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.

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: intelligence (1 for smart; 0 for not)
- W3: handy/inventiveness (scale from 0 for none to 1 for MacGyver)
- W4: running speed (continuous measure from 0 to 5)
- 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 $\Psi(P_0) = E_0\big[\bar{Q}_0(1,W) - \bar{Q}_0(0,W)\big]$

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`.

2. Use SuperLearner to estimate $E_0(Y|A,W) = \bar{Q}_0(A,W)$, which is the conditional probability of surviving given the exposure (being a good guy) and baseline covariates. Specify the SuperLearner library with the following algorithms: `SL.glm`, `SL.step` and `SL.glm.interaction`. In practice, you will probably want to include more fun/creative algorithms.

3. Use SuperLearner to estimate the treatment mechanism $g_0(A=1|W) = P_0(A=1|W)$, which is the conditional probability of the exposure, given baseline covariates.
Hint: There are no transformed covariates to create.

4. Use these estimates to create the clever covariate:

$$H_n(A,W) = \left(\frac{\mathbb{I}(A=1)}{g_n(A=1|W)} - \frac{\mathbb{I}(A=0)}{g_n(A=0|W)}\right)$$

Calculate `H.AW` for each subject based on his/her observed exposure. Calculate `H.1W` and `H.0W` based on a set exposure.

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

5. Update the initial estimates.

   (a) Run logistic regression of the outcome Y on the clever covariate $H_n(A,W)$, 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 `eps` denote the resulting maximum likelihood estimate of the coefficient on the clever covariate `H.AW`.

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

6. Substitute the updated fits into the target parameter mapping:

$$\hat{\Psi}_{TMLE}(P_n) = \frac{1}{n}\sum_{i=1}^{n}\left[\bar{Q}_n^*(1,W_i) - \bar{Q}_n^*(0,W_i)\right]$$

Solution:

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

[1] 5000
```

```
> #-----------------
> #  Estimate Q_0(A,W) with Superlearner
> #---------------------
> library("SuperLearner")
> # specify the library
> SL.library<- c("SL.glm", "SL.step", "SL.glm.interaction")


> # data frame X with baseline covariates and exposure
> X<-subset(ObsData, select=c(A, W1, W2, W3, W4))
> # create data frames with A=1 and A=0
> X1 <- X0<-X
> X1$A<-1
> X0$A<- 0
> # create newdata by stacking
> newdata<- rbind(X,X1,X0)


> # call superlearner
> Qinit<- SuperLearner(Y=ObsData$Y, X=X, newX=newdata, SL.library=SL.library,
+   family="binomial")
> Qinit


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




                             Risk         Coef
SL.glm_All             0.1614136 0.0000000000
SL.step_All            0.1614136 0.0007246263
SL.glm.interaction_All 0.1392227 0.9992753737


> # pred prob of survival given A,W
> QbarAW <- Qinit$SL.predict[1:n]
> # predicted probability of survival for each subject given A=1 and W
> Qbar1W <- Qinit$SL.predict[(n+1): (2*n)]
> # predicted probability of survival for each subject given A=0 and W
> Qbar0W <- Qinit$SL.predict[(2*n+1): (3*n)]


> # note the simple substitution estimator would be
> PsiHat.SS<-mean(Qbar1W - Qbar0W)
> PsiHat.SS


[1] 0.05327646


> ##########
> # 2.  Estimate g_0(A|W) with SuperLearner
> ############
> # creating data frame with only baseline cov
> W<- subset(ObsData, select= -c(A,Y))
> # call superlearner
> gHatSL<- SuperLearner(Y=ObsData$A, X=W, SL.library=SL.library, family="binomial")
> gHatSL

Call:
SuperLearner(Y = ObsData$A, X = W, family = "binomial", SL.library = SL.library)



                             Risk       Coef
SL.glm_All             0.1547378 0.03184716
SL.step_All            0.1548556 0.00000000
SL.glm.interaction_All 0.1486269 0.96815284

> # generate the predicted prob of being a good guy, given baseline cov
> gHat1W<- gHatSL$SL.predict
> # generate the predicted prob of not being a good guy, given baseline cov
> gHat0W<- 1- gHat1W

> # summary of propensity scores
> summary(gHat1W)

       V1
 Min.   :0.01110
 1st Qu.:0.08839
 Median :0.27992
 Mean   :0.34960
 3rd Qu.:0.58753
 Max.   :0.96706

> summary(gHat0W)

       V1
 Min.   :0.03294
 1st Qu.:0.41247
 Median :0.72008
 Mean   :0.65040
 3rd Qu.:0.91161
 Max.   :0.98890

> #IPTW estimator of the ATE is
> PsiHat.IPTW<- mean(as.numeric(ObsData$A==1)*ObsData$Y/gHat1W) -
+   mean(as.numeric(ObsData$A==0)*ObsData$Y/gHat0W)
> PsiHat.IPTW

[1] -0.01530971

> #------------
> # 3. Create the clever covariate H_n^*(A,W)$ for each subject
> # numerator is the indicator of the obsv txt.
> # denominator is the predicted probability of observed exp, given baseline cov
> #--------------
> H.AW<- as.numeric(ObsData$A==1)/gHat1W - as.numeric(ObsData$A==0)/gHat0W
> # equiv: H.AW<- (2*ObsData$A-1)/ gHatAW
>
> # also want to evaluate the clever covariates at A=1 and A=0 for all subjects
> H.1W<- 1/gHat1W
> H.0W<- -1/gHat0W

> #---------
> # 4. Update the initial estimate of Qbar_0(A,W)
> # logit( Qbar_n^0(eps) )= logit( Qbar_n^0) + epsHAW
> # run logistic regression of Y on H.AW using the logit of initQbarAW.predict as offset
> #------------
> logitUpdate<- glm(ObsData$Y ~ -1 +offset(qlogis(QbarAW)) + H.AW, family='binomial')
> eps<- logitUpdate$coef
> eps

       H.AW
-0.02093744

> #  calc the predicted values for each subj under each txt
> QbarAW.star<- plogis(qlogis(QbarAW)+ eps*H.AW)
> Qbar1W.star<- plogis(qlogis(Qbar1W)+ eps*H.1W)
> Qbar0W.star<- plogis(qlogis(Qbar0W)+ eps*H.0W)

> # since the clever cov is not changing, updating will not have any effect
> coef(glm(ObsData$Y ~ -1 +offset(qlogis(QbarAW.star)) + H.AW, family=binomial))

        H.AW
1.630499e-17

> # 5. Estimate Psi(P_0) as the emp mean of the difference in the pred
> # outcomes under A=1 and A=0
> PsiHat.TMLE<- mean(Qbar1W.star) - mean(Qbar0W.star)

> # comparing the estimates...
> c(PsiHat.SS, PsiHat.IPTW, PsiHat.TMLE)

[1]  0.053276457 -0.015309711  0.008009102
```

The TMLE estimate of $\Psi(P_0)$ is 0.8%. After controlling for baseline covariates, the marginal difference in the probability of survival among “good guys” and among “bad guys” was 0.008. Under the causal assumptions, the risk of survival is essentially 0% higher if the subject was considered a “good guy”.

```
> # in terms of percents
> c(PsiHat.SS, PsiHat.IPTW, PsiHat.TMLE)*100

[1]  5.3276457 -1.5309711  0.8009102
```

3 Step 7. Inference and interpret results:

Our goal is not just to generate point estimate of

$$\Psi(P_0) = E_0[E_0(Y|A=1,W) - E_0(Y|A=0,W)]$$

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 $\hat{\Psi}(P_n)$ of $\Psi(P_0)$ is asymptotically linear with influence curve $IC(O_i)$ if

$$\sqrt{n}\Big(\hat{\Psi}(P_n) - \Psi(P_0)\Big) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} IC(O_i) + o_{P_0}(1)$$

where the remainder term $o_{P_0}(1)$ converges to zero in probability (as sample size goes to infinity). The influence curve has mean zero $E_0(IC) = 0$ and finite variance $Var_0(IC) < \infty$. In words, the estimator $\hat{\Psi}(P_n)$ minus the truth $\Psi(P_0)$ can be written as an empirical mean of a function of the observed data (plus a second order term that goes to zero):

$$\hat{\Psi}(P_n) - \Psi(P_0) = \frac{1}{n}\sum_{i=1}^{n} IC(O_i) + o_{P_0}(1/\sqrt{n})$$

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**:

$$\sqrt{n}\Big(\hat{\Psi}(P_n) - \Psi(P_0)\Big) \rightarrow^{D} N\big(0, Var(IC)\big)$$

Thereby, a robust approach to estimating the variance of an asymptotically linear estimator $\hat{\Psi}(P_n)$ 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 $g_0(A|W)$ are estimated consistently.
- TMLE is asymptotically linear under stronger conditions. See page 96 of *Targeted Learning* for details. Briefly, if $g_n(A|W)$ is a consistent estimator of the treatment mechanism $g_0(A|W)$, then TMLE is asymptotically linear with influence curve, that is *conservatively* approximated by the efficient influence curve at the possibly misspecified limit $\bar{Q}^*(A,W)$ and $g_0(A|W)$.
- The influence curve for observation O_i at the true data generating distribution P_0 is

$$IC(O_i) = \left(\frac{\mathbb{I}(A_i=1)}{g_0(A_i=1|W)} - \frac{\mathbb{I}(A_i=0)}{g_0(A_i=0|W)}\right)\big(Y_i - \bar{Q}_0(A_i,W_i)\big) + \bar{Q}_0(1,W_i) - \bar{Q}_0(0,W_i) - \Psi(P_0)$$

  This is a function of the unit data O_i and P_0 (unknown). However, we have estimated the relevant pieces:

  - the clever covariate $H_n(A_i,W_i) = \left(\frac{\mathbb{I}(A_i=1)}{g_n(A_i=1|W)} - \frac{\mathbb{I}(A_i=0)}{g_n(A_i=0|W)}\right)$
  - the residual, which is the observed outcome minus the predicted probability: $\big(Y_i - \bar{Q}_n^*(A_i,W_i)\big)$

- the difference in the predicted probability of surviving under $A = 1$ and the predicted probability of surviving under $A = 0$:

$$\bar{Q}_n^*(1, W_i) - \bar{Q}_n^*(0, W_i)$$

- the target parameter: $\hat{\Psi}(P_n)$

- Under the above assumptions, the sample variance of the estimated influence curve gives us a conservative estimate of the asymptotic variance of the standardized TMLE. Dividing the sample variance by n gives us an estimate of the finite sample variance of the estimator.

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

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` and divide by `n` to obtain an estimate of σ^2/n. We denote this estimate of the standard error $\hat{\sigma}$.

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

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

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

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

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

Solution:

```
> # recall point estimate
> PsiHat.TMLE

[1] 0.008009102

> # evaluate the influence curve for all observations
> IC <- H.AW*(ObsData$Y - QbarAW.star) + Qbar1W.star - Qbar0W.star - PsiHat.TMLE
> summary(IC)

       V1
 Min.   :-46.39176
 1st Qu.: -0.27459
 Median :  0.03241
 Mean   :  0.00000
 3rd Qu.:  0.30310
 Max.   : 20.47199
```

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


             [,1]
[1,] 0.0004098322


> # obtain confidence intervals:
> c(PsiHat.TMLE  -1.96*sqrt(varHat.IC),  PsiHat.TMLE  +1.96*sqrt(varHat.IC))


[1] -0.03166975  0.04768795


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


          [,1]
[1,] 0.6923836
```

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 0.8% with 95% confidence intervals [-3.2%, 4.8%]. The two-sided p-value was 0.69. Inference was based on the sample variance of the estimated influence curve.

```
> #-----------
> # using the tmle package
> #------------
> library("tmle")
> # using the initial estimates for the conditional mean outcome and treatment mech
> tmle.QGiven.gGiven<- tmle(Y=ObsData$Y, A=ObsData$A, W=W,
+     Q=cbind(Qbar0W, Qbar1W),g1W=gHat1W, family="binomial")
> tmle.QGiven.gGiven$estimates$ATE


$psi
[1] 0.008850164

$var.psi
             [,1]
[1,] 0.0003442739

$CI
[1] -0.02751689  0.04521722

$pvalue
         [,1]
[1,] 0.633377
```

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 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. Load the `R` package `tmle`.
2. Set the true value to $\psi_0 = 0$, the number of observations n to 5,000 and the number of iterations R to 5 (to start).
3. Create 3 empty vectors of size R:
   - `pt.est` for the point estimates
   - `ci.cov` for the confidence interval coverage
   - `reject` as indicator if the null hypothesis of no effect would be rejected at the $\alpha = 0.05$ level.
4. Within a for loop from `r in 1:R`, do the following:
   (a) Draw a new sample using the `generateData` function, which is given below.
   ```
   > NewData<- generateData(n)
   ```
   (b) Create a data frame `W` of the baseline covariates.
   (c) Call the `tmle` package. Obtain initial estimates of $\bar{Q}_0(A, W)$ and $g_0(A|W)$ with SuperLearner, using the default library.
   ```
   > out<- tmle(Y=NewData$Y, A=NewData$A, W=W, family='binomial')
   ```
   (d) Save the point estimate as an element in vector `pt.est`.
   ```
   > pt.est[r]<-out$estimates$ATE$psi
   ```
   (e) Determine whether the calculated confidence interval contains the true value and save this indicator (true/false) as an element in vector `ci.cov`:
   ```
   > ci.cov[r]<-out$estimates$ATE$CI[1]<= Psi.P0 & Psi.P0 <= out$estimates$ATE$CI[2]
   ```
   (f) 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`:
   ```
   > reject[r]<-out$estimates$ATE$pvalue< 0.05
   ```
5. 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.)
6. Create a histogram of the point estimates.
7. What proportion of calculated confidence intervals contain the true value? What proportion of tests were falsely rejected?

```
> generateData<- function(n){
+   W1 <- rbinom(n, size=1, prob=0.5) #male
+   W2 <- rbinom(n, size=1, prob=0.5) #smart
+   W3 <- runif(n, min=0, max=1) # MacGyver
+   W4 <- runif(n, min=0, max=5) #running speed
+   A<- rbinom(n, size=1, prob= plogis(1+2*W1*W2-W4))
+   Y<- rbinom(n, size=1, prob=  plogis(-1.5+0*A-2*W3+0.5*W4+5*W1*W2*W4))
+
```

```
+   # counterfactual
+   Y.1<- rbinom(n, size=1, prob= plogis(-1.5+0*1-2*W3+0.5*W4+5*W1*W2*W4))
+   Y.0<- rbinom(n, size=1, prob= plogis(-1.5+0*0-2*W3+0.5*W4+5*W1*W2*W4))
+
+   # return data.frame
+   data.frame(W1,W2,W3,W4,A,Y,Y.1,Y.0)
+ }
```

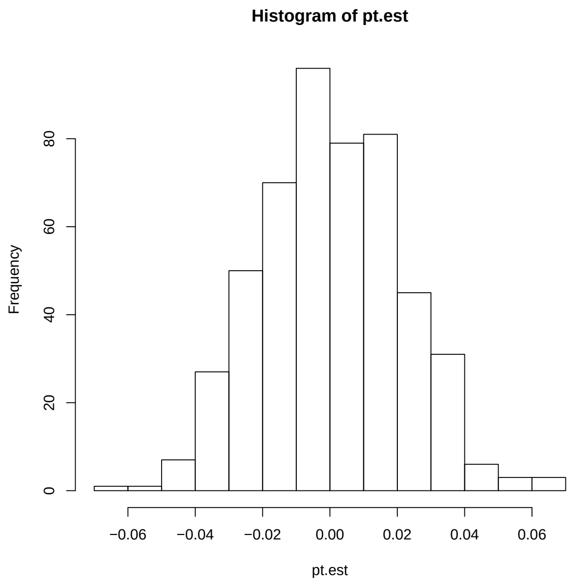
Solution:

```
> library('tmle')
> # set the true value to 0
> Psi.P0 = 0
> # set the number of observations
> n=5000
> # 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)
+
+   # create a data frame of baseline covariates
+   W<- subset(NewData, select=c(W1,W2,W3,W4))
+
+   # run the tmle package
+   out<- tmle(Y=NewData$Y, A=NewData$A, W=W, family='binomial')
+   pt.est[r]<-out$estimates$ATE$psi
+   ci.cov[r]<-out$estimates$ATE$CI[1]<= Psi.P0 & Psi.P0 <= out$estimates$ATE$CI[2]
+   reject[r]<-out$estimates$ATE$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 $n = 5,000$, 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 $n = 5,000$.



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 P_0. Instead, we have a single sample of $O_i,\ i = 1, \ldots, n$, drawn from P_0. Non-parametric bootstrap approximates resampling from P_0 by resampling from the empirical distribution P_n. 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:

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

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

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

   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 P_0 and (2) the estimator not changing behavior drastically if sample from a distribution P_n near 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. Within a `for` loop from `b in 1:B`,
   (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 average treatment effect 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 estimates `PsiHat.SS.b`, `PsiHat.IPTW.b` and `PsiHat.TMLE.b` as row `b` in matrix `estimates.` We are appending `.b` in order to distinguish the point estimates from the observed data and the point estimates from the bootstrapped samples.
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` and `colMeans`. Create histograms of the estimates. For the three estimators, get the sample variance of the point estimates over the `B` bootstrapped samples and save the estimates as `varHat.SS`, `varHat.IPTW` and `varHat.TMLE`, respectively.
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.

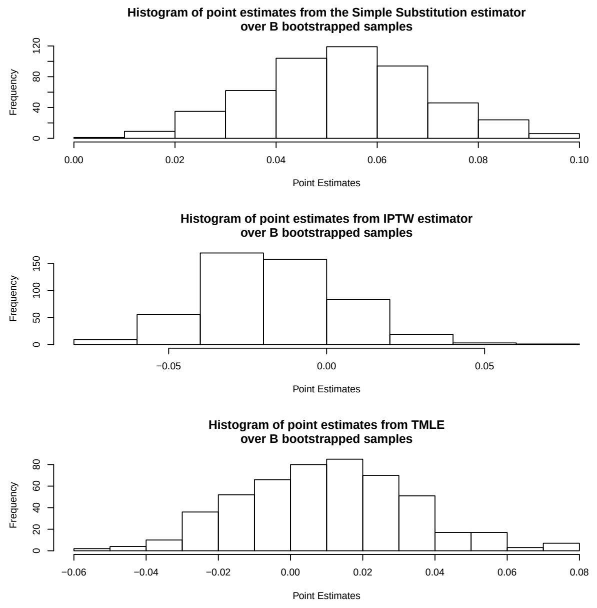
Solution:

```
> #################
> # NP-Boot for B=500 bootstrapped samples
> ###################
> # 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,]
+
+   # data frame X with baseline cov and intervention
```

```
+   X<- subset(bootData, select=c(W1, W2, W3, W4, A) )
+
+   # create data frames with A=1 and A=0
+   X1 <- X0<-X
+   X1$A<-1
+   X0$A<- 0
+   # create newdata by stacking
+   newdata<- rbind(X,X1,X0)
+
+   # call superlearner
+   Qinit<- SuperLearner(Y=bootData$Y, X=X, newX=newdata, SL.library=SL.library,
+     family="binomial")
+
+   # pred prob of survival given observed A,W
+   QbarAW <- Qinit$SL.predict[1:n]
+   # predicted probability of survival given A=1 and W
+   Qbar1W <- Qinit$SL.predict[(n+1): (2*n)]
+   # predicted probability of survival given A=0 and W
+   Qbar0W <- Qinit$SL.predict[(2*n+1): (3*n)]
+
+   #  simple substitution estimator
+   PsiHat.SS.b<-mean(Qbar1W - Qbar0W)
+
+   ##########
+   # Estimate g_0(A|W) with SuperLearner
+   ############
+   W<- subset(bootData, select=c(W1,W2,W3,W4))
+
+   # call superlearner
+   gHatSL<- SuperLearner(Y=bootData$A, X=W, SL.library=SL.library, family="binomial")
+
+   # generate the predicted prob of being a good guy, given baseline cov
+   gHat1W<- gHatSL$SL.predict
+   # generate the predicted prob of not being a good guy, given baseline cov
+   gHat0W<- 1- gHat1W
+
+   # the IPTW estimator of the ATE is
+   PsiHat.IPTW.b<- mean(bootData$A*bootData$Y/gHat1W) -
+     mean((1-bootData$A)*bootData$Y/gHat0W)
+
+   ###################
+   # 3. Estimate to create the clever covariate H_n^*(A,W)$ for each subject
+   # numerator is the indicator of the obsv txt.
+   # denominator is the predicted probability of observed exp, given baseline cov
+   ###############
+
+   # since the txt is binary, we can use a short cut
+   H.AW<- bootData$A/gHat1W - (1-bootData$A)/gHat0W
+   # equiv: H.AW<- (2*bootData$A-1)/ gHatAW
+
+   # also want to evaluate the clever covariates at A=1 and A=0 for all women
+   H.1W<- 1/gHat1W
+   H.0W<- -1/gHat0W
```

```
+
+   ####################
+   # 4. Update the initial estimate of Qbar_0(A,W)
+   # logit( Qbar_n^0(eps) )= logit( Qbar_n^0) + epsHAW
+   # run logistic regression of Y on H.AW using the logit of initQbarAW.predict as offset
+   ######################
+   logitUpdate<- glm(bootData$Y ~ -1 +offset(qlogis(QbarAW)) + H.AW, family=binomial)
+   eps<- logitUpdate$coef
+
+   #  calc the predicted values for each subj under each txt
+   QbarAW.star<- plogis(qlogis(QbarAW)+ eps*H.AW)
+   Qbar1W.star<- plogis(qlogis(Qbar1W)+ eps*H.1W)
+   Qbar0W.star<- plogis(qlogis(Qbar0W)+ eps*H.0W)
+
+   # 5. Estimate Psi(P_0) as the emp mean of the difference in the pred
+   # outcomes under A=1 and A=0
+   PsiHat.TMLE.b<- mean(Qbar1W.star - Qbar0W.star)
+
+   # 6. Save the estimates.
+   estimates[b,]<-c(PsiHat.SS.b, PsiHat.IPTW.b, PsiHat.TMLE.b)
+
+   # keep track of the iteractions completed
+   print(b)
+ }
> colnames(estimates)<-c("SimpSubs", "IPTW", "TMLE")


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


    SimpSubs              IPTW                 TMLE
 Min.   :0.003168   Min.   :-0.079003   Min.   :-0.055441
 1st Qu.:0.041647   1st Qu.:-0.030562   1st Qu.:-0.007051
 Median :0.053228   Median :-0.017926   Median : 0.010052
 Mean   :0.053143   Mean   :-0.016539   Mean   : 0.009960
 3rd Qu.:0.064471   3rd Qu.:-0.002543   3rd Qu.: 0.025108
 Max.   :0.098185   Max.   : 0.068028   Max.   : 0.079603


> # get the mean
> colMeans(estimates)


    SimpSubs         IPTW         TMLE
 0.053142560 -0.016538715  0.009959928


> #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()

> ##########
> diag(var(estimates))

    SimpSubs         IPTW         TMLE
0.0002690370 0.0004638349 0.0005523982

> # compare est variance with est. Influence Curve for TMLE
> varHat.IC

             [,1]
[1,] 0.0004098322

> varHat.SS<- var(estimates[,"SimpSubs"])
> varHat.IPTW<- var(estimates[,"IPTW"])
> varHat.TMLE<- var(estimates[,"TMLE"])

> #---------------
> # 95% Confidence intervals
> #-------------
> # CI simple subs estimator assuming a normal dist & then using quantiles
> c(PsiHat.SS-1.96*sqrt(varHat.SS),  PsiHat.SS  +1.96*sqrt(varHat.SS))

[1] 0.02112785 0.08542506

> quantile(estimates[,"SimpSubs"], prob=c(0.025,0.975))

      2.5%      97.5%
0.02119103 0.08477203

> #------------------
> # CI for IPTW assuming normal & also using quantiles
> c(PsiHat.IPTW  -1.96*sqrt(varHat.IPTW),  PsiHat.IPTW +1.96*sqrt(varHat.IPTW))

[1] -0.05752189  0.02690247

> quantile(estimates[,"IPTW"], prob=c(0.025,0.975))

       2.5%       97.5%
-0.05789435  0.03019138

> #------------------
> # CI for TMLE assuming normal & also using quantiles
> c(PsiHat.TMLE  -1.96*sqrt(varHat.TMLE),  PsiHat.TMLE  +1.96*sqrt(varHat.TMLE))
```

```
[1] -0.03805708  0.05407528

> quantile(estimates[,"TMLE"], prob=c(0.025,0.975))

       2.5%       97.5%
-0.03018670  0.05867388

> save(pt.est, ci.cov, reject, estimates, file='RLab6_answers.Rdata')
```



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 SuperLearner 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. You can implement an influence curve based estimator for the variance of the IPTW estimator yourself, as we did here for TMLE. Alternatively, for the modified H-T estimator, which can be implemented by fitting a weighted regression, the robust sandwich estimator (which can be called using standard software) will provide an influence curve based variance estimate. 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.

- TMLE requires estimation of both the conditional mean function $\bar{Q}_0(A, W)$ 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 $IC_n(\bar{Q}^*_n, g_n)$ 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.

Solution:

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 $\mathcal{M}^F$), all exogenous errors are independent. The causal risk difference $\Psi^F(P_{U,X})$ 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
> # output: ObsData + counterfactuals
> #--------
> generateData<- function(n){
+   W1 <- rbinom(n, size=1, prob=0.5) #male
+   W2 <- rbinom(n, size=1, prob=0.5) #smart
+   W3 <- runif(n, min=0, max=1) # MacGyver
+   W4 <- runif(n, min=0, max=5) #running speed
+   A<- rbinom(n, size=1, prob= plogis(1+2*W1*W2-W4))
+   Y<- rbinom(n, size=1, prob=  plogis(-1.5+0*A-2*W3+0.5*W4+5*W1*W2*W4))
+
+   # counterfactual
+   Y.1<- rbinom(n, size=1, prob= plogis(-1.5+0*1-2*W3+0.5*W4+5*W1*W2*W4))
+   Y.0<- rbinom(n, size=1, prob= plogis(-1.5+0*0-2*W3+0.5*W4+5*W1*W2*W4))
+
+   # return data.frame
+   data.frame(W1,W2,W3,W4,A,Y,Y.1,Y.0)
+ }


> #------------
> # Creation of RLab6.Inference.csv
> #-----
> set.seed(252)
> ObsData<- generateData(n=5000)
> ObsData<- subset(ObsData, select=c(W1,W2,W3,W4,A,Y) )
> 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=250000)
> # Simpy take the difference in mean the counterfactuals
> Psi.F<- mean(TrueData$Y.1) - mean(TrueData$Y.0)
> Psi.F

[1] 0.00032
```

Since we know the true conditional mean $\bar{Q}_0(A, W)$, we could evaluate the statistical estimand $\Psi(P_0)$ by drawing a huge number of observations and taking the difference of the mean predicted outcomes under the intervention and the control.

```
> # Recall TrueData consists of 100,000 observations
> Qbar.true<- glm(Y~A+W3+W4+W1:W2:W4, family="binomial", data=TrueData)
> txt<- control <- TrueData
> txt$A=1; control$A=0
> Psi.P0<- mean( predict(Qbar.true, newdata=txt, type="response")) -
+   mean(predict(Qbar.true, newdata=control, type="response"))
> Psi.P0

[1] -0.0003074739
```

If we increased n enough, these values should be identical. The slight difference, here, is due to estimation.