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 $\Psi(P_0) = E_0\left[\bar{Q}_0(1, W) - \bar{Q}_0(0, W)\right]$

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`.

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:

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

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`.

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)
+ }
```

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 $1, \ldots, n$ 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.

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.