Code

Migariane

Targeted Maximum Likelihood Estimation for a Binary Outcome: Tutorial and Guided Implementation

By: Miguel Angel Luque Fernandez

June 20th, 2017

1 Introduction

During the last 30 years, **modern epidemiology** has been able to identify significant limitations of classic epidemiologic methods when the focus is to explain the main effect of a risk factor on a disease or outcome.

Causal Inference based on the **Neyma-Rubin Potential Outcomes Framework** (Rubin, 2011), first introduced in Social Science by Donal Rubin (Rubin, 1974) and later in Epidemiology and Biostatistics by James Robins (Greenland and Robins, 1986), has provided the theory and statistical methods needed to overcome recurrent problems in observational epidemiologic research, such as:

1. non-collapsibility of the odds and hazard ratios,
2. impact of paradoxical effects due to conditioning on colliders,
3. selection bias related to the vague understanding of the effect of time on exposure and outcome and,
4. effect of time-dependent confounding and mediators,
5. etc.

Causal effects are often formulated regarding comparisons of potential outcomes, as formalised by Rubin (Rubin, 2011). Let A denote a binary exposure, **W** a vector of potential confounders, and Y a binary outcome. Given A, each individual has a pair of potential outcomes: the outcome when exposed, denoted Y_1, and the outcome when unexposed, Y_0. These quantities are referred to as **potential outcomes** since they are hypothetical, given that it is only possible to observe a single realisation of the outcome for an individual; we observe Y_1 only for those in the exposure group and Y_0 only for those in the unexposed group (Rubin, 1974). A common causal estimand is the **Average Treatment Effect** (ATE), defined as $E[Y_1 - Y_0]$.

Classical epidemiologic methods use regression adjustment to explain the main effect of a risk factor measure on a disease or outcome. Regression adjustment control for confounding but requires making the assumption that the effect measure is constant across levels of confounders included in the model. However, in non-randomized observational studies, the effect measure is not constant across groups given the different distribution of individual characteristics at baseline.

James Robins in 1986 demonstrated that using the **G-formula** a generalization of the **standardisation**, allows obtaining a unconfounded marginal estimation of the ATE under causal untestable assumptions, namely conditional mean independence, positivity and consistency or stable unit treatment value assignment (SUTVA) (Greenland and Robins, 1986), (Robins *et al.*, 2000):

2 The G-Formula and ATE estimation

$$\psi(P_0) = \sum_w \left[\sum_y P(Y = y \mid A = 1, W = w) - \sum_y P(Y = y \mid A = 0, W = w) \right] P(W = w)$$

where,

$$P(Y = y \mid A = a, W = w) = \frac{P(W = w, A = a, Y = y)}{\sum_y P(W = w, A = a, Y = y)}$$

is the conditional probability distribution of Y = y, given A = a, W = w and,

$$P(W = w) = \sum_{y,a} P(W = w, A = a, Y = y)$$

The ATE can be estimated **non-parametrically** using the G-formula. However, the **course of dimensionality** in observational studies limits its estimation. Hence, the estimation of the ATE using the G-formula relies mostly on **parametric modelling** and maximum likelihood estimation.

The correct model specification in parametric modelling is crucial to obtain unbiased estimates of the true ATE (Rubin, 2011). Alternatively, propensity score methods, introduced by Rosenbaum and Rubin (Rosenbaum and Rubin, 1983), are also commonly used for estimation of the ATE. The propensity score is a balancing score that can be used to create statistically equivalent exposure groups to estimate the ATE via matching, weighting, or stratification (Rosenbaum and Rubin, 1983).

However, very low or very high propensity scores can lead to very large weights, resulting in unstable ATE estimates with high variance and values outside the constraints of the statistical model (Lunceford and Davidian, 2004).

Furthermore, when analyzing observational data with a large number of variables and potentially complex relationships among them, model misspecification during estimation is of particular concern. Hence, the correct model specification in parametric modelling is crucial to obtain unbiased estimates of the true ATE (Laan and Rose, 2011).

However, Mark van der Laan and Rubin (Laan and Rubin, 2006) introduced in 2006 a **double-robust** estimation procedure to **reduce bias** against misspecification. The targeted maximum likelihood estimation (**TMLE**) is a semiparametric, efficient substitution estimator (Laan and Rose, 2011).

3 TMLE

TMLE allows for data-adaptive estimation while obtaining valid statistical inference based on the targeted minimum loss-based estimation and machine learning algorithms to minimise the risk of model misspecification (Laan and Rose, 2011). The main characteristics of **TMLE** are:

1. **TMLE** is a general algorithm for the construction of double-robust, semiparametric, efficient substitution estimators. **TMLE** allows for data-adaptive estimation while obtaining valid statistical inference.
2. **TMLE** implementation uses the G-computation estimand (G-formula). Briefly, the **TMLE** algorithm uses information in the estimated exposure mechanism P(A|W) to update the initial estimator of the conditional expectation of the outcome given the treatment and the set of covariates W, E_0(Y|A,W).
3. The targeted estimates are then substituted into the parameter mapping Ψ. The updating step achieves a targeted bias reduction for the parameter of interest $\Psi(P_0)$ (the true target parameter) and serves to solve the efficient score equation, namely the Influence Curve (IC). As a result, **TMLE** is a **double-robust** estimator.
4. **TMLE** it will be consistent for $\Psi(P_0)$ if either the conditional expectation E $_0$(Y|A,W) or the exposure mechanism P_0(A|W) are estimated consistently.
5. **TMLE** will be efficient if the previous two functions are consistently estimated achieving the lowest asymptotic variance among a large class of estimators. These asymptotic properties typically translate into **lower bias and variance** in finite samples (Bühlmann *et al.*, 2016).
6. The general formula to estimate the ATE using the TMLE method:

$$\psi TMLE, n = \Psi(Q_n^*) = \frac{1}{n}\sum_{i=1}^{n} \bar{Q}_n^1(1,\ W_i) - \bar{Q}_n^1(0,\ W_i). \ (1)$$

7. The efficient influence curve (IC) based on Hampel seminal paper (Hampel, 1974) is applied for statistical inference using TMLE:

$$IC_n(O_i) = \left(\frac{I(A_i = 1)}{g_n(1\,|W_i))} - \frac{I(A_i = 0)}{g_n(0\,|W_i))}\right)\left[Y_i - \bar{Q}_n^1(A_i, W_i)\right] + \bar{Q}_n^1(1,\ W_i) - \bar{Q}_n^1(0,\ W_i) - \psi TMI$$

where the variance of the ATE:

$$\sigma(\psi_0) = \sqrt{\frac{Var(IC_n)}{n}}. (3)$$

8. The procedure is available with standard software such as the **tmle** package in R (Gruber and Laan, 2011).

4 Structural causal framework

4.1 Direct Acyclic Graph (DAG)

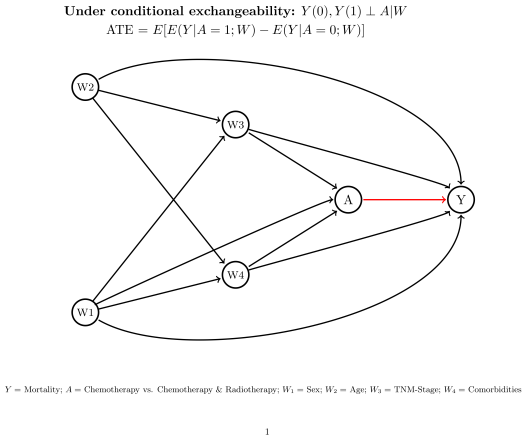

Figure 1. Direct Acyclic Graph (DAG)
Source: Miguel Angel Luque-Fernandez

4.2 DAG interpretation

The ATE is interpreted as the population risk difference in one-year mortality for laryngeal cancer patients treated with chemotherapy versus radiotherapy. Under causal assumptions, and compared with radiotherapy, the risk difference of one-year mortality for patients treated with chemotherapy increases by approximately **20%**.

5 Causal assumptions

To estimate the value of the true causal target parameter $\psi(P_0)$ with a model for the true data generation process P_0 under the counterfactual framework augmented additional untestable cuasal assumptions have to be considered (Rubin, 2011), (Laan and Rose, 2011):

5.1 CMI or Randomization

(Y_0, Y_1 $\perp$A|W) or conditional mean independence (CMI) of the binary treatment effect (A) on the outcome (Y) given the set of observed covariates (W), where W = (W1, W2, W3, ... , W_k).

5.2 Positivity

a ϵ A: P(A=a | W) > 0
P(A=1|W=w) > 0 and P(A=0| W = w) > 0 for each possible w.

5.3 Consistency or SUTVA

The Stable Unit Treatment Value Assumption (SUTVA) incorporates both this idea that **units do not interfere** with one another, and also the concept that for each unit there is only a **single version of each treatment level**.

6 TMLE flow chart

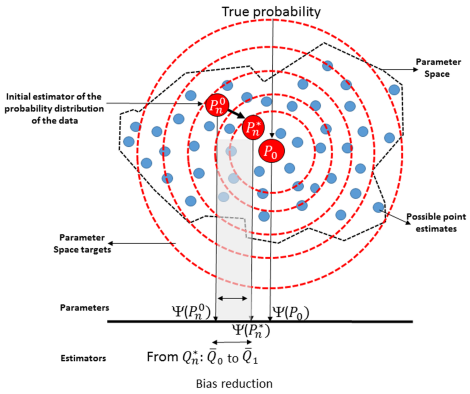

Figure 2. TMLE flow chart (Road map)
Adapted from: Mark van der Laan and Sherri Rose. Targeted learning: causal inference for observational and experimental dataSpringer Series in Statistics, 2011.

7 Data generation

7.1 Simulation

In R we create a function to generate the data. The function will have as input **number of draws** and as output the generated **observed data** (ObsData) including the counterfactuals (Y1, Y0).

The simulated data replicationg the DAG in Figure 1:

1. Y: mortality binary indicator (1 death, 0 alive)
2. A: binary treatment (1 Chemotherapy, 0 Radiotherapy)
3. W1: Gender (1 male; 0 female)
4. W2: Age at diagnosis (0 <65; 1 >=65)
5. W3: Cancer TNM classification (scale from 1 to 4; 1: early stage no metastasis; 4: advanced stage with metastasis)
6. W4: Comorbidities (scale from 1 to 5)

Hide

```
options(digits=4)
generateData <- function(n){
  w1 <- rbinom(n, size=1, prob=0.5)
  w2 <- rbinom(n, size=1, prob=0.65)
  w3 <- round(runif(n, min=0, max=4), digits=3)
  w4 <- round(runif(n, min=0, max=5), digits=3)
  A  <- rbinom(n, size=1, prob= plogis(-0.4 + 0.2*w2 + 0.15*w3 + 0.2
*w4 + 0.15*w2*w4))
  Y  <- rbinom(n, size=1, prob= plogis(-1 + A -0.1*w1 + 0.3*w2 + 0.2
5*w3 + 0.2*w4 + 0.15*w2*w4))

  # counterfactual
  Y.1 <- rbinom(n, size=1, prob= plogis(-1 + 1 -0.1*w1 + 0.3*w2 + 0.
25*w3 + 0.2*w4 + 0.15*w2*w4))
  Y.0 <- rbinom(n, size=1, prob= plogis(-1 + 0 -0.1*w1 + 0.3*w2 + 0.
25*w3 + 0.2*w4 + 0.15*w2*w4))

  # return data.frame
  data.frame(w1, w2, w3, w4, A, Y, Y.1, Y.0)
}
set.seed(7777)
ObsData <- generateData(n=10000)
True_Psi <- mean(ObsData$Y.1-ObsData$Y.0);
cat(" True_Psi:", True_Psi)
```

```
 True_Psi: 0.198
```

Hide

```
Bias_Psi <- lm(data=ObsData, Y~ A + w1 + w2 + w3 + w4)
cat("\n")
```

Hide

```
cat("\n Naive_Biased_Psi:",summary(Bias_Psi)$coef[2, 1])
```

```
 Naive_Biased_Psi: 0.2073
```

Hide

```
Naive_Bias <- ((summary(Bias_Psi)$coef[2, 1])-True_Psi); cat("\n Nai
ves bias:", Naive_Bias)
```

```
 Naives bias: 0.009269
```

Hide

```
Naive_Relative_Bias <- (((summary(Bias_Psi)$coef[2, 1])-True_Psi)/Tr
ue_Psi)*100; cat("\n Relative Naives bias:", Naive_Relative_Bias,"%"
)
```

```
 Relative Naives bias: 4.682 %
```

7.2 Data visualization

Hide

```
# DT table = interactive
# install.packages("DT") # install DT first
library(DT)
datatable(head(ObsData, n = nrow(ObsData)), options = list(pageLengt
h = 5, digits = 2))
```

Show 5 **entries** **Search:**

	w1	w2	w3	w4	A	Y	Y.1	Y.0
1	0	1	3.282	3.541	1	1	0	0
2	1	0	0.047	1.425	1	1	0	0
3	1	0	3.354	4.158	1	1	1	0
4	1	1	2.085	1.737	1	1	0	0
5	0	1	2.487	0.419	1	1	0	1

Showing 1 to 5 of 10,000 entries

Previous 1 2 3 4 5 ... 2000 Next

8 TMLE simple implementation

8.1 Step 1: Q_0(A,W)

Estimation of the initial probability of the outcome (Y) given the treatment (A) and the set of covariates (W), denoted as Q_0(A,**W**). To estimate Q_0(A,**W**) we can use a standard logistic regression model:

$$\text{logit}[P(Y = 1|A, W)] = \beta_0 + \beta_1 A + \hat{\beta}_2^T W.$$

Therefore, we can estimate the initial probability as follows:

$$\bar{Q}^0(A, W) = \text{expit}(\hat{\beta}_0 + \hat{\beta}_1 A + \hat{\beta}_2^T W).$$

The predicted probability can be estimated using the Super-Learner library implemented in the R package "Super-Learner" (Van der Laan *et al.*, 2007) to include any terms that are functions of A or W (e.g., polynomial terms of A and W, as well as the interaction terms of A and W, can be considered).

Consequently, for each subject, the predicted probabilities for both potential outcomes $\bar{Q}^0(0, W)$ and $\bar{Q}^0(1, W)$ can be estimated by setting A = 0 and A = 1 for everyone respectively:

$$\bar{Q}^0(0, W) = \text{expit}(\hat{\beta}_0 + \hat{\beta}_2^T W),$$

and,

$$\bar{Q}^0(1, W) = \text{expit}(\hat{\beta}_0 + \hat{\beta}_1 A + \hat{\beta}_2^T W).$$

Note: see appendix one for a short introduction to the Super-Learner and ensemble learning techniques.

Hide

```
ObsData <-subset(ObsData, select=c(w1,w2,w3,w4,A,Y))
Y  <- ObsData$Y
A  <- ObsData$A
w1 <- ObsData$w1
w2 <- ObsData$w2
w3 <- ObsData$w3
w4 <- ObsData$w4
m  <- glm(Y ~ A + w1 + w2 + w3 + w4, family=binomial, data=ObsData)
Q  <- cbind(QAW = predict(m),
            Q1W = predict(m, newdata=data.frame(A = 1, w1, w2, w3, w
4)),
            Q0W = predict(m, newdata=data.frame(A = 0, w1, w2, w3, w
4)))
Q0 <- as.data.frame(Q)
Y1 <- Q0$Q1W
Y0 <- Q0$Q0W
QA1 <- exp(Y1)/(1+exp(Y1))
QA0 <- exp(Y0)/(1+exp(Y0))
#Inverse logit (probability scale)
psi <- (exp(Y1)/(1+exp(Y1)) - exp(Y0)/(1+exp(Y0)))
Psi <- mean(exp(Y1)/(1+exp(Y1)) - exp(Y0)/(1+exp(Y0))); cat("\n Q0:"
, Psi)
```

```
 Q0: 0.1992
```

Hide

```
df <- round(cbind(Logit=(Q),Pr.Y1=QA1,Pr.Y0=QA0,Psi=psi), digits= 3)
```

Visualizing the first step:

Hide

```
datatable(head(df, n = nrow(df)), options = list(pageLength = 5, dig
its = 3))
```

Show 5 **entries** **Search:**

	QAW	Q1W	QoW	Pr.Y1	Pr.Yo	Psi
1	2.368	2.368	1.341	0.914	0.793	0.122
2	0.045	0.045	-0.981	0.511	0.273	0.239
3	1.824	1.824	0.798	0.861	0.69	0.172
4	1.346	1.346	0.32	0.794	0.579	0.214
5	1.15	1.15	0.123	0.759	0.531	0.229

Showing 1 to 5 of 10,000 entries

Previous 1 2 3 4 5 ... 2000 Next

8.2 Step 2: $g_0(A, W)$

Estimation of the probability of the treatment (A) given the set of covariates (W), denoted as $g_0(A, W)$. We can use again a logistic regression model and to improve the prediction algorithm we can use the Super-Learner library or any other machine learning strategy:

$$\text{logit}[P(A = 1|W)] = \alpha_0 + \alpha_1^T W.$$

Then, we estimate the predicted probability of P(A|W) = $\hat{g}(1, W)$ using:

$$\hat{g}(1, W) = \text{expit}\,(\hat{\alpha_0} + \hat{\alpha_1^T} W).$$

Hide

```
g <- glm(A ~ w2 + w3 + w4, family = binomial)
g1W = predict(g, type ="response");cat("\n Propensity score = g1W","
\n");summary(g1W)
```

```
Propensity score = g1W
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.358   0.594   0.681   0.671   0.759   0.875
```

8.3 Step 3: HAW and ϵ

This step aims to find a better prediction model targeted at minimising the mean squared error (MSE) for the potential outcomes. For the ATE on step convergence is guaranteed given $\bar{Q}^0$ and $\hat{g}(1, W)$.

The fluctuation parameters ($\hat{\epsilon}_0$, $\hat{\epsilon}_1$) are estimated using maximum likelihood procedures by setting logit($\bar{Q}^0(A, W)$) as an offset in a intercept-free logistic regression with H_0 and H_1 as independent variables:

$$\bar{Q^1}(A, W) = \text{expit}\left[\text{logit}\left(\bar{Q^0}(A, W)\right) + \hat{\epsilon_0} H_0(A, W) + \hat{\epsilon_1} H_1(A, W)\right] (5)$$

$$\bar{Q^1}(0, W) = \text{expit}\left[\text{logit}\left(\bar{Q^0}(A, W)\right) + \hat{\epsilon_0} H_0(0, W)\right]$$

$$\bar{Q^1}(1, W) = \text{expit}\left[\text{logit}\left(\bar{Q^0}(A, W)\right) + \hat{\epsilon_1} H_1(1, W)\right]$$

Where,

$$H_0(A, W) = -\frac{I(A = 0)}{\hat{g}(0|W)} \text{ and, } H_1(A, W) = \frac{I(A = 1)}{\hat{g}(1|W)}$$

are the stabilized inverse probability of treatment (A) weights (IPTW), namelly the **clever covariates** and **I** defines an indicator function (note that $\hat{g}(A|W)$ is estimted from step 2).

Hide

```
#Clever covariate and fluctuating/substitution paramteres
h <- cbind(gAW=(A/g1W - (1 - A) / (1 - g1W)), g1W = (1/g1W), g0W=(-1
/ (1 - g1W)))
epsilon <- coef(glm(Y ~ -1 + h[,1] + offset(Q[,"QAW"]), family = bin
omial));cat("\n Epsilon:",epsilon)
```

```
 Epsilon: 0.001189
```

Hide

```
df <- round(cbind(Q0,PS=(g1W),H=(h),epsilon), digits= 4)
```

Visualizing the 3rd step (PS = propensity score; H = IPTW or clever covarites):

Hide

```
datatable(head(df, n = nrow(df)), options = list(pageLength = 5, dig
its = 3))
```

Show 5 **entries** **Search:**

	QAW	Q1W	Q0W	PS	H.gAW	H.g1W	H.g0W	epsil
1	2.3678	2.3678	1.3413	0.8065	1.2399	1.2399	-5.1681	0.(
2	0.0452	0.0452	-0.9812	0.4596	2.1759	2.1759	-1.8504	0.(
3	1.8244	1.8244	0.7979	0.7595	1.3166	1.3166	-4.1582	0.(
4	1.3463	1.3463	0.3199	0.6705	1.4914	1.4914	-3.0351	0.(
5	1.1499	1.1499	0.1234	0.5939	1.6839	1.6839	-2.4622	0.(

Showing 1 to 5 of 10,000 entries

Previous 1 2 3 4 5 ... 2000 Next

8.4 Step 4 $\bar{Q}_n^*$: from $\bar{Q}_0^0$ to $\bar{Q}_1^1$

Afterwards, the estimated probability of the potential outcomes is updated by the substitution parameters ($\hat{\epsilon_0}$, $\hat{\epsilon_1}$). The substitution update is performed by setting A = 0 and A = 1 for each subject in the initial estimate probability of the potential outcomes $\bar{Q}^0(0, W)$, $\bar{Q}^0(1, W)$, as well as in the clever covariates $H_0(0, W)$ and $H_1(1, W)$.

For the $\Psi(\bar{Q}_n^*)$, the updated estimate of the potential outcomes only needs one iteration $\Psi(\bar{Q}_n^*)$ from $\bar{Q}^0(A, W)$ => $\bar{Q}^1(A, W)$. Therefore, model (5) targets $E[\hat{Y}_{A=0}]$ and $E[\hat{Y}_{A=1}]$ simultaneously by including both $H_0(A, W)$ and $H_1(A, W)$ in the model. Hence ψ is finally estimated as follows:

$$\psi TMLE, n = \Psi(Q_n^*) = \frac{1}{n}\sum_{i=1}^{n} \bar{Q}_n^1(1,\ W_i) - \bar{Q}_n^1(0,\ W_i). \ (1)$$

Hide

```
Qstar <- plogis(Q + epsilon*h)
psi <- (Qstar[,"Q1W"] - Qstar[,"Q0W"])
Psi <- mean(Qstar[,"Q1W"] - Qstar[,"Q0W"]);
cat("TMLE_Psi:", Psi)
```

```
TMLE_Psi: 0.2004
```

Hide

```
cat("\n TMLE.SI_bias:", abs(True_Psi-Psi))
```

```
 TMLE.SI_bias: 0.002383
```

Hide

```
cat("\n Relative_TMLE.SI_bias:",abs(True_Psi-Psi)/True_Psi*100,"%")
```

```
 Relative_TMLE.SI_bias: 1.204 %
```

Visualizing the 4th step (H = IPTW or clever covarites):

Hide

```
df <- round(cbind(Q0=(Q0),H=(h),epsilon,psi), digits= 4)
datatable(head(df, n = nrow(df)), options = list(pageLength = 5, dig
its = 3))
```

Show 5 **entries** **Search:**

	Qo.QAW	Qo.Q1W	Qo.QoW	H.gAW	H.g1W	H.goW	ep
1	2.3678	2.3678	1.3413	1.2399	1.2399	-5.1681	
2	0.0452	0.0452	-0.9812	2.1759	2.1759	-1.8504	
3	1.8244	1.8244	0.7979	1.3166	1.3166	-4.1582	
4	1.3463	1.3463	0.3199	1.4914	1.4914	-3.0351	
5	1.1499	1.1499	0.1234	1.6839	1.6839	-2.4622	

Showing 1 to 5 of 10,000 entries

Previous 1 2 3 4 5 ... 2000 Next

Hide

```
cat("\n Psi first row:", plogis((0.0012*1.2399) + (2.3678)) - (plogi
s((0.0012*-5.1681) + (1.3413))))
```

```
 Psi first row: 0.1228
```

8.5 Step 5: Inference

TMLE uses the efficient influence curve (IC) for inference (i.e., to obtain standard errors for ψ).

$$IC_n(O_i) = \left(\frac{I(A_i = 1)}{g_n(1 \mid W_i))} - \frac{I(A_i = 0)}{g_n(0 \mid W_i))} \right) \left[Y_i - \bar{Q}_n^1(A_i, W_i) \right] + \bar{Q}_n^1(1, W_i) - \bar{Q}_n^1(0, W_i) - \psi T$$

where the standard deviation for ψ is estimated as follows:

$$\sigma(\psi_0) = \sqrt{\frac{Var(IC_n)}{n}}. (3)$$

Note: see appendix two for a short introduction to the theory of the Influence Curve.

Hide

```
Q  <- as.data.frame(Q)
Qstar <- as.data.frame(Qstar)
IC <- h[,1]*(Y-plogis(Q$QAW)) + plogis(Qstar$Q1W - Qstar$Q0W) - Psi;
summary(IC)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -1.217  -0.794   0.520   0.351   0.797   6.823
```

Hide

```
n <- nrow(ObsData)
varHat.IC <- var(IC)/n; varHat.IC
```

```
[1] 9.836e-05
```

Hide

```
#Psi and 95%CI for Psi
cat("\n TMLE_Psi:", Psi)
```

```

 TMLE_Psi: 0.2004
```

Hide

```
cat("\n 95%CI:", c(Psi-1.96*sqrt(varHat.IC),  Psi+1.96*sqrt(varHat.IC)))
```

```

 95%CI: 0.1809 0.2198
```

Hide

```
cat("\n TMLE.SI_bias:", abs(True_Psi-Psi))
```

```

 TMLE.SI_bias: 0.002383
```

Hide

```
cat("\n Relative_TMLE.SI_bias:",abs(True_Psi-Psi)/True_Psi*100,"%")
```

```

 Relative_TMLE.SI_bias: 1.204 %
```

9 TMLE vs. AIPTW

1. The advantages of **TMLE** have repeatedly been demonstrated in both simulation studies and applied analyses (Laan and Rose, 2011).
2. Evidence shows that **TMLE** provides the less unbiased ATE estimate compared with other double-robust estimators (Neugebauer and Laan, 2005), (Laan and Rose, 2011) such as the combination of regression adjustment with inverse probability of treatment weighting (IPTW-RA) and the augmented inverse probability of treatment weighting (AIPTW). The **AIPTW** estimation is a two-step procedure with two equations (propensity score and mean outcome equations).
3. To estimate the ATE using the **AIPTW** estimator one can set the estimation equation (EE) (4) equal to zero and use bootstrap to derive 95% confidence intervals (CI). However, solving the EE using the generalized method of moments (GMM), stacking both equations (propensity score and outcome), reduces the estimation and inference steps to only one. However, given that the propensity score in equation (4) can easily fall outside the range [0, 1] (if for some observations $g_n(1|W_i)$ is close to 1 or 0) the **AIPTW** estimation can be unstable (near violation

of the positivity assumption). **AIPTW** instability under near violation of the positivity assumption represents the price of not being a substitution estimator as **TMLE**.

$$\psi_0^{AIPTW-ATE} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{I(A_i=1)}{g_n(1\,|W_i))} - \frac{I(A_i=0)}{g_n(0\,|W_i))}\right)\left[Y_i - \bar{Q}_n^0(A_i, W_i)\right] + \frac{1}{n}\sum_{i=1}^{n}\bar{Q}_n^0(1,\ W_i) - \bar{Q}$$

Hide

```
AIPTW <- mean((h[,1]*(Y - plogis(Q$QAW)) + (plogis(Q$Q1W) - plogis(Q
$Q0W))));AIPTW
```

```
[1] 0.2004
```

Hide

```
cat("\n AIPTW_bias:", abs(True_Psi - AIPTW))
```

```

 AIPTW_bias: 0.002379
```

Hide

```
cat("\n Relative_AIPTW_bias:",abs(True_Psi - AIPTW) / True_Psi*100,"
%")
```

```

 Relative_AIPTW_bias: 1.202 %
```

The simple TMLE algorithm shows similar relative bias than AIPTW. However, here below, we can see that TMLE performance, compared with AIPTW, improves when calling the Super-Learner and ensemble learning techniques integrated into the TMLE algorithm.

10 TMLE using the Super-Learner

With TMLE we can call the Super-Learner (SL). The SL is a R-package using V-fold cross-validation and ensembled learning (prediction using all the predictions of multiple stacked learning algorithms) techniques to improve model prediction performance (Breiman, 1996).

The basic implementation of TMLE in the R-package **tmle** uses by default three algorithms:
1. SL.glm (main terms logistic regression of A and W),
2. SL.step (stepwise forward and backward model selection using AIC criterion, restricted to second order polynomials) and,
3. SL.glm.interaction (a glm variant that includes second order polynomials and two by two interactions of the main terms included in the model).

The principal interest of calling the Super-Learner is to obtain the less-unbiased estimated for $\bar{Q}_n^0(A, W)$ and $g_0(A, W)$. It is achieved by obtaining the smallest expected loss function for Y or A (binary outcomes), respectively. For instance, the negative logarithmic loss function for Y is computed as the minimizer of the expected squared error loss:

$$\bar{Q}_0 = \arg\min_{\bar{Q}} E_0 L(O, \bar{Q}),$$

where $L(O, \bar{Q})$ is:

$$(Y - \bar{Q}(A, W))^2$$

Note: see appendix one for a short introduction to the Super-Learner and ensemble learning techniques.

1. **Step One**: $\bar{Q}_n^0(A, W)$ prediction

Hide

```
#E(Y|A,W) prediction
library(SuperLearner)
#Specify SuperLearner libraries
SL.library <- c("SL.glm","SL.step","SL.glm.interaction")
#Data frame with X with baseline covariates and exposure A
X <- subset(ObsData, select = c(A, w1, w2, w3, w4))
n <- nrow(ObsData)
#Create data frames with A=1 and A=0
X1<-X0<-X
X1$A <-1
X0$A <-0
#Create new data by stacking X, X1, and X0
newdata <- rbind(X,X1,X0)
#Call superlearner
Qinit <- SuperLearner(Y = ObsData$Y, X = X, newX = newdata, SL.libra
ry=SL.library, family="binomial")
Qinit
```

```
Call:
SuperLearner(Y = ObsData$Y, X = X, newX = newdata, family = "binomia
l", SL.library = SL.library)

                          Risk   Coef
SL.glm_All              0.1766 0.6002
SL.step_All             0.1766 0.0000
SL.glm.interaction_All 0.1767 0.3998
```

Hide

```
#Predictions
#Pred prob of mortality (Y) given A, W
QbarAW <- Qinit$SL.predict[1:n]
#Pred prob of dying for each subject given A=1 and w
Qbar1W <- Qinit$SL.predict[(n+1):(2*n)]
#Pred prob of dying for each subject given A=0 and w
Qbar0W <- Qinit$SL.predict[(2*n+1):(3*n)]
#Simple substitution estimator Psi(Q0)
PsiHat.SS <- mean(Qbar1W - Qbar0W);PsiHat.SS
```

```
[1] 0.199
```

2. **Step two**: $g_0(A, W)$ prediction

Hide

```
#Step 2 g_0(A|W) with SuperLearner
w <- subset(ObsData, select=c(w1,w2,w3,w4))
gHatSL <- SuperLearner(Y=ObsData$A, X = w, SL.library = SL.library,
family = binomial)
gHatSL;mean(gHatSL)
```

```
Call:
SuperLearner(Y = ObsData$A, X = w, family = binomial, SL.library = S
L.library)

                          Risk   Coef
SL.glm_All              0.2091 0.0000
SL.step_All             0.2091 0.3803
SL.glm.interaction_All 0.2090 0.6197
[1] NA
```

Hide

```
#Generate the pred prob of A=1 and, A=0 given covariates
gHat1W <- gHatSL$SL.predict
gHat0W <- 1 - gHat1W
#Step 3: Clever covariate
HAW <- as.numeric(ObsData$A==1)/gHat1W - as.numeric(ObsData$A==0)/gH
at0W;mean(HAW)
```

```
[1] 0.002954
```

Hide

```
H1W <-  1/gHat1W
H0W <- -1/gHat0W
```

3. **Steps 3 and 4**: fluctuation step and substitution estimation for $\bar{Q}_n^0(A, W)$ to $\bar{Q}_n^1(A, W)$

Hide

```
#Step 4: Substitution estimaiton Q* of the ATE.
logitUpdate <- glm(ObsData$Y ~ -1 + offset(qlogis(QbarAW)) + HAW, fa
mily='binomial')
eps <- logitUpdate$coef;eps
```

```
      HAW
0.0004483
```

Hide

```
#Calculating the predicted values for each subject under each treatm
ent A=1, A=0
QbarAW.star <- plogis(qlogis(QbarAW)+eps*HAW)
Qbar1W.star <- plogis(qlogis(Qbar1W)+eps*H1W)
Qbar0W.star <- plogis(qlogis(Qbar0W)+eps*H0W)
PsiHat.TMLE.SL <- mean(Qbar1W.star) - mean(Qbar0W.star)
cat("PsiHat.TMLE.SL:", PsiHat.TMLE.SL)
```

```
PsiHat.TMLE.SL: 0.1995
```

Hide

```
cat("\n PsiHat.TMLE.SL_bias:", abs(True_Psi - PsiHat.TMLE.SL))
```

```

 PsiHat.TMLE.SL_bias: 0.001456
```

Hide

```
cat("\n Relative_PsiHat.TMLE.SL_bias:",abs(True_Psi - PsiHat.TMLE.SL
)/True_Psi*100,"%")
```

```

 Relative_PsiHat.TMLE.SL_bias: 0.7354 %
```

TMLE with machine learning algorithms decreases bias compared with the previous AIPTW and TMLE (without Super Learner) estimators.

11 R-TMLE

Using the R-package **tmle**.

The basic implementation of TMLE in the R-package **tmle** uses by default three algorithms:
1. SL.glm (main terms logistic regression of A and W),
2. SL.step (stepwise forward and backward model selection using AIC criterion, restricted

to second order polynomials) and,
3. SL.glm.interaction (a glm variant that includes second order polynomials and two by two interactions of the main terms included in the model).

Hide

```
library(tmle)
set.seed(7777)
w <- subset(ObsData, select=c(w1,w2,w3,w4))
tmle <- tmle(Y, A, W=w)
cat("TMLER_Psi:", tmle$estimates[[2]][[1]],";","95%CI(", tmle$estima
tes[[2]][[3]],")")
```

```
TMLER_Psi: 0.1994 ; 95%CI( 0.1799 0.2189 )
```

Hide

```
cat("\n TMLE_bias:", abs(True_Psi-tmle$estimates[[2]][[1]]))
```

```
 TMLE_bias: 0.001382
```

Hide

```
cat("\n Relative_TMLE_bias:",abs(True_Psi-tmle$estimates[[2]][[1]])/
True_Psi*100,"%")
```

```
 Relative_TMLE_bias: 0.6979 %
```

TMLE implementation in the R-package **tmle** improves the estimation of the inverse-propability of treatment weights. It bounds by default the distribution of the weights for the propensity score to (0.025th and 0.975th percentiles) to decrease the impact of near-positivity violations. Also, note that at each time that we run the super-learner the V-fold cross-validation splits the data randomly. So, we have to set a seed for replicability. It is why the bias using the **tmle** R-package decreases from 0.7354 to 0.6979.

12 R-TMLE decreasing bias by calling more advanced machine-learning libraries

In addition to the default algorithms implemented in the R-tmle package, we can even decrease more the bias of our estimation by calling more efficient machine learing algorithms, such as generalized additive models, Random Forest and, Recursive Partitioning and Regression Trees:

Hide

```
SL.TMLER.Psi <- tmle(Y=Y, A=A, W=w, family="binomial",
    Q.SL.library = c("SL.glm", "SL.step", "SL.glm.interaction", "SL.
gam", "SL.randomForest", "SL.rpart"),
    g.SL.library = c("SL.glm", "SL.step", "SL.glm.interaction", "SL.
gam", "SL.randomForest", "SL.rpart"))
cat("SL.TMLER.Psi:", SL.TMLER.Psi$estimates[[2]][[1]],";","95%CI(",
SL.TMLER.Psi$estimates[[2]][[3]],")")
```

```
SL.TMLER.Psi: 0.1993 ; 95%CI( 0.1799 0.2187 )
```

Hide

```
cat("\n SL.TMLER.Psi_bias:", abs(True_Psi-SL.TMLER.Psi$estimates[[2]
][[1]]))
```

```
 SL.TMLER.Psi_bias: 0.001292
```

Hide

Hide

```
cat("\n Relative_SL.TMLER.Psi_bias:",abs(True_Psi-SL.TMLER.Psi$estim
ates[[2]][[1]])/True_Psi*100,"%")
```

```
 Relative_SL.TMLER.Psi_bias: 0.6523 %
```

13 Conclusions

We have demonstrated:

1. **TMLE excels** the AIPTW estimator and,
2. TMLE **best performance** is obtained when calling more advanced **Super-Learner** algorithms.

14 Appendix One

Efron in 1982 showed that the empirical **Influence Curve** estimate of standard error is the same as the one obtained using the **infinitesimal jackknife** and the **nonparametric delta method** (Efron and Efron, 1982).

1. The Delta Method uses the first order of the Taylor series expansion:

$$f(x) \approx f(x) - f(\mu) + (x - \mu)f'(\mu);$$

where $f'(\mu)$ is the derivative of the function with respect to X evaluated at the mean of X. Therefore, squaring both terms, the variance is approximately estimate as follwos:

$$\mathrm{E}[f(x) - f(\mu)]^2 \approx \mathrm{E}(x - \mu)^2 \times [f'(\mu)]^2;$$

The left-hand side of the above equation is approximately the variance of $f(x)$ and applied to the empirical distribution of X, the sample estimate of variance for X replaces:

$$\mathrm{E}(x_i - \mu)^2.$$

The infinitesimal jackknife estimate of the standard error is defined as follows:

$$SD_{ij}(\theta_e) = \left(\frac{\sum_{i=1}^{n} U_i^2}{n^2}\right)^{1/2};$$

where θ_e is the estimate of the parameter θ and U_i is a **directional derivative** in the direction of the ith coordinate centered at the mean of the empirical distribution function.

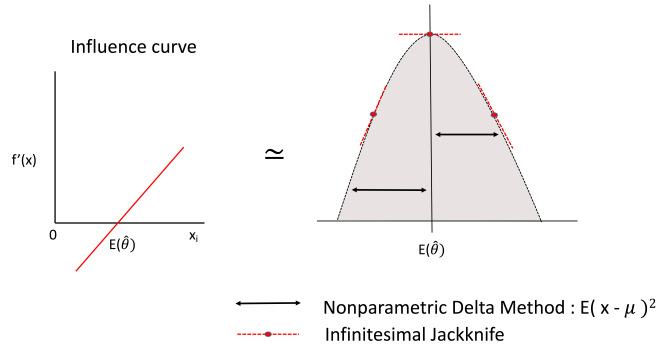

Figure 3. Estimate of the ψ Standard Error using the efficient Influence Curve.
Image credit: Miguel Angel Luque-Fernandez.

15 Appendix Two

With TMLE we can call the R-package **Super-Learner (SL)**. The *SL* uses **cross-validation** and **ensembled learning** (using all the predictions of multiple stacked learning algorithms) techniques to improve model prediction performance (Breiman, 1996).

The **SL** algorithm provides a system based on V-fold cross-validation (Efron and Gong, 1983) (10-folds) to combine adaptively multiple algorithms into an improved estimator, and returns a function than can be used for prediction in new datasets.

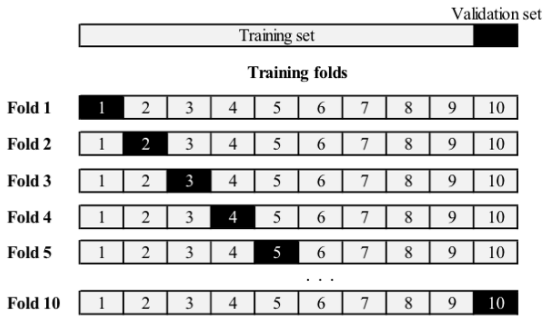

Figure 4: 10-fold cross-validation algorithm.
The basic implementation of TMLE in the R-package **tmle** uses by default three algorithms:
1. SL.glm (main terms logistic regression of A and W),
2. SL.step (stepwise forward and backward model selection using AIC criterion, restricted to second order polynomials) and,
3. SL.glm.interaction (a glm variant that includes second order polynomials and two by two interactions of the main terms included in the model).

The principal interest of calling the Super-Learner is to obtain the less-unbiased estimated for $\bar{Q}_n^0(A, W)$ and $g_0(A, W)$. It is achieved by obtaining the smallest expected loss function for Y or A (binary outcomes), respectively. For instance, the negative logarithmic loss function for Y is computed as the minimizer of the expected squared error loss:

$$\bar{Q}_0 = \arg\min_{\bar{Q}} E_0 L(O, \bar{Q}),$$

where $L(O, \bar{Q})$ is:

$$(Y - \bar{Q}(A, W))^2$$

The **SL** algorithm first split the data into ten blocks and fits each of the selected algoriths on the training set (non-shaded blocks), then predicts the estimated probabilities of the outcome (Y) using the validation set (shaded block) for each algorithm, based on the corresponding training set. Afterwards, the **SL** estimates the the cross-validating risk for each algorithm averaging the risks across validation sets resulting in one estimated cross-validated risk for each algorithm. Finally, the **SL** selects the combination of Z that minimises the cross-validation risk, defined as the minimum mean square error for each of the selected algorithms using Y and Z. A weighted combination of the algorithms (ensemble learning) in Z is then used to predict the outcome (Y) (see Figure 5).

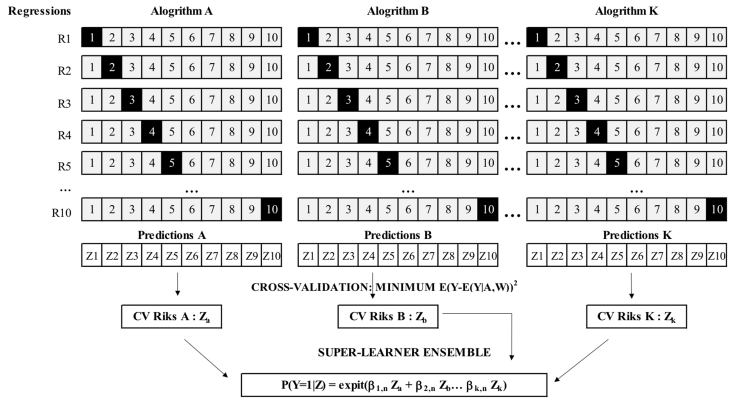

Figure 5: Flow Diagram for the Super-Learner algorithm.

16 Abbreviations

TMLE: Targeted maximum likelihood estimation
SL: Super Learner
IPTW: Inverse probability of treatment weighting
AIPTW: Augmented inverse probability of treatment weighting
MSE: Mean squared error
SE: Standard error
EE: Estimation equations
GMM: Generalised method of moments
O: Observed ordered data structure
W: Vector of covariates
A: Binary treatment or exposure
Y: Binary outcome
Y_1, Y_O: Counterfactual outcomes with binary treatment A
P_0: True data-generating distribution
$\Psi(P_0)$:True target parameter
$\psi_0 = \Psi(P_0)$: True target parameter value
g_0: Propensity score for the treatment mechanism (A)
g_0: Estimate of g_0
ϵ: Fluctuation parameter
ϵ_n : Estimate of ϵ H_n^* : Clever covariate estimate (inverse probability of treatment weight)
$L(O, \bar{Q})$: Example of a loss function where it is a function of O and $\bar{Q}$
$(Y - \bar{Q}(A, W))^2$: Expected squared error loss
$\bar{Q}_0$: Conditional mean of outcome given parents; $E_0(Y|A, W)$
$\bar{Q}_n$: Estimate of $\bar{Q}_0$
$\bar{Q}_n^0$: Initial estimate of $\bar{Q}_0$
$\bar{Q}_n^1$: First updated estimate of $\bar{Q}_0$
$\bar{Q}_n^*$: Targeted estimate of $\bar{Q}_n^0$ in TMLE procedure; $\bar{Q}_n^*$ may equal $\bar{Q}_n^1$

17 Session Info

Hide

```
devtools::session_info()
```

```
Session info -------------------------------------------------------
-------------------
```

```
 setting  value
 version  R version 3.4.0 Patched (2017-05-08 r72665)
 system   x86_64, darwin15.6.0
 ui       RStudio (1.0.143)
 language (EN)
 collate  en_US.UTF-8
 tz       Europe/London
 date     2017-05-24
```

```
Packages -----------------------------------------------------------
-------------------
```

```
 package      * version date       source
 backports      1.0.5   2017-01-18 CRAN (R 3.4.0)
 base         * 3.4.0   2017-05-10 local
 base64enc      0.1-3   2015-07-28 CRAN (R 3.4.0)
 codetools      0.2-15  2016-10-05 CRAN (R 3.4.0)
 compiler       3.4.0   2017-05-10 local
 datasets     * 3.4.0   2017-05-10 local
 devtools       1.13.1  2017-05-13 CRAN (R 3.4.0)
 digest         0.6.12  2017-01-27 CRAN (R 3.4.0)
 DT           * 0.2     2016-08-09 CRAN (R 3.4.0)
 evaluate       0.10    2016-10-11 CRAN (R 3.4.0)
 foreach      * 1.4.3   2015-10-13 CRAN (R 3.4.0)
 gam          * 1.14-4  2017-04-25 CRAN (R 3.4.0)
 graphics     * 3.4.0   2017-05-10 local
 grDevices    * 3.4.0   2017-05-10 local
 highr          0.6     2016-05-09 CRAN (R 3.4.0)
 htmltools      0.3.6   2017-04-28 CRAN (R 3.4.0)
 htmlwidgets    0.8     2016-11-09 CRAN (R 3.4.0)
 iterators      1.0.8   2015-10-13 CRAN (R 3.4.0)
 jsonlite       1.4     2017-04-08 CRAN (R 3.4.0)
 knitr          1.16    2017-05-18 CRAN (R 3.4.0)
 magrittr       1.5     2014-11-22 CRAN (R 3.4.0)
 markdown       0.8     2017-04-20 CRAN (R 3.4.0)
 memoise        1.1.0   2017-04-21 CRAN (R 3.4.0)
 methods      * 3.4.0   2017-05-10 local
 nnls         * 1.4     2012-03-19 CRAN (R 3.4.0)
 randomForest * 4.6-12  2015-10-07 CRAN (R 3.4.0)
 Rcpp           0.12.10 2017-03-19 CRAN (R 3.4.0)
 rmarkdown      1.5     2017-04-26 CRAN (R 3.4.0)
 rpart        * 4.1-11  2017-03-13 CRAN (R 3.4.0)
 rprojroot      1.2     2017-01-16 CRAN (R 3.4.0)
 rsconnect      0.8     2017-05-21 Github (rstudio/rsconnect@106b6a0
)
 rstudioapi     0.6     2016-06-27 CRAN (R 3.4.0)
 splines      * 3.4.0   2017-05-10 local
 stats        * 3.4.0   2017-05-10 local
 stringi        1.1.5   2017-04-07 CRAN (R 3.4.0)
 stringr        1.2.0   2017-02-18 CRAN (R 3.4.0)
 SuperLearner * 2.0-21  2016-12-01 CRAN (R 3.4.0)
 tmle         * 1.2.0-5 2017-01-07 CRAN (R 3.4.0)
 tools          3.4.0   2017-05-10 local
 utils        * 3.4.0   2017-05-10 local
 withr          1.0.2   2016-06-20 CRAN (R 3.4.0)
 yaml           2.1.14  2016-11-12 CRAN (R 3.4.0)
```

18 Thank you

Thank you for participating in this tutorial.
If you have updates or changes that you would like to make, please send me (https://github.com/migariane/MALF) a pull request. Alternatively, if you have any questions, please e-mail me. You can cite this repository as:
Luque-Fernandez MA, (2017). Taregeted Maximum Likelihood Estimation for a Binary Outcome: Tutorial and Guided Implementation. GitHub repository, http://migariane.github.io/TMLE.nb.html (http://migariane.github.io/TMLE.nb.html).

Miguel Angel Luque Fernandez
E-mail: *miguel-angel.luque at lshtm.ac.uk*
Twitter @WATZILEI

19 References

Breiman L. (1996). Stacked regressions. *Machine learning* **24**: 49–64.

Bühlmann P, Drineas P, Laan M van der, Kane M. (2016). Handbook of big data. CRC Press.

Efron B, Efron B. (1982). The jackknife, the bootstrap and other resampling plans. SIAM.

Efron B, Gong G. (1983). A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American Statistician* **37**: 36–48.

Greenland S, Robins JM. (1986). Identifiability, exchangeability, and epidemiological confounding. *International journal of epidemiology* **15**: 413–419.

Gruber S, Laan M van der. (2011). Tmle: An r package for targeted maximum likelihood estimation. *UC Berkeley Division of Biostatistics Working Paper Series*.

Hampel FR. (1974). The influence curve and its role in robust estimation. *Journal of the American Statistical Association* **69**: 383–393.

Laan M van der, Rose S. (2011). Targeted learning: Causal inference for observational and experimental data. Springer Series in Statistics.

Laan MJ van der, Rubin D. (2006). Targeted maximum likelihood learning. *The International Journal of Biostatistics* **2**.

Lunceford JK, Davidian M. (2004). Stratification and weighting via the propensity score in estimation of causal treatment effects: A comparative study. *Statistics in Medicine* **23**: 2937–2960 (tel:2937–2960).

Neugebauer R, Laan M van der. (2005). Why prefer double robust estimators in causal inference? *Journal of Statistical Planning and Inference* **129**: 405–426.

Robins JM, Hernan MA, Brumback B. (2000). Marginal structural models and causal inference in epidemiology. *Epidemiology* 550–560.

Rosenbaum PR, Rubin DB. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika* **70**: 41–55.

Rubin DB. (2011). Causal inference using potential outcomes. *Journal of the American Statistical Association*.

Rubin DB. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology* **66**: 688.

Van der Laan MJ, Polley EC, Hubbard AE. (2007). Super learner. *Statistical applications in genetics and molecular biology* **6**.