary(outcome.regression)
```

```
Call:
glm(formula = Y ~ A + W1 + W2 + W3 + W4, family = "binomial",
    data = data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.57758  -0.18757  -0.02634   0.12886   0.42717

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.56997    1.10532  -2.325   0.0201 *
A           -0.91140    1.10506  -0.825   0.4095
W1          -0.43300    0.85257  -0.508   0.6115
W2           0.54557    1.49736   0.364   0.7156
W3          -0.04968    0.84444  -0.059   0.9531
W4          -0.21740    0.85189  -0.255   0.7986
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5.6447  on 99  degrees of freedom
Residual deviance: 4.1661  on 94  degrees of freedom
AIC: 25.473

Number of Fisher Scoring iterations: 6

> #---------------------------------------------------------------------
> #2. Using the fitted regression, obtain the
> # predicted outcomes given the set exposure and measured covariates
> #---------------------------------------------------------------------
>
> # 2a. Copy the original dataset data into two new data frames and set the
> # exposure level
> 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
> # the exposure A has been set 1 for all observations in data.txt
> colMeans(data.txt)

         W1          W2          W3          W4           A           Y
 0.54000000  0.54204879 -0.09806121 -0.01002969  1.00000000  0.06467146

> # the exposure A has been set 0 for all observations in data.untxt
> colMeans(data.untxt)

         W1          W2          W3          W4           A           Y
 0.54000000  0.54204879 -0.09806121 -0.01002969  0.00000000  0.06467146

> #---------------------------------------------------------------------
> # 2b. predicted outcome for each observation under treatment/exposure
```

```
> # and given its covariates
> #---------------------------------------------------------------------
> predict.outcome.txt<- predict(outcome.regression, newdata=data.txt, type='response')
> head(predict.outcome.txt)


        1         2         3         4         5         6
0.03104416 0.03498880 0.03262777 0.03244616 0.03703749 0.06480627


> #-----------------------------------------------
> # 2c. predicted outcome for each observation under no exposure (untreated)
> # and given its covariates
> #-----------------------------------------------
> predict.outcome.untxt<- predict(outcome.regression, newdata=data.untxt, type='response')
> head(predict.outcome.untxt)


        1         2         3         4         5         6
0.07382238 0.08273851 0.07741384 0.07700279 0.08733006 0.14704764


> #---------------------------------------------------------------------
> # 3. Obtain a point estimate by averaging the predicted outcomes over the
> # confounder distribution
> #---------------------------------------------------------------------
> # estimate of E[E(Y|A=1,W)]
> mean(predict.outcome.txt)

[1] 0.03372176


> # estimate of E[E(Y|A=0,W)]
> mean(predict.outcome.untxt)

[1] 0.07946353


> # our point estimate of Psi(P)
> Simple.Subs <- mean(predict.outcome.txt - predict.outcome.untxt)
> Simple.Subs

[1] -0.04574178
```

The point estimate based on the simple substitution estimator (i.e. parametric G-computation) is -4.57%. The true value is $\Psi(\mathbb{P}) = -2.14\%$.

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 ϵ 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 β_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:

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

```
> 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.

Solution:

```
> #=============================================================================
> # 1. Estimate the conditional mean outcome E(Y|A,W) with Super Learner
> #=============================================================================
> #
> # 1a. load the Super Learner package
> library("SuperLearner")


> # 1b. Specify the library of candidate algorithms
> SL.library<- c("SL.glm", "SL.glm.interaction", "SL.gam")


> # 1c. call Super Learner to estimated the conditional mean outcome E(Y|A,W)
> SL.outcome.regression<- SuperLearner(Y=data$Y,
                                       X=subset(data, select=-Y),
                                       SL.library=SL.library, family="binomial")
> SL.outcome.regression
```

```
Call:
SuperLearner(Y = data$Y, X = subset(data, select = -Y), family = "binomial",
    SL.library = SL.library)


                             Risk      Coef
SL.glm_All             0.002716358 0.0000000
SL.glm.interaction_All 0.001244653 0.2942867
SL.gam_All             0.001037665 0.7057133
```

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. The algorithm with the lowest average cross-validated risk estimate was generalized additive models (`SL.gam`). It was also given the most weight when building the best combination of prediction algorithms. The weight given to logistic regression with all interactions (`SL.glm.interaction`) is not trivial.

```
> #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # Side-Note on Super Learner:
```

Running Super Learner multiple times may result in slightly different risk estimates and weights. The package assigns the folds randomly (i.e the first 10 observations are not always set to Fold 1, the second 10 to Fold 2, and so forth). To improve the stability, we could try increasing the number of folds. We could also improve the performance of Super Learner by adding more algorithms. The following code expands the library and increases the number of folds to `V=20`:

```
> # Let's expand the library to  include stepwise logistic regression with and
> # without interactions and neural nets
> SL.library.expanded<- c("SL.glm", "SL.glm.interaction", "SL.gam",
                          "SL.step", "SL.step.interaction", "SL.nnet")
> SuperLearner(Y=data$Y, X=subset(data, select=-Y), SL.library=SL.library.expanded,
              family="binomial", cvControl=list(V=20))
> #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> #
> # Now back to TMLE
> #


> #=============================================================================
> # 2. Using the Super Learner fit, obtain
> # the predicted outcomes under the observed exposure,
> # under the exposure, and under no exposure
> #=============================================================================
> # 2a. predicted outcome given the observed exposure and measured covariates
> SL.predict.outcome<- predict(SL.outcome.regression, newdata=data)$pred
> #
> # viewing the initial predictions of the outcome given the observed exposure &
> # and covariates for the first 6 units
> head(SL.predict.outcome)


            [,1]
[1,] 0.068670239
[2,] 0.003057562
```

```
[3,] 0.014942617
[4,] 0.006741282
[5,] 0.003170273
[6,] 0.013573947


> #
> # 2b. predicted outcome given the exposure and measured covariates
> SL.predict.outcome.txt<- predict(SL.outcome.regression, newdata=data.txt)$pred
> head(SL.predict.outcome.txt)


            [,1]
[1,] 0.068670239
[2,] 0.003057562
[3,] 0.014942617
[4,] 0.006741282
[5,] 0.003170273
[6,] 0.013573947


> # In this particular case, the first 6 predictions under the observed exposure
> # are the same as the first 6 predictions under the treatment. This is only because the
> # first 6 observations receive the exposure (i.e. have A=1).


> # 2c. predicted outcome given no exposure and measured covariates
> SL.predict.outcome.untxt<- predict(SL.outcome.regression, newdata=data.untxt)$pred
> head(SL.predict.outcome.untxt)


            [,1]
[1,] 0.100696934
[2,] 0.005257821
[3,] 0.023233247
[4,] 0.011626204
[5,] 0.005447676
[6,] 0.023795039


> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # Side note: the substitution estimator based on our Super Learner fit would be
> mean(SL.predict.outcome.txt - SL.predict.outcome.untxt)


[1] -0.02527743
```

As discussed in the slides, Super Learner's job is to obtain the best prediction function for the outcome Y using as input the exposure and covariates (A, W). This is a much different (and harder) job than estimating our target parameter $\Psi(\mathbb{P}) = \mathbb{E}\big[\mathbb{E}(Y|A=1,W) - \mathbb{E}(Y|A=0,W)\big]$, which is just one number. As a result, our Super Learner-based estimates of the conditional mean outcome $\mathbb{E}(Y|A,W)$ will have the wrong bias-variance trade-off; specifically, the estimator will have too much bias relative to its variance. As a direct result, there is no reliable way to obtain statistical inference for a (non-targeted) substitution estimator, based on Super Learner initial estimates.

```
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
> #============================================================================
> # 3.  Estimate the propensity score P(A=1|W) with Super Learner
> #============================================================================
> # call Super Learner for estimation of the pscore
> SL.pscore<- SuperLearner(Y=data$A, X=subset(data, select= -c(A,Y)),
                           SL.library=SL.library, family="binomial")
> SL.pscore


Call:
SuperLearner(Y = data$A, X = subset(data, select = -c(A, Y)), family = "binomial",
    SL.library = SL.library)


                             Risk      Coef
SL.glm_All              0.2289346 0.0000000
SL.glm.interaction_All 0.2208972 0.1866247
SL.gam_All              0.2091882 0.8133753
```

The algorithm with the lowest average cross-validated risk estimate was again generalized additive models (`SL.gam`). It was also given the most weight when building the best combination of prediction algorithms. The weight given to main terms logistic regression with interactions (`SL.glm.interaction`) is notable.

As before, running Super Learner multiple times may result in slightly different risk estimates and weights. Again, we can improve the performance of Super Learner by expanding the library and by increasing the number of folds.

```
> #============================================================
> # 4. Using the Super Learner fit, create the clever covariate
> #============================================================
> # 4a. generate the predicted prob of being exposed, given baseline covariates
> SL.predict.prob.txt<- SL.pscore$SL.predict
> # this is equivalent to
> SL.predict.prob.txt2<- predict(SL.pscore, newdata=data)$pred
> sum(SL.predict.prob.txt !=SL.predict.prob.txt2)

[1] 0

> # can again examine the distribution of estimated propensity scores
> summary(SL.predict.prob.txt)

       V1
 Min.   :0.1193
 1st Qu.:0.1848
 Median :0.2717
 Mean   :0.3200
 3rd Qu.:0.4159
 Max.   :0.8834

> # 4b. generate the predicted prob of not being exposed, given baseline covariates
> SL.predict.prob.untxt<- 1- SL.predict.prob.txt
```

```
> # 4c. Create the clever covariate H(A,W) for each unit
> H.AW<- as.numeric(data$A==1)/SL.predict.prob.txt -
   as.numeric(data$A==0)/SL.predict.prob.untxt
> summary(H.AW)

       V1
 Min.   :-3.35632
 1st Qu.:-1.34276
 Median :-1.20460
 Mean   :-0.06693
 3rd Qu.: 1.65755
 Max.   : 5.94661

> # 2e. also want to evaluate the clever covariates at A=1 and A=0 for all units
> H.1W<- 1/SL.predict.prob.txt
> H.0W<- -1/SL.predict.prob.untxt

> # checking
> tail(data.frame(data$A, H.AW, H.1W, H.0W))

    data.A      H.AW     H.1W      H.0W
95       0 -1.194149 6.150670 -1.194149
96       0 -1.228788 5.370863 -1.228788
97       0 -1.219359 5.558743 -1.219359
98       1  2.206388 2.206388 -1.828921
99       0 -2.410726 1.708855 -2.410726
100      0 -1.183290 6.455835 -1.183290

> #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # side note: the IPTW estimator based on the Super Learner estimates of the pscore
> # note the relation between the clever covariate and the IPTW weights :)
> mean( H.AW*data$Y)

[1] -0.02757504
```

In this case, Super Learner was only concerned doing the best job predicting the exposure A given the measured covariates W. As a result, an IPTW using Super Learner for initial estimation of the propensity score will have the wrong bias-variance trade-off for the parameter of interest. Again, there is no reliable way to obtain statistical inference (i.e. create confidence intervals and test hypotheses).

```
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

> #=========================================================================
> # 5. Target the initial estimator of the conditional mean outcome E(Y|A,W)
> # with information in the estimated propensity score P(A=1|W)
> #=========================================================================
> # 5a. run logistic regression of Y on H.AW using the logit of initial estimator
> # as offset
> logitUpdate<- glm(data$Y~ -1 +offset(qlogis(SL.predict.outcome)) + H.AW,
                    family='binomial')
> summary(logitUpdate)
```

```
Call:
glm(formula = data$Y ~ -1 + offset(qlogis(SL.predict.outcome)) +
    H.AW, family = "binomial")

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.26072  -0.09934  -0.03551   0.07072   0.24617

Coefficients:
     Estimate Std. Error z value Pr(>|z|)
H.AW  0.01253    0.19833   0.063     0.95

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.2727  on 100  degrees of freedom
Residual deviance: 1.2688  on  99  degrees of freedom
AIC: 15.514

Number of Fisher Scoring iterations: 5


> # 5b. Denote the estimated coefficient (i.e. the fluctuation coefficient) as epsilon
> epsilon<- logitUpdate$coef
> epsilon


      H.AW
0.01252757


> #  5c. update the initial estimators
> 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)


> #================================================================================
> # 6. Estimate Psi(P) as the empirical mean of the difference in the predicted
> # outcomes under A=1 and A=0
> #================================================================================
> TMLE<- mean(target.predict.outcome.txt- target.predict.outcome.untxt)
> TMLE


[1] -0.02154402


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


[1] 0.04887533


> # and estimate E[E(Y|A=0,W)]
> mean(target.predict.outcome.untxt)


[1] 0.07041936
```

Note: if our goal is to estimate each treatment-specific mean, we recommend using a two-dimensional clever covariate, as is implemented in the `ltmle` package.

```
> # comparing the estimates (in percent)
> c(Simple.Subs, TMLE)*100

[1] -4.574178 -2.154402
```

The point estimate from the simple substitution estimator (i.e. parametric G-comp.) was -4.57%. The point estimate from IPTW was -4.50%; see Appendix A. The point estimate from TMLE was -2.15%. The true value of the statistical estimand was -2.14%. To evaluate the performance of these estimators (e.g. bias, variance and mean squared error), we would draw another independent sample of size n, implement the 3 estimators, and repeat 500 or so times. See Appendix B.

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)
```

Solution:

```
> #====================
> # Re-loading the observed data for the workshop & resetting the seed
> data<- read.csv("CausalWorkshop.csv")
> set.seed(1)


> # Load the Super Learner & ltmle packages
> library("SuperLearner")
> library("ltmle")


> ?ltmle


> # call ltmle with Super Learner (same libraries)
> ltmle.package<- ltmle(data=data, Anodes='A', Ynodes='Y', abar=list(1,0),
                        SL.library=SL.library)


> summary(ltmle.package)


Estimator:  tmle
Call:
ltmle(data = data, Anodes = "A", Ynodes = "Y", abar = list(1,
    0), SL.library = SL.library)

Treatment Estimate:
   Parameter Estimate:  0.050476
    Estimated Std Err:  0.0055011
              p-value:  <2e-16
    95% Conf Interval: (0.039694, 0.061258)

Control Estimate:
   Parameter Estimate:  0.07113
    Estimated Std Err:  0.0062687
              p-value:  <2e-16
    95% Conf Interval: (0.058843, 0.083416)

Additive Treatment Effect:
   Parameter Estimate:  -0.020653
    Estimated Std Err:  0.0055602
              p-value:  0.00020361
    95% Conf Interval: (-0.031551, -0.0097555)
```

The `ltmle` package provides estimates and inference for the expected outcome under the exposure ("Treatment Estimate"), the expected outcome under no exposure ("Control Estimate"), and the Additive Treatment Effect. If the outcome is binary, the package will also return estimates of the risk ratio and odds ratio. (See Example1 in the help file.)

The point estimates from `ltmle` package might differ from our code for several reasons. First, the `ltmle` package uses a two-dimensional clever covariate in updating step. This allows us to obtain targeted estimates of the marginal risk under the exposure, the marginal risk under no exposure, the marginal risk difference, the marginal risk ratio and the marginal odds ratio (after controlling for measured confounders).

In our code, we used a one-dimensional clever covariate for simplicity and to focus on the marginal risk difference. Second, the `ltmle` package bounds the estimated propensity scores. This bounding is included to deal with theoretical and practical positivity violations. For further discussion, see Petersen et al. (2012). Finally, the Super Learner algorithm could split the data into different folds. (This is why we reset the seed.)

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

6 Statistical inference and interpretation of the results:

If we apply an estimator to our observed data (n i.i.d. copies of O drawn from $\mathbb{P}$), we get an estimate (a number). The estimator is function of random variables; so it is a random variable. It has a distribution, which we can study theoretically or using simulations.

We did not cover statistical inference in this workshop. Briefly, parametric G-computation, IPTW estimator and TMLE are, under reasonable regularity conditions, asymptotically linear. As a result, they are asymptotically normal with mean zero and variance given by the variance of their influence curve. Alternatively the non-parametric bootstrap can be used to obtain inference.

The strength of our interpretations depend on the whether and to what degree our identifiability assumptions have been met. If our needed assumptions do not hold, we also have a statistical interpretation as the difference in the expected outcome associated with all units being exposed compared to no units unexposed, after accounting for the measured confounders. If the needed assumptions (randomization+positivity) held, we could interpret our estimates as the average treatment effect (or the causal risk difference).

Appendix A: Inverse Probability of Treatment Weighting (IPTW) estimator

- **Step 1. Estimate the propensity score $\mathbb{P}(A = 1|W)$ with parametric regression.** First, we run a regression of the exposure A on the covariates W. This provides an estimate of the propensity score $\hat{\mathbb{P}}(A = 1|W)$. Again, we are using the "hat" notation to denote an estimate based on n observations. Suppose we are willing to assume the conditional probability of being exposed was accurately described by the following main terms function of the covariates:

$$logit\big[\mathbb{P}(A = 1|W)\big] = \beta_0 + \beta_1 W1 + \beta_2 W2 + \beta_3 W3 + \beta_4 W4.$$

  (a.) Use the `glm` function to fit the propensity score $\mathbb{P}(A = 1|W)$ with main terms parametric logistic regression.

```
> pscore.regression<- glm(A~W1+W2+W3+W4, family="binomial", data=data)
> summary(pscore.regression)
```

  As before, the argument `family="binomial"` specifies logistic regression, and the argument `data=data` specifies the data set for fitting the regression.

- **Step 2. Using the fitted propensity score regression, create the weights.**

Next, we obtain for all observations the predicted probability of being exposed, given the measured covariates: $\hat{\mathbb{P}}(A = 1|W)$. We can also calculate the predicted probability of not being exposed, given the measured covariates, as

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

Then, we obtain the weights as the inverse of the predicted probability of the *observed* exposure, given the baseline covariates.

(a.) Use the `predict` function to obtain for each observation i the predicted probability of being exposed, given the measured covariates $\hat{\mathbb{P}}(A_i = 1|W_i)$.

```
> predict.prob.txt <- predict(pscore.regression, type= "response")
```

Again, we need the argument `type="response"` to return the predictions on the relevant scale (probabilities - not log-odds).

We have created a vector `predict.prob.txt`, which contains the estimated propensity scores. We can create a summary and a histogram with

```
> summary(predict.prob.txt)
> hist(predict.prob.txt)
>
```

(b.) For each observation i, obtain the predicted probability of not being exposed, given the baseline covariates:

$$\hat{\mathbb{P}}(A_i = 0|W) = 1 - \hat{\mathbb{P}}(A_i = 1|W_i)$$

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

We have created a vector `predict.prob.untxt`, which contains the estimated probability of not being exposed, given the covariates. As before, we can create a summary and a histogram with

```
> summary(predict.prob.untxt)
> hist(predict.prob.untxt)
>
```

(c.) Create the weights. The weight for exposed units is 1 over the predicted probability of being exposed (treated), while the weight for unexposed units is 1 over the predicted probability of being unexposed (untreated). For observation i, we can write the weight as

$$wt_i = \frac{\mathbb{I}(A_i = 1)}{\hat{\mathbb{P}}(A_i = 1|W_i)} + \frac{\mathbb{I}(A_i = 0)}{\hat{\mathbb{P}}(A_i = 0|W_i)}$$

where $\mathbb{I}(A_i = a)$ is an indicator function, equaling 1 if unit i has exposure status a and equaling 0 otherwise.

- Specifically, if observation i was exposed (treated), the first term evaluates to the inverse-estimated propensity score and the second term evaluates to zero: $1/\hat{\mathbb{P}}(A_i = 1|W_i) + 0$.

- Likewise, if observation i was not exposed (untreated), the first term evaluates to zero and the second term evaluates to the inverse-predicted probability of not being exposed: $0 + 1/\hat{\mathbb{P}}(A_i = 0|W_i)$.

```
> wt <- as.numeric(data$A==1)/predict.prob.txt +
                  as.numeric(data$A==0)/predict.prob.untxt
```

- We are coding indicator functions with the `as.numeric` function, applied to a logical statement. We are evaluating whether unit i was exposed $A_i \stackrel{?}{=} 1$ vs. not exposed $A_i \stackrel{?}{=} 0$. We are accessing the exposure column A of the `data` data frame with the $ operator.

- We can check to make sure we have inverted the proper probabilities and create a summary of the weights.

```
> tail(data.frame(data$A, 1/predict.prob.txt, 1/predict.prob.untxt, wt))
> summary(wt)
>
```

- **Step 3. Estimate the statistical parameter by averaging the weighted outcomes.** Finally, we estimate $\Psi(\mathbb{P})$ by taking the difference in the sample average of the weighted outcomes:

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

```
> # estimate of E[E(Y|A=1,W)]
> mean(as.numeric(data$A==1)*wt*data$Y)
> # estimate of E[E(Y|A=0,W)]
> mean(as.numeric(data$A==0)*wt*data$Y)
> # our point estimate for Psi(P)
> IPTW<- mean(as.numeric(data$A==1)*wt*data$Y) -
        mean(as.numeric(data$A==0)*wt*data$Y)
> IPTW
>
```

- Consistency of IPTW depends on consistent estimation of the propensity score: $\mathbb{P}(A=1|W)$. If the *a priori* specified regression model is incorrect, our point estimates can be biased and inference misleading.
- Please note there are lots of different IPTW estimators, and we are just focusing on one type.
- We could simplify the code by skipping the creation of the weights:

$$\Psi_{IPTW}(\hat{\mathbb{P}}) = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\left(\frac{\mathbb{I}(A_i=1)}{\hat{\mathbb{P}}(A_i=1|W_i)} - \frac{\mathbb{I}(A_i=0)}{\hat{\mathbb{P}}(A_i=0|W_i)}\right)}_{\text{clever covariate in TMLE}}Y_i$$

```
> mean( (as.numeric(data$A==1)/predict.prob.txt -
          as.numeric(data$A==0)/predict.prob.untxt)  *data$Y)
>
```

Solution:

```
> #---------------------------------------------------------------------
> # 1.  Estimate the propensity score P(A=1|W) with main terms logistic regression
> #---------------------------------------------------------------------
> pscore.regression<- glm(A~W1+W2+W3+W4, family="binomial", data=data)
> summary(pscore.regression)


Call:
glm(formula = A ~ W1 + W2 + W3 + W4, family = "binomial", data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3603  -0.8825  -0.7527   1.2876   2.1745

Coefficients:
```

```
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.8459     0.5697  -1.485    0.138
W1            0.3441     0.4542   0.758    0.449
W2           -0.3435     0.7705  -0.446    0.656
W3           -0.6663     0.4654  -1.432    0.152
W4            0.4185     0.4595   0.911    0.362

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 125.37  on 99  degrees of freedom
Residual deviance: 121.70  on 95  degrees of freedom
AIC: 131.7

Number of Fisher Scoring iterations: 4


> #=====================================================================
> # 2. Using the fitted regression, create the weights
> #=====================================================================#
> #
> #---------------------------------------------------------------------
> # 2a. predicted probability of being exposed, given the observed covariates P(A=1|W)
> #---------------------------------------------------------------------
> predict.prob.txt <- predict(pscore.regression, type= "response")
> # can summarize the estimated propensity scores
> summary(predict.prob.txt)


   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.09401 0.24993 0.31715 0.32000 0.37047 0.60355


> # can also look at their distribution with a histogram
> hist(predict.prob.txt)


> #---------------------------------------------------------------------
> # 2b. predicted probability of not being exposed, given the observed covariates
> # P(A=0|W)
> #---------------------------------------------------------------------
> predict.prob.untxt <- 1 - predict.prob.txt
> # viewing a histogram of 1 - the estimated pscore
> hist(predict.prob.untxt)


> # We can aslo look at the distribution of estimated propensity scores by exposure status.
> # distribution among exposed units
> hist(predict.prob.txt[data$A==1])
> # distribution among unpexposed units
> hist(predict.prob.txt[data$A==0])


> #---------------------------------------------------------------------
> # 2c. Calculate the weights
> #---------------------------------------------------------------------
> #
> wt <- as.numeric(data$A==1)/predict.prob.txt +
```

```
   as.numeric(data$A==0)/predict.prob.untxt
> # double-checking our code is good
> # for obs with A=1, the wt should equal 1/predict.prob.txt
> # for obs with A=0, the wt should equal 1/predict.prob.untxt
> tail(data.frame(data$A, 1/predict.prob.txt, 1/predict.prob.untxt, wt))

    data.A X1.predict.prob.txt X1.predict.prob.untxt       wt
95       0            4.090605              1.323561 1.323561
96       0            3.496448              1.400569 1.400569
97       0            4.034962              1.329493 1.329493
98       1            2.982709              1.504360 2.982709
99       0            2.304259              1.766719 1.766719
100      0            4.052189              1.327634 1.327634

> summary(wt)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.239   1.370   1.566   2.042   2.302  10.637

> #---------------------------------------------------------------------
> # 3. Point estimate:
> #---------------------------------------------------------------------
> # estimate of E[E(Y|A=1,W)]
> mean(as.numeric(data$A==1)*wt*data$Y)

[1] 0.03310559

> # estimate of E[E(Y|A=0,W)]
> mean(as.numeric(data$A==0)*wt*data$Y)

[1] 0.07811292

> # our point estimate for Psi(P)
> IPTW<- mean(as.numeric(data$A==1)*wt*data$Y) -
       mean(as.numeric(data$A==0)*wt*data$Y)
> IPTW

[1] -0.04500732
```

The point estimate based on IPTW is -4.5%. The true value is $\Psi(\mathbb{P}) = -2.14\%$.

It is worth noting that we could further simplify the code by skipping the creation of the weights:

$$\Psi_{IPTW}(\hat{\mathbb{P}}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\left(\frac{\mathbb{I}(A_i=1)}{\hat{\mathbb{P}}(A_i=1|W_i)} - \frac{\mathbb{I}(A_i=0)}{\hat{\mathbb{P}}(A_i=0|W_i)}\right)}_{\text{clever covariate in TMLE}} Y_i$$

```
> mean( (as.numeric(data$A==1)/predict.prob.txt -
          as.numeric(data$A==0)/predict.prob.untxt)*data$Y)

[1] -0.04500732
```

Appendix B: Using simulations to evaluate the performance of candidate estimators

Simulations are useful for evaluating the properties of estimators. We will focus on estimating the bias, variance, mean squared error (MSE), and confidence interval coverage (proportion of times the 95% confidence intervals contain the true parameter value) of the 3 estimators. Specifically, for 2500 iterations, we will sample $n = 100$ i.i.d. observations from the true distribution $\mathbb{P}$, implement the 3 estimators, and save the resulting point estimates. We will do this process both when the intervention has an effect and when it does not.

```
> #-------------------------
> EFFECT<- T # specify whether or not there is an effect
> nReps<- 2500 # number of iterations
> sample.size<- 100
> # vectors to hold the point estimates from each sample
> Simple.Subs<- IPTW<- TMLE<- rep(NA, nReps)
> #
> # Same the library of candidate algorithms for Super Learner
> SL.library<- c("SL.glm", "SL.glm.interaction", "SL.gam")
> #
> for(r in 1:nReps){

   FullData<- generateData(n=sample.size, effect=EFFECT)
   # the generateData function is given below
   # the generateData function  returns the counterfactuals,
   # but the observed data only consist of covariates, exposure
   # and the observed outcome; so we must subset
   data<- subset(FullData, select=c(W1,W2,W3,W4, A, Y) )

   #----------------------
   # Simple Substituion - Parametric Gcomp.
   #----------------------
   outcome.regression<- glm(Y~A+W1+W2+W3+W4, family='binomial', data=data)
   data.txt<- data.untxt <- data
   data.txt$A <-1
   data.untxt$A <- 0
   predict.outcome.txt<- predict(outcome.regression, newdata=data.txt, type='response')
   predict.outcome.untxt<- predict(outcome.regression, newdata=data.untxt, type='response')
   # assigning the resulting point estimate to element r of vector 'Simple.Subs'
   Simple.Subs[r] <- mean(predict.outcome.txt - predict.outcome.untxt)

   #-------------
   #   IPTW (compact code)
   #-------------
   pscore.regression<- glm(A~W1+W2+W3+W4, family="binomial", data=data)
   predict.prob.txt <- predict(pscore.regression, type= "response")
   predict.prob.untxt <- 1 - predict.prob.txt
   # assigning the resulting point estimate to element r of vector 'IPTW'
   IPTW[r]<- mean( (as.numeric(data$A==1)/predict.prob.txt -
                    as.numeric(data$A==0)/predict.prob.untxt)* data$Y)

   #---------------
   # TMLE with Super Learner
   #--------------
   SL.outcome.regression<- SuperLearner(Y= data$Y, X=subset(data, select= -Y),
```

```
                                SL.library=SL.library, family="binomial")
   SL.predict.outcome<- predict(SL.outcome.regression, newdata=data)$pred
   SL.predict.outcome.txt<- predict(SL.outcome.regression, newdata=data.txt)$pred
   SL.predict.outcome.untxt<- predict(SL.outcome.regression, newdata=data.untxt)$pred
   #
   SL.pscore<- SuperLearner(Y=data$A, X=subset(data, select= -c(A,Y)),
                            SL.library=SL.library, family="binomial")
   SL.predict.prob.txt<- SL.pscore$SL.predict
   SL.predict.prob.untxt<- 1- SL.predict.prob.txt
   #
   H.AW<- as.numeric(data$A==1)/SL.predict.prob.txt -
    as.numeric(data$A==0)/SL.predict.prob.untxt
   H.1W<- 1/SL.predict.prob.txt
   H.0W<- -1/SL.predict.prob.untxt
   #
   logitUpdate<- glm(data$Y~ -1 +offset(qlogis(SL.predict.outcome)) + H.AW,
                     family='binomial')
   eps<- logitUpdate$coef

   target.predict.outcome.txt<- plogis(qlogis(SL.predict.outcome.txt)+ eps*H.1W)
   target.predict.outcome.untxt<- plogis(qlogis(SL.predict.outcome.untxt)+ eps*H.0W)
   # assigning the resulting point estimate to element r of vector 'TMLE'
   TMLE[r]<- mean(target.predict.outcome.txt- target.predict.outcome.untxt)

   # keep track of where we are.
   print(r)

 }
> #----------------------------
```

Let's write a quick function to evaluate estimator performance in terms of the average value, bias, variance. mean squared error (MSE) and confidence interval coverage.

```
> #----------------------------
> performance <- function(estimates, truth) {
   ave <- mean(estimates)
   bias <- mean(estimates - truth)
   var <- var(estimates)
   mse <- mean((estimates - truth) ^ 2)
   z <- bias / sqrt(var)
   cover <- pnorm(1.96 - z) - pnorm(-1.96 - z)
   data.frame(ave,bias,var, mse,cover) * 100 # Return in percent
 }
> #
> # Evaluating the performance when there is an effect Psi.F=-2.14% (See Appendix C)
> SimpleSubs.performance<- performance(estimates=Simple.Subs, truth=Psi.F)
> IPTW.performance<- performance(estimates=IPTW, truth=Psi.F)
> TMLE.performance<- performance(estimates=TMLE, truth=Psi.F)
> #---------------------

> #============================
> data.frame(rbind(SS=SimpleSubs.performance, IPTW=IPTW.performance, TMLE=TMLE.performance))

          ave        bias         var         mse    cover
SS   -3.624630 -1.48379390 0.010217057 0.032229413 68.83546
```

```
IPTW -3.430012 -1.28917588 0.009949849 0.026565613 74.72268
TMLE -2.182807 -0.04197159 0.004131514 0.004147478 94.95157
```

```
> #================================
```

Over the 2,500 simulated data sets, TMLE has the lowest bias (average deviation between the true value and the point estimate), variance, and mean squared error. We also see that TMLE achieves 95% coverage of the created confidence intervals. (For simplicity, these confidence intervals were constructed using the true as opposed to estimated variance.) The other estimators tend to overestimate the intervention effect. This bias is substantial enough to prevent accurate inferences as indicated by the coverage of the confidence intervals being much less than nominal.

```
> #====================
> # Repeating this exercise when there is not a treatment effect: Psi.F=0
> # We would use the same code, but just set EFFECT=F
> # Just giving the results here
> #======================
```

```
> rm(Simple.Subs, IPTW, TMLE)
> load("SERtalks_effectFALSE_nReps2500.Rdata")
```

```
> SimpleSubs.performance<- performance(estimates=Simple.Subs, truth=Psi.F)
> IPTW.performance<- performance(estimates=IPTW, truth=Psi.F)
> TMLE.performance<- performance(estimates=TMLE, truth=Psi.F)
```

```
> data.frame(rbind(SS=SimpleSubs.performance, IPTW=IPTW.performance, TMLE=TMLE.performance))
```

```
             ave        bias         var         mse    cover
SS   -1.92965774 -1.92965774 0.016552426 0.053781595 67.70235
IPTW -1.72418341 -1.72418341 0.016753137 0.046474520 73.44695
TMLE -0.05917859 -0.05917859 0.006798044 0.006830346 94.94139
```

Again over the 2,500 simulated data sets, the TMLE is the least bias, least variable and achieves nominal confidence interval coverage. The other estimators are biased.

Overall, consistency of the simple substitution estimator depends on consistent estimation of $\mathbb{E}(Y|A,W)$. Consistency of IPTW estimators depends on consistent estimation of $\mathbb{P}(A=1|W)$. TMLE is double robust! Even when the conditional mean function $\mathbb{E}(Y|A,W)$ is misspecified or $\mathbb{P}(A=1|W)$ is misspecified, we obtain a consistent estimate of $\Psi(\mathbb{P})$. If both $\mathbb{E}(Y|A,W)$ and $\mathbb{P}(A=1|W)$ are consistently estimated, then TMLE will achieve lowest asymptotic possible variance over a large class of (semi-parametric) estimators. Formally, an estimator is *consistent* if the point estimates converge (in probability) to the estimand as sample size $n \to \infty$. This is an asymptotic property. Here, we only have one sample of size $n = 100$. To evaluate the consistency, we would need to do multiple runs at increasing samples sizes, e.g. $n = 500$, $n = 1000$, $n = 2,500$, $n = 5,000$, $n = 50,000$, $n = 100,000$.

Appendix C: A specific data generating process

The following code was used to generate the data set `CausalWorkshop.csv`. In this data generating process (one of many compatible with the structural causal model), *there is no unmeasured confounding.*

```
> library('MASS')
> #-------
> # generateData - function to generate the data
> # input: number of draws, whether or not there is a treatment effect
> # output: observed data + counterfactuals
> #--------
> generateData<- function(n, effect){

   W1 <- rbinom(n, size=1, prob=0.5)
   W2<- runif(n, min=0, max=1)

   # W3 and W4 are drawn from a multivariate normal (i.e. correlated)
   s=1
   Sigma<- matrix(0.85*s*s, nrow=2, ncol=2)
   diag(Sigma)<- s^2
   Z<- mvrnorm(n, rep(0,2), Sigma)
   W3<- Z[,1]; W4<- Z[,2];

   # generate the propensity score P(A=1|W)
   pscore<- plogis(-1.25 - .1*(W1+W2) +.45*W3*W4)

   A<- rbinom(n, size=1, prob= pscore)

   U.Y<- rnorm(n, 0, s)

   # generate the counterfactual outcome with A=0
   Y.0<- generateY(W1=W1, W2=W2, W3=W3, W4=W4, A=0, U.Y=U.Y)

   if(!effect){ # if there is no effect, the counterfactual under txt =
     # the counterfactual under the control
     Y.1<- Y.0
   }else{ # otherwise, generated the counterfactual outcome with A=1
     Y.1<- generateY(W1=W1, W2=W2, W3=W3, W4=W4, A=1, U.Y=U.Y)
   }

   # assign the observed outcome based on the observed exposure
   Y<- rep(NA, n)
   Y[A==1]<- Y.1[A==1]
   Y[A==0]<- Y.0[A==0]

   data<- data.frame(W1, W2, W3, W4, A, Y, Y.1, Y.0)
   data
 }
> #---------
> # generateY: function to generate the outcome given the
> #   baseline covariates, exposure and background error U.Y
> #------------------
> generateY<- function(W1, W2, W3, W4, A, U.Y){
   W1*plogis(-2.5  +.5*W2 -1*W4 -0.5*A  -2*W4*W4 -.2*W4*A + .25*U.Y) +
     (1-W1)*plogis(-2  +.5*W2 -1*W3 -0.5*A  -2*W3*W3 -.2*W3*A + .25*U.Y)
 }


> #------------
> # Creation of CausalWorkshop.csv
> #--------------
```

```
> FullData<- generateData(n=100, effect= T)
> # remove unobservable counterfactuals
> ObsData<- subset(FullData, select=c(W1,W2,W3,W4, A, Y) )
> write.csv(ObsData, file="CausalWorkshop.csv")
> #------------
```

We obtained the true value of the causal parameter $\mathbb{E}(Y_1 - Y_0)$ by drawing a huge number of observations and taking the mean difference in the counterfactual outcomes.

```
> TrueData<- generateData(n=500000, effect=T)
> # Expected counterfactual outcome if all were exposed
> mean(TrueData$Y.1)

[1] 0.04190283

> # Expected counteractual outcome if none were exposed
> mean(TrueData$Y.0)

[1] 0.06329689

> # Simply take the mean difference in the counterfactuals
> Psi.F<- mean(TrueData$Y.1 - TrueData$Y.0)
> Psi.F

[1] -0.02139406
```

The average treatment effect $\mathbb{E}(Y_1 - Y_0)$ is -2.14%.

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.