R Lab 6 - Inference

Introduction to Causal Inference

Goals:
1. Review estimation based on the simple substitution estimator, inverse probability of treatment weighted (IPTW) estimator, and TMLE.
2. Use the sample variance of the influence curve to obtain inference for TMLE.
3. Use the non-parametric bootstrap to obtain inference for all 3 algorithms.

1 Background: The Lost World - Jurassic Park II

Dr. Alan Grant: "T-Rex doesn't want to be fed. He wants to hunt. Can't just suppress 65 million years of gut instinct." - Michael Crichton

Suppose we are interested in estimating the causal effect of "being a good guy" on survival on Isla Sorna, where dinosaurs have been living free after Jurassic Park was shut down. Suppose we have data on the following variables

- $W1$: gender (1 for male; 0 for female)
- $W2$: previously had traveled to (and survived) Jurassic Park (1 for yes; 0 for no)
- $W3$: intelligence (scale from 0 to 1; with higher values for smarter)
- $W4$: maritial arts training (scale from 0 to 5; with higher values for more)
- A: "good guy" (1 for yes; 0 for no)
- Y: survival (1 for yes; 0 for no)

Let $W = (W1, W2, W3, W4)$ be the vector of baseline covariates.

http://www.thesambarnes.com/web-project-management/account-management-for-the-web-project-manager-part-1/

2 Step 6. Estimate $\Psi(\mathbb{P}_0) = \mathbb{E}_0\left[\bar{Q}_0(1, W) - \bar{Q}_0(0, W)\right]$

This is a review of Lab 5. Re-use your code.

1. Import the data set `RLab6.Inference.csv` and assign it to object `ObsData`. Assign the number of subjects to `n`. Set the seed to 1.

2. Load the `SuperLearner` package. Specify the Super Learner library with the following algorithms: `SL.glm`, `SL.step` and `SL.glm.interaction`. In practice, we would want to use a larger library with a mixture of simple (e.g. parametric) and more aggressive libraries.

3. Use Super Learner to estimate $\mathbb{E}_0(Y|A,W) = \bar{Q}_0(A,W)$, which is the conditional probability of surviving given the exposure (being a good guy) and baseline covariates.

4. Evaluate the simple substitution estimator by plugging the estimates $\hat{\bar{Q}}(1,W)$ and $\hat{\bar{Q}}(0,W)$ into the target parameter mapping:
$$\hat{\Psi}_{SS}(\hat{\mathbb{P}}) = \frac{1}{n}\sum_{i=1}^{n} \hat{\bar{Q}}(1,W_i) - \hat{\bar{Q}}(0,W_i)$$

5. Use Super Learner to estimate the exposure mechanism $g_0(A=1|W) = \mathbb{P}_0(A=1|W)$, which is the conditional probability of being a "good guy", given baseline covariates.

6. Use these estimates to create the clever covariate:
$$\hat{H}(A,W) = \left(\frac{\mathbb{I}(A=1)}{\hat{g}(1|W)} - \frac{\mathbb{I}(A=0)}{\hat{g}(0|W)}\right)$$
Calculate `H.AW` for each subject based on his/her observed exposure. Also evaluate the clever covariate at $A = 1$ and $A = 0$: `H.1W` and `H.0W`

```
> H.AW<- as.numeric(ObsData$A==1)/gHat1W - as.numeric(ObsData$A==0)/gHat0W
> H.1W<- 1/gHat1W
> H.0W<- -1/gHat0W
```

7. Evaluate the IPTW estimator by taking the empirical mean of the weighted observations:
$$\hat{\Psi}_{IPTW}(\hat{\mathbb{P}}) = \frac{1}{n}\sum_{i=1}^{n} \hat{H}(A_i,W_i) \times Y_i$$

8. Update the initial estimates.

   (a) Run logistic regression of the outcome Y on the clever covariate $\hat{H}(A,W)$, using the logit of the initial estimate as offset and suppressing the intercept.

   ```
   > logitUpdate<- glm(ObsData$Y ~ -1 +offset(qlogis(QbarAW)) + H.AW, family='binomial')
   > #  We suppress the intercept by including -1 on the right hand side.
   > #  Logit(x)=log[x/(1-x)]... in R, qlogis(x)
   > #  family='binomial' runs logistic regression
   ```

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

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

   (b) Update the initial estimates of $\bar{Q}_0(A,W)$, $\bar{Q}_0(1,W)$ and $\bar{Q}_0(0,W)$ according to the fluctuation model

   ```
   > QbarAW.star<- plogis(qlogis(QbarAW)+ epsilon*H.AW)
   > Qbar1W.star<- plogis(qlogis(Qbar1W)+ epsilon*H.1W)
   > Qbar0W.star<- plogis(qlogis(Qbar0W)+ epsilon*H.0W)
   > # logit: qlogis
   > # inverse-logit: plogis
   ```

9. Substitute the updated fits into the target parameter mapping:
$$\hat{\Psi}_{TMLE}(\mathbb{P}_n) = \frac{1}{n}\sum_{i=1}^{n} \left[\hat{\bar{Q}}^*(1,W_i) - \hat{\bar{Q}}^*_n(0,W_i)\right]$$

Solution:

```
> #-----------------------------------------------------
> # 1.  Import the data set and assign it to object ObsData; explore
> #-----------------------------------------------------
> ObsData<- read.csv("RLab6.Inference.csv")
> n<- nrow(ObsData)
> n

[1] 2500

> set.seed(1)

> #-----------------------------------------------------
> # 2. load the SuperLearner library and specify the package
> #-----------------------------------------------------
> library("SuperLearner")
> # specify the library
> SL.library<- c("SL.glm", "SL.step", "SL.glm.interaction")

> #-----------------------------------------------------
> #3-9: Create a function to handcode TMLE with Super Learner
> #-----------------------------------------------------
> run.tmle <- function(ObsData, SL.library){
+
+   #-------------------------------------------
+   # Estimate the conditional mean outcome Qbar(A,W)
+   #-------------------------------------------
+
+   # dataframe X with baseline covariates and exposure
+   X<-subset(ObsData, select=c(A, W1, W2, W3, W4))
+   # set the A=1 in X1 and the A=0 in X0
+   X1 <- X0<-X
+   X1$A <- 1         # under exposure
+   X0$A <- 0        # under control
+
+   # call Super Learner for estimation of QbarAW
+   QbarSL<- SuperLearner(Y=ObsData$Y, X=X, SL.library=SL.library, family="binomial")
+   # QbarSL
+
+   # initial estimates of the outcome, given the observed exposure & covariates
+   QbarAW <- predict(QbarSL, newdata=ObsData)$pred
+   # estimates of the outcome, given A=1 and covariates
+   Qbar1W<- predict(QbarSL, newdata=X1)$pred
+   # estimates of the outcome, given A=0 and covariates
+   Qbar0W<- predict(QbarSL, newdata=X0)$pred
+
+   # simple substitution estimator:
+   PsiHat.SS<-mean(Qbar1W - Qbar0W)
+
+   #-------------------------------------------
+   # Estimate the exposure mechanism g(A|W)
+   #------------------------------------------
+
+   # call Super Learner for the exposure mechanism
+   gHatSL<- SuperLearner(Y=ObsData$A, X=subset(ObsData, select= -c(A,Y)),
+                       SL.library=SL.library, family="binomial")
+   # generate predicted prob being exposed, given baseline covariates
+   gHat1W<- gHatSL$SL.predict
+   # predicted prob of not being exposed, given baseline covariates
+   gHat0W<- 1- gHat1W
+
+   # # predicted prob of observed exposure, given baseline cov
+   # gHatAW<- rep(NA, n)
+   # gHatAW[ObsData$A==1]<- gHat1W[ObsData$A==1]
+   # gHatAW[ObsData$A==0]<- gHat0W[ObsData$A==0]
+
+   #--------------------------------------------------
+   # Clever covariate H(A,W) for each subject
+   #--------------------------------------------------
+   H.AW<- as.numeric(ObsData$A==1)/gHat1W - as.numeric(ObsData$A==0)/gHat0W
+
+   # also want to evaluate the clever covariates at A=1 and A=0 for all subjects
+   H.1W<- 1/gHat1W
+   H.0W<- -1/gHat0W
+
+
+   #IPTW estimator of the G-computation formula:
+   PsiHat.IPTW <-mean( H.AW*ObsData$Y)
+
+   #------------------------------------------
+   # Update the initial estimator of Qbar_0(A,W)
+   #------------------------------------------
+   logitUpdate<- glm(ObsData$Y ~ -1 +offset(qlogis(QbarAW)) + H.AW, family='binomial')
+   epsilon <- logitUpdate$coef
+
+   QbarAW.star<- plogis(qlogis(QbarAW)+ epsilon*H.AW)
+   Qbar1W.star<- plogis(qlogis(Qbar1W)+ epsilon*H.1W)
+   Qbar0W.star<- plogis(qlogis(Qbar0W)+ epsilon*H.0W)
+
+   #------------------------------------------
+   # Estimate Psi(P_0)
+   #------------------------------------------
+
+   PsiHat.TMLE <- mean(Qbar1W.star - Qbar0W.star)
+
+   #------------------------------------------
+   # Return point estimates, targeted estimates of Qbar_0(A,W),
+   # and thevector of clever covariates
+   #------------------------------------------
+
+   estimates <- data.frame(cbind(PsiHat.SS=PsiHat.SS, PsiHat.IPTW, PsiHat.TMLE))
+   Qbar.star <- data.frame(cbind(QbarAW.star, Qbar1W.star, Qbar0W.star))
+   names(Qbar.star)<- c('QbarAW.star', 'Qbar1W.star', 'Qbar0W.star')
+   list(estimates=estimates, Qbar.star=Qbar.star, H.AW=H.AW)
+ }


> out <- run.tmle(ObsData=ObsData, SL.library=SL.library)
> est <- out$estimates
> est*100


  PsiHat.SS PsiHat.IPTW PsiHat.TMLE
1  4.349321   -1.752565   0.3233087
```

The TMLE estimate of $\Psi(\mathbb{P}_0)$ is 0.3%. After controlling for baseline covariates, the marginal difference in the probability of survival among "good guys" and among "bad guys" was 0.003. Under the causal assumptions, the risk of survival is essentially 0% higher if the subject was considered a "good guy".

3 Step 7. Inference and interpret results:

Our goal is not just to generate point estimate of

$$\Psi(\mathbb{P}_0) = \sum_w \left[\bar{Q}_0(A=1, W=w) - \bar{Q}_0(0, W=w)\right]\mathbb{P}_0(W=w)$$

Instead, we also want to quantify the statistical uncertainty in that estimate (e.g. hypothesis testing and confidence intervals). *In the this lab, we will use the sample variance of the estimated influence curve to obtain inference for the TMLE. We will also implement the non-parametric bootstrap for variance estimation for the three classes of estimators.*

3.1 Review of Asymptotic Linearity

An estimator $\hat{\Psi}(\mathbb{P}_n)$ of $\Psi(\mathbb{P}_0)$ is asymptotically linear with influence curve $IC(O_i)$ if

$$\sqrt{n}\Big(\hat{\Psi}(\mathbb{P}_n) - \Psi(\mathbb{P}_0)\Big) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} IC(O_i) + o_P(1)$$

where the remainder term $o_P(1)$ converges to zero in probability (as sample size goes to infinity). The influence curve has mean zero $\mathbb{E}_0(IC) = 0$ and finite variance $Var_0(IC) < \infty$. In words, the estimator $\hat{\Psi}(\mathbb{P}_n)$ minus the truth $\Psi(\mathbb{P}_0)$ can be written as an empirical mean of a function of the observed data (plus a second order term that goes to zero):

$$\hat{\Psi}(\mathbb{P}_n) - \Psi(\mathbb{P}_0) = \frac{1}{n}\sum_{i=1}^{n} IC(O_i) + o_P(1/\sqrt{n})$$

As a result, the estimator is **consistent**; as sample size goes to infinity, the estimator converges (in probability) to the estimand. The estimator is also **asymptotically normal**:

$$\sqrt{n}\Big(\hat{\Psi}(\mathbb{P}_n) - \Psi(\mathbb{P}_0)\Big) \to^D \; N\big(0, Var(IC)\big)$$

Thereby, a robust approach to estimating the variance of an asymptotically linear estimator $\hat{\Psi}(\mathbb{P}_n)$ is the sample variance of the estimated influence curve, divided by n.

3.2 Obtaining Inference for TMLE with Influence Curves

- TMLE is consistent if either the conditional mean function $\bar{Q}_0(A,W)$ or the treatment mechanism $g_0(A|W)$ is estimated consistently.
- TMLE is asymptotically linear under stronger conditions, detailed on page 96 of *Targeted Learning*. Also see Mark's HAL-TMLE lecture for a 1pg proof on the asymptotic linearity of the TMLE and on how to get rid of these conditions.
- The influence curve for TMLE at the true data generating distribution $\mathbb{P}_0$ is given by

$$IC(O_i) = \left(\frac{\mathbb{I}(A_i=1)}{g_0(A_i=1|W)} - \frac{\mathbb{I}(A_i=0)}{g_0(A_i=0|W)}\right)\left[Y_i - \bar{Q}_0(A_i,W_i)\right] + \bar{Q}_0(1,W_i) - \bar{Q}_0(0,W_i) - \Psi(\mathbb{P}_0)$$

  This is a function of the unit data O_i and $\mathbb{P}_0$ (unknown). However, we have estimated the relevant pieces:

  - the clever covariate: $\hat{H}(A_i,W_i) = \left(\frac{\mathbb{I}(A_i=1)}{\hat{g}(A_i=1|W)} - \frac{\mathbb{I}(A_i=0)}{\hat{g}(A_i=0|W)}\right)$
  - the residual, which is the observed outcome minus the targeted prediction: $\left(Y_i - \hat{\bar{Q}}^*(A_i,W_i)\right)$
  - the difference in the targeted predictions given $A=1$ and given $A=0$: $\hat{\bar{Q}}^*(1,W_i) - \hat{\bar{Q}}^*(0,W_i)$
  - the target parameter: $\hat{\Psi}(\hat{\mathbb{P}})$

- Therefore, we estimate the variance of the TMLE with the sample variance of the estimated influence curve, scaled by sample size:

$$\hat{\sigma}^2 = Var(\hat{IC})/n$$

3.3 Estimate the variance of the TMLE

1. For all observations, calculate the influence curve.

```
> IC <- H.AW*(ObsData$Y - QbarAW.star) + Qbar1W.star - Qbar0W.star - PsiHat.TMLE
```

2. Take the sample variance of `IC`, divide by `n`, and take the square-root to obtain an estimate of the standard error $\hat{\sigma}$.

3. Now we can calculate confidence intervals based on the standard normal distribution:

$$\hat{\Psi}_{TMLE}(\hat{\mathbb{P}}) \pm 1.96\ \hat{\sigma}$$

4. We can also conduct tests of hypotheses. For example, let the null hypothesis be no effect $H_0 : \psi_0 = 0$. Then the p-value for a two sided test can be calculated as

$$pvalue = 2\text{Prob}\left(Z \geq \left|\frac{\hat{\Psi}_{TMLE}(\hat{\mathbb{P}}) - \psi_0}{\hat{\sigma}}\right|\right)$$

   where $Z \sim N(0,1)$.
   Hint: use the `pnorm` function and specify `lower.tail=F`.

Solution:

```
> #  point estimate
> PsiHat.TMLE <- est$PsiHat.TMLE
> # evaluate the influence curve for all observations
> H.AW <- out$H.AW
> QbarAW.star <- out$Qbar.star[,'QbarAW.star']
> Qbar1W.star <- out$Qbar.star[,'Qbar1W.star']
> Qbar0W.star <- out$Qbar.star[,'Qbar0W.star']
> IC <- H.AW*(ObsData$Y - QbarAW.star) + Qbar1W.star - Qbar0W.star - PsiHat.TMLE
> summary(IC)


       V1
 Min.   :-20.29974
 1st Qu.: -0.25102
 Median :  0.05805
 Mean   :  0.00000
 3rd Qu.:  0.26612
 Max.   : 25.86237


> # estimate sigma^2 with the variance of the IC divided by n
> varHat.IC<-var(IC)/n
> varHat.IC


             [,1]
[1,] 0.0008203395


> # standard error estimate
> se <- sqrt(varHat.IC)
> se


           [,1]
[1,] 0.02864157


> # obtain 95% two-sided confidence intervals:
> alpha <- 0.05
> c(PsiHat.TMLE+qnorm(alpha/2, lower.tail=T)*se,
+  PsiHat.TMLE+qnorm(alpha/2, lower.tail=F)*se)


[1] -0.05290336  0.05936953


> # calculate the pvalue
> 2* pnorm( abs(PsiHat.TMLE /se), lower.tail=F )


          [,1]
[1,] 0.9101249
```

For this data generating process, the true value of the statistical estimand $\psi_0 = 0$. (See the Appendix.) Under the causal assumptions, there is no effect "being a good guy" on the risk of mortality. Our point estimate from TMLE was 0.3% with 95% confidence intervals [-5.3%, 5.9%]. The two-sided p-value was 0.91.

3.4 Checking 95% Confidence Interval Coverage and Type I Error Rates

In this data generating process, the true value of the target parameter $\psi_0 = 0$. We can check the coverage of the 95% confidence intervals as well as the Type I error rates by (i) drawing an independent sample of size n from $\mathbb{P}_0$, (ii) implementing the estimator (obtaining a point estimate and variance estimate), (iii) calculating the 95% confidence interval, (iv) testing the null hypothesis of no effect at an $\alpha = 0.05$ significance level, (v) repeating this process many times. The proportion of confidence intervals that contain the true value ψ_0 provides an estimate of the confidence interval coverage. The proportion of the tests, where the null hypothesis was *falsely* rejected, provides an estimate of the type I error rate.

1. Set the true value to $\psi_0 = 0$, the number of observations n to 2500 and the number of iterations R to 5 (to start).
2. Create 3 empty vectors of size R:
   - `pt.est` for the point estimates
   - `ci.cov` for the 95% confidence interval coverage
   - `reject` as indicator if the null hypothesis of no effect would be rejected at the $\alpha = 0.05$ level.
3. For R repetitions do the following,
   (a) Draw a new sample using the `generateData` function, which is given in Appendix A.
   ```
   > NewData<- generateData(n, effect=F, get.psi.F=F)
   ```
   (b) Use your own code or the `ltmle` package to calculate the point estimate, create confidence intervals, and calculate the p-value to test the null hypothesis of no effect.
      i. Save the point estimate as an element in vector `pt.est`.
      ii. Determine whether the calculated confidence interval contains the true value and save this indicator (true/false) as an element in vector `ci.cov`:
      iii. Determine whether the null hypothesis was rejected at $\alpha = 0.05$ significance level and save this indicator (true/false) as an element in vector `reject`.
4. When you are confident that your code is working, increase the number of iterations `R=500` and rerun your code. (This may take a long time.)
5. Create a histogram of the point estimates.
6. What proportion of calculated confidence intervals contain the true value? What proportion of tests were falsely rejected?

Solution:

```
> library('ltmle')
> # set the true value to 0
> Psi.P0 <- 0
> # set the number of observations
> n <- 2500
> # set the number of iterations
> R <- 500
> # create the empty vectors
> pt.est<- ci.cov<- reject<- rep(NA, R)
> #  the for loop
> for(r in 1:R){
+   # draw a new sample
```

```
+   NewData<- generateData(n=n, effect=F, get.psi.F=F)
+
+   # run the tmle package
+   out <- ltmle(data=NewData, Anodes='A', Ynodes='Y', abar=list(1,0),
+                       SL.library=SL.library, estimate.time=F)
+   out <- summary(out)$effect.measures$ATE
+   pt.est[r]<- out$estimate
+   ci.cov[r]<- out$CI[1]<= Psi.P0 & Psi.P0 <= out$CI[2]
+   reject[r]<-out$pvalue< 0.05
+
+   # keep track of the iteractions completed
+   print(r)
+ }
```

```
> # create pdf of the histogram of point estimates
> pdf(file="RLab6_hist_tmle.pdf")
> hist(pt.est)
> dev.off()
```

```
> # Confidence interval coverage
> mean(ci.cov)
```

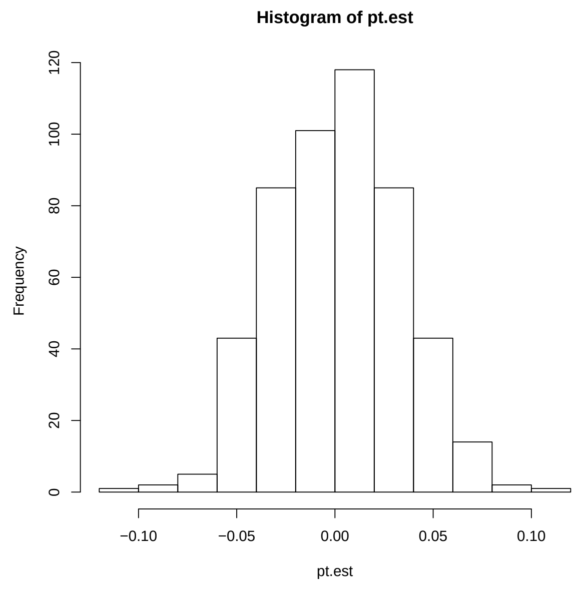
```
[1] 0.938
```

Over $R = 500$ simulated data sets of size $n = 2500$, the 94% of the calculated confidence intervals (based on the sample variance of the estimated influence curve and assuming normality) contained the true parameter value of 0.

```
> # Type I error rate -
> mean(reject)
```

```
[1] 0.062
```

The null hypothesis was falsely reject in 6.2% of the $R = 500$ simulated data sets of size $n = 2500$.



Solution Fig. 1: Histogram of point estimates.

4 The non-parametric bootstrap for variance estimation

In most settings, we do not know the true distribution of the observed data $\mathbb{P}_0$. Instead, we have a single sample of $O_i,\ i = 1, \dots, n$, drawn from $\mathbb{P}_0$. Non-parametric bootstrap approximates resampling from $\mathbb{P}_0$ by resampling from the empirical distribution $\hat{\mathbb{P}}$. The specific steps are

1. Generate a single bootstrap sample by sampling *with replacement* n times from the original sample. This puts a weight of $1/n$ on each resampled observation.
2. Apply our estimator to the bootstrap sample to obtain a point estimate.
3. Repeat this process B times. This gives us an estimate of the distribution of our estimator.
4. Estimate the variance of the estimator across the B bootstrap samples:

   $$\hat{\sigma}^2_{Boot} = \frac{1}{B} \sum_{b=1}^{B} \left(\hat{\Psi}(\hat{\mathbb{P}}^b) - \bar{\hat{\Psi}}(\hat{\mathbb{P}}^b) \right)^2$$

   where $\mathbb{P}^b_n$ is the b^{th} bootstrap sample from the empirical distribution $\hat{\mathbb{P}}$ and $\bar{\hat{\Psi}}(\hat{\mathbb{P}}^b)$ is the average of the point estimates across the bootstrapped samples.
5. Assuming a normal distribution, a 95% confidence interval is

   $$\hat{\Psi}(\hat{\mathbb{P}}) \pm 1.96\ \hat{\sigma}_{Boot}$$

   Alternatively, we can use the 2.5% and 97.5% quantiles of the bootstrap distribution.

Note: Theory supporting the use of the non-parametric bootstrap relies on (1) the estimator being asymptotically linear at $\mathbb{P}_0$ and (2) the estimator not changing behavior drastically if sample from a distribution $\hat{\mathbb{P}}$ near $\mathbb{P}_0$.

4.1 Implement the non-parametric bootstrap for variance estimation

1. Let `B` be the number of bootstrap samples. When writing the code, set `B` to 5. Then after we are sure the code is working properly, increase `B` to 500.

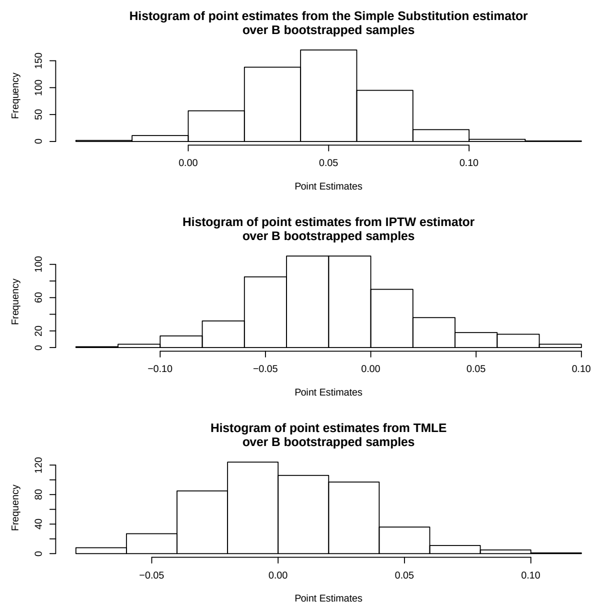
2. Create data frame `estimates` as an empty matrix with `B` rows by `3` columns.
3. Repeat the following `B` times:
   (a) Create bootstrap sample `bootData` by sampling with replacement from the observed data. First, `sample` the indices $1, \dots, n$ with replacement. Then assign the observed data from the re-sampled subjects to `bootData`.
   ```
   > bootIndices<- sample(1:n, replace=T)
   > bootData<- ObsData[bootIndices,]
   ```
   (b) Estimate the G-computation identifiability result (equal to the ATE under the needed assumptions) using the simple substitution estimator, IPTW and TMLE.
   Hint: Copy the relevant code from the previous section, but be sure to change the `ObsData` to `bootData` where appropriate.
   (c) Save the resulting point estimates as row `b` in matrix `estimates`.
4. When you are confident that your code is working, increase the number of bootstrapped samples `B` and rerun your code. Note: creating `B=500` bootstrapped and running the estimators can take a long time.
5. Explore the bootstrapped estimates with `summary`. Create histograms of the estimates.
6. Then assuming a normal distribution, compute the 95% confidence interval for the point estimates.
7. Finally use the `quantiles` function to obtain the 2.5% and 97.5% quantiles of the bootstrap distribution and to compute the 95% confidence interval for the point estimates.

Solution:

```
> #################
> # NP-Boot for B=500 bootstrapped samples
> ###################
> SL.library<- c("SL.glm", "SL.step", "SL.glm.interaction")
> # number of bootstrap samples
> B=500
> # data frame for estimates based on the boot strap sample
> estimates<- data.frame(matrix(NA, nrow=B, ncol=3))
> # for loop from b=1 to total number of bootstrap samples
> for(b in 1:B){
+
+   # sample the indices 1 to n with replacement
+   bootIndices<- sample(1:n, replace=T)
+   bootData<- ObsData[bootIndices,]
+
+   # calling the above function
+   estimates[b,]<- run.tmle(ObsData=bootData, SL.library=SL.library)$estimates
+
+   # keep track of the iteractions completed
+   print(b)
+ }
> colnames(estimates)<-c("SimpSubs", "IPTW", "TMLE")
> save(estimates, file='RLab6_boot.Rdata')
```

```
> #---------------------------------
> # Explore the bootstrapped point estimates
> #---------------------------------
> summary(estimates)

    SimpSubs              IPTW                 TMLE
 Min.   :-0.02846   Min.   :-0.121594   Min.   :-0.077137
 1st Qu.: 0.02990   1st Qu.:-0.041472   1st Qu.:-0.019286
 Median : 0.04496   Median :-0.019044   Median : 0.001306
 Mean   : 0.04443   Mean   :-0.017327   Mean   : 0.002585
 3rd Qu.: 0.05948   3rd Qu.: 0.003805   3rd Qu.: 0.023893
 Max.   : 0.12121   Max.   : 0.090773   Max.   : 0.105517

> #saving the histograms as a pdf
> pdf(file="RLab6_hist_boot.pdf")
> par(mfrow=c(3,1))
> hist(estimates[,1], main="Histogram of point estimates from the Simple Substitution estimator
+ over B bootstrapped samples", xlab="Point Estimates")
> hist(estimates[,2], main="Histogram of point estimates from IPTW estimator
+ over B bootstrapped samples",  xlab="Point Estimates")
> hist(estimates[,3], main="Histogram of point estimates from TMLE
+ over B bootstrapped samples",  xlab="Point Estimates")
> dev.off()

> #---------------------------------
> # 95% Confidence intervals
> #---------------------------------
> create.CI <- function(pt, boot, alpha=0.05){
+   Zquant <- qnorm(alpha/2, lower.tail=F)
+   CI.normal <- c(pt - Zquant*sd(boot), pt + Zquant*sd(boot) )
+   CI.quant  <- quantile(boot, prob=c(0.025,0.975) )
+   out<- data.frame(rbind(CI.normal, CI.quant))*100
+   colnames(out)<- c('CI.lo', 'CI.hi')
+   out
+ }
> # CI for SS assuming a normal dist & then using quantiles
> create.CI(pt=est$PsiHat.SS, boot=estimates[,"SimpSubs"])

                CI.lo    CI.hi
CI.normal -0.10397238 8.802615
CI.quant  -0.02707204 9.153535

> # CI for IPTW assuming normal & also using quantiles
> create.CI(pt=est$PsiHat.IPTW, boot=estimates[,"IPTW"])

              CI.lo    CI.hi
CI.normal -8.997483 5.492353
CI.quant  -8.709283 6.593722

> # CI for TMLE assuming normal & also using quantiles
> create.CI(pt=est$PsiHat.TMLE, boot=estimates[,"TMLE"])
```

```
              CI.lo    CI.hi
CI.normal -5.685597 6.332214
CI.quant  -5.120778 6.541158
```



Solution Fig. 2: Histograms of the point estimates obtained from the simple substitution estimator, IPTW, and TMLE using $B = 500$ bootstrapped samples... If the distribution are highly non-normal, be concerned. The estimator, itself, might be non-normal. The bootstrap may not be doing a good job approximating the true sampling distribution of the estimator. Using 2.5% and 97.5% quantiles is providing the desired coverage under the bootstrap distribution.

5 Concluding Remarks

- Valid statistical inference using both influence curves and the non-parametric bootstrap relies on using an asymptotically linear estimator. The estimator must converge to a normal limit and bias must go to 0 at rate faster than $1/\sqrt{n}$.
- There is no theory guaranteeing that the simple substitution estimator using Super Learner is asymptotically linear (or even has a limit distribution).
- Statistical inference for an IPTW estimator can based on the non-parametric bootstrap or on an estimate of the influence curve of the IPTW estimator. Using the skills learned in this lab, you can implement an influence curve based estimator for the variance of the IPTW estimator yourself. Alternatively, for the modified Horvitz-Thompson (H-T) estimator, which can be implemented by fitting a weighted regression, the robust sandwich estimator will provide an influence curve based variance estimate. (Therefore, both a point estimate and inference for the modified H-T estimator can be obtained with standard software.) If the treatment mechanism $g_0(A|W)$ was estimated with a correctly specified parametric model, the resulting standard error estimates will be conservative. However, if the weights (i.e. the treatment mechanism) were estimated using Super Learner, there is no guarantee that standard software is conservative or that the estimator satisfies the conditions for the non-parametric bootstrap (e.g. asymptotic linearity).
- TMLE requires estimation of both the conditional mean outcome $\bar{Q}_0(A, W)$ and the treatment mechanism $g_0(A|W)$. If the treatment mechanism $g_0(A|W)$ is consistently estimated, TMLE will be asymptotically linear with variance *conservatively* approximated by the sample variance of the estimated influence curve

$\hat{IC}(\bar{Q}_n^*, g_n)$ divided by n. If both are consistently estimated, TMLE will be efficient and achieve the lowest asymptotic variance possible among a large class of regular estimators... See Mark's Lecture for additional details.

Appendix: A specific data generating experiment

The following code was used to generate the data set `RLab6.Inference.csv`. In this data generating process (one of many compatible with the SCM $\mathcal{M}^{\mathcal{F}}$), all exogenous errors are independent. The causal risk difference $\Psi^{\mathcal{F}}(\mathbb{P}_{U,X})$ is 0. The counterfactual probability of survival would be 0% higher if all subjects were "good guys" than if none were.

```
> #-----------------------------------------
> # 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=T, get.psi.F=F){
+
+   W1 <- rbinom(n, size=1, prob=0.5)
+   W2 <- rbinom(n, size=1, prob=0.5)
+   W3 <- runif(n, min=0, max=1)
+   W4 <- runif(n, min=0, max=5)
+
+   pscore <- plogis(1+2*W1*W2-W4)
+   A<- rbinom(n, size=1, prob=pscore)
+
+   U.Y<- runif(n,0,1)
+   # 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)
+   if(!get.psi.F){
+     # if not getting the true value of the causal parameter
+     data<- subset(data, select=c(W1,W2,W3,W4,A,Y) )
+   }
+   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){
+  prob <- plogis(-1.5+A-2*W3+0.5*W4+5*W1*W2*W4)
+  as.numeric(U.Y < prob)
+ }
```

Solution:

```
> #------------
> # Creation of RLab6.Inference.csv
> #-----
> set.seed(252)
> ObsData<- generateData(n=2500, effect=F, get.psi.F=F)
> write.csv(ObsData, file="RLab6.Inference.csv", row.names=F)
> #------------
```

```
> #----------------------------------------
> #  Evaluate the causal parameter by drawing a huge number of observations and
> # taking the difference in the means of the counterfactual outcomes.
> set.seed(252)
> TrueData<- generateData(n=100000, effect=F, get.psi.F=T)
> # Simpy take the difference in mean the counterfactuals
> Psi.F<- mean(TrueData$Y.1-TrueData$Y.0)
> Psi.F
```

```
[1] 0
```