R Lab 6 - Inference

Laura B. Balzer

Biostat 683 - Intro. to Causal Inference

Goals:
1. Review estimation based on the simple substitution estimator, inverse probability of treatment weighted (IPTW) estimator, and targeted maximum likelihood estimation (TMLE).
2. Use the sample variance of the estimated 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

- $W1$: age (1 for young; 0 for old)
- $W2$: 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)
- $W4$: martial 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 baseline, adjustment variables.

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

2 Step 6. Estimate $\Psi(\mathbb{P}_0) = \mathbb{E}_0\big[\mathbb{E}_0(Y|A=1,W) - \mathbb{E}_0(Y|A=0,W)\big]$

THIS IS A REVIEW OF R LAB 5. RE-USE YOUR CODE.

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

2. Load the `SuperLearner` package (Polley et al., 2018). 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 flexible libraries.

3. Use Super Learner to estimate $\mathbb{E}_0(Y|A,W) = \mathbb{P}_0(Y=1|A,W)$, which is the conditional probability of surviving given "good guy" status and baseline covariates.

4. Evaluate the simple substitution estimator by plugging the estimates $\hat{\mathbb{E}}(Y|A=1,W)$ and $\hat{\mathbb{E}}(Y|A=0,W)$ into the target parameter mapping:

$$\hat{\Psi}_{SS}(\hat{\mathbb{P}}) = \frac{1}{n}\sum_{i=1}^{n}\Big(\hat{\mathbb{E}}(Y|A=1,W_i) - \hat{\mathbb{E}}(Y|A=0,W_i)\Big)$$

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

6. Use these estimates to create the clever covariate:

$$\hat{H}(A,W) = \left(\frac{\mathbb{I}(A=1)}{\hat{\mathbb{P}}(A=1|W)} - \frac{\mathbb{I}(A=0)}{\hat{\mathbb{P}}(A=0|W)}\right)$$

Calculate `H.AW` for each participant based on their observed exposure. Also evaluate the clever covariate at $A=1$ and $A=0$.

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.

   (b) Use the resulting estimated coefficient $\hat{\epsilon}$ to update the initial estimates of $\hat{\mathbb{E}}(Y|A,W)$, $\hat{\mathbb{E}}(Y|A=1,W)$, and $\hat{\mathbb{E}}(Y|A=0,W)$.

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

$$\hat{\Psi}_{TMLE}(\hat{\mathbb{P}}) = \frac{1}{n}\sum_{i=1}^{n}\Big(\hat{\mathbb{E}}^*(Y|A=1,W_i) - \hat{\mathbb{E}}^*(Y|A=0,W_i)\Big)$$

3 Step 7. Inference and interpret results:

Our goal is not just to generate point estimate; we also want to quantify the statistical uncertainty in that estimate. In other words, for hypothesis testing and confidence interval construction, we need an estimate of our algorithm's variability *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 $\hat{\Psi}(\hat{\mathbb{P}})$ of $\Psi(\mathbb{P}_0)$ is asymptotically linear with influence curve $IC(O)$ if

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

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 $\hat{\Psi}(\hat{\mathbb{P}})$ minus the truth $\Psi(\mathbb{P}_0)$ can be written as an empirical mean of a function of the observed data (plus a term that going to zero in probability):

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

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**:

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

Thereby, a robust approach to estimating the variance of an asymptotically linear estimator $\hat{\Psi}(\hat{\mathbb{P}})$ 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 outcome $\mathbb{E}_0(Y|A,W)$ or the propensity score $\mathbb{P}_0(A = 1|W)$ is estimated consistently.
- TMLE is asymptotically linear under stronger conditions, detailed on page 96 of *Targeted Learning* (van der Laan and Rose, 2011).
- The influence curve for TMLE for observation i at the true data generating distribution $\mathbb{P}_0$ is given by

$$\begin{aligned} IC(O_i) = &\left(\frac{\mathbb{I}(A_i = 1)}{\mathbb{P}_0(A = 1|W_i)} - \frac{\mathbb{I}(A_i = 0)}{\mathbb{P}_0(A = 0|W_i)}\right)\big[Y_i - \mathbb{E}_0(Y|A_i, W_i)\big] \\ &+ \mathbb{E}_0(Y|A = 1, W_i) - \mathbb{E}_0(Y|A = 0, W_i) - \Psi(\mathbb{P}_0) \end{aligned}$$

  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: $\hat{H}(A_i, W_i) = \left(\frac{\mathbb{I}(A_i=1)}{\hat{\mathbb{P}}(A=1|W)} - \frac{\mathbb{I}(A_i=0)}{\hat{\mathbb{P}}(A=0|W)}\right)$
  - the residual, which is the observed outcome minus the targeted prediction: $\big(Y_i - \hat{\mathbb{E}}^*(Y|A_i, W_i)\big)$
  - the difference in the targeted predictions given $A = 1$ and given $A = 0$: $\hat{\mathbb{E}}^*(Y|A_i = 1, W_i) - \hat{\mathbb{E}}^*(Y|A_i = 0, W_i)$
  - 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.
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 95% 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`.

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) implementing a two-sided hypothesis test at the $\alpha = 0.05$ 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 an indicator that the 95% confidence interval included the truth ψ_0
- `reject` for an indicator that the null hypothesis of no association was 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 `RLab6_datagen.R` and in Appendix A.
   ```
   > NewData<- generateData(n, effect=F, get.psi.star=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 of the effect 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 few minutes.)
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?

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 $O_i,\ i = 1, \ldots, n$, drawn from $\mathbb{P}_0$. Non-parametric bootstrap approximates re-sampling from $\mathbb{P}_0$ by re-sampling from the empirical distribution $\hat{\mathbb{P}}$. The specific steps are

1. Generate a single bootstrap dataset by sampling *with replacement* n times from the original sample. This puts a weight of $1/n$ on each re-sampled 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. Take the sample variance of the point estimates from the bootstrap samples: $\hat{\sigma}^2_{Boot}$.
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 we 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 $1, \ldots, n$ with replacement. Then assign the observed data from the re-sampled participants to `bootData`.

   ```
   > bootIndices<- sample(1:n, replace=T)
   > bootData<- ObsData[bootIndices,]
   ```

   (b) Estimate the G-computation identifiability result (equal to the average treatment effect under the needed assumptions) using the simple substitution estimator, IPTW and TMLE.
   Hint: Copy the relevant code from Section-2, but be sure to use `bootData` to obtain a point estimate, instead of the `ObsData`.

   (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 few minutes.

5. Explore the bootstrapped estimates with `summary`. Create histograms of the bootstrapped estimates.

6. Then assuming a normal distribution, compute the 95% confidence interval for each algorithm.

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 each algorithm.

5 Concluding Remarks

- Valid statistical inference using both influence curves and the non-parametric bootstrap requires the estimator to be asymptotically linear. The estimator must converge to a normal limit and bias must go to 0 at rate faster than $1/\sqrt{n}$.
- **Simple Substitution:** There is no theory guaranteeing that the simple substitution estimator using Super Learner is asymptotically linear (or even has a limit distribution). However, if the conditional mean outcome $\mathbb{E}_0(Y|A,W)$ was estimated with a **correctly specified** parametric regression, statistical inference for the simple substitution estimator can based on the non-parametric bootstrap or the Delta Method.
- **IPTW:** If the propensity score $\mathbb{P}_0(A=1|W)$ was estimated using Super Learner, there is no guarantee that resulting IPTW is asymptotically linear (or even has a limit distribution). However, if the propensity score $\mathbb{P}_0(A=1|W)$ was estimated with a **correctly specified** parametric regression, statistical inference for the 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 variance estimator yourself. Alternatively, for the modified Horvitz-Thompson (i.e., stablized IPTW) 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 IPTW estimator can be obtained with standard software.) If the propensity score $\mathbb{P}_0(A=1|W)$ was estimated with a correctly specified parametric model, the resulting standard error estimates will be conservative.
- **TMLE** requires estimation of both the conditional mean outcome $\mathbb{E}_0(Y|A,W)$ and the propensity score $\mathbb{P}_0(A=1|W)$. If the propensity score $\mathbb{P}_0(A=1|W)$ is consistently estimated, TMLE will be asymptotically linear with variance *conservatively* approximated by the sample variance of the estimated influence curve $\hat{IC}$ 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.

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 $\mathcal{M}^*$), all background factors are independent. The causal risk difference $\Psi^*(\mathbb{P}^*)$ is 0. The counterfactual probability of survival would be 0% higher if all participants were "good guys" than if none were.

```
> source('RLab6_datagen.R')
> #----------------------------------------
> # generateData - function to generate the data
> # input: number of draws, whether or not there is a treatment effect,
> #   whether or not to return the counterfactuals or the observed data
> # output: counterfactuals or the observed data
> #----------------------------------------
> generateData


function (n, effect = T, get.psi.star = 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)
    Y.0 <- generateY(W1 = W1, W2 = W2, W3 = W3, W4 = W4, A = 0,
        U.Y = U.Y)
    if (!effect) {
        Y.1 <- Y.0
    }
    else {
        Y.1 <- generateY(W1 = W1, W2 = W2, W3 = W3, W4 = W4,
            A = 1, U.Y = U.Y)
    }
    Y <- rep(NA, n)
    Y[A == 1] <- Y.1[A == 1]
    Y[A == 0] <- Y.0[A == 0]
    if (get.psi.star) {
        data <- data.frame(Y.1, Y.0)
    }
    else {
        data <- data.frame(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)
}
```

References

E. Polley, E. LeDell, C. Kennedy, and M. van der Laan. *SuperLearner: Super Learner Prediction*, 2018. URL http://CRAN.R-project.org/package=SuperLearner. R package version 2.0-24.

M. van der Laan and S. Rose. *Targeted Learning: Causal Inference for Observational and Experimental Data.* Springer, New York Dordrecht Heidelberg London, 2011.