R Lab 1 - Defining the Causal Parameter & Introduction to Simulations in R

Introduction to Causal Inference

Goals:
1. Review the structural causal model (SCM), counterfactuals, and causal parameters, defined with and without a working marginal structural model (MSM).
2. Given a specific data generating process, obtain the value of the target causal parameter $\Psi^F(P_{U,X})$ in closed form.
3. Introduce simulations in R. Translate a specific data generating process, which is an element of the causal model, into an R simulation.
4. Simulate the counterfactual outcome Y_a for levels of the exposure of interest, and obtain the value of target causal parameter using simulations.

Next lab:
We will obtain the value of the statistical estimand, which under the needed identifiability assumptions equals target causal parameter. We will also implement the simple substitution estimator, based on the G-computation identifiability result, and use simulations to evaluate the properties of estimators.

Reminder:
This is not an R class. However, software is an important bridge between the statistical concepts and implementation.

1 Background Story

Suppose we are interested the causal effect of butterbeer consumption on happiness among wizards at Hogwarts. Specifically, we want to know if the average happiness would be higher if all wizards consumed butterbeer or if all wizards did not. Let $W1$ be a summary measure the student's pre-exposure covariates, including age, house, gender, friendship with Dumbledore and enemy status with Snape. Let $W2$ be an additional baseline covariate, indicating whether the student had permission to travel to Hogsmeade, a location where butterbeer is sold. We consider a binary exposure A, indicating consumption of butterbeer ($A = 1$) or not ($A = 0$). Denote the outcome happiness with Y. Finally, suppose having a permission $W2$ only affects the exposure A, but has no direct effect on the happiness Y.

This study can be translated into the following, structural causal model (SCM) $\mathcal{M}^F$:

$$\begin{aligned}
\text{Endogenous Nodes: } & X = (W1, W2, A, Y) \\
\text{Exogenous Nodes: } & U = (U_{W1}, U_{W2}, U_A, U_Y) \sim P_U \\
\text{Structural Equations } F\text{: } & W1 = f_{W1}(U_{W1}) \\
& W2 = f_{W2}(W1, U_{W2}) \\
& A = f_A(W1, W2, U_A) \\
& Y = f_Y(W1, A, U_Y)
\end{aligned}$$

http://www.funcage.com/blog/10-incredibly-nerdy-dog-costumes/

1. **Draw the corresponding directed acyclic graph (DAG).**
2. **Are there any exclusion restrictions? Are there any independence assumptions?**
3. **Define the counterfactual outcomes of interest with formal notation and in words.**
4. **In the SCM framework, how are counterfactuals derived? What does the SCM tell us about all possible distributions for these counterfactuals?**

2 Target causal parameters defined without a MSM

Suppose our target causal parameter is the average treatment effect:

$$\begin{aligned}\Psi^F(P_{U,X}) &= E_{U,X}(Y_1) - E_{U,X}(Y_0) \\ &= E_{U,X}\left[f_Y(W1, 1, U_Y)\right] - E_{U,X}\left[f_Y(W1, 0, U_Y)\right]\end{aligned}$$

This is the difference in the expected counterfactual happiness if all wizards were to drink butterbeer and the expected counterfactual happiness if all wizards were not to drink butterbeer. In the second line, we have replaced the counterfactual outcome Y_a with the corresponding structural equation.

In the previous section, we specified a SCM $\mathcal{M}^F$, reflecting our limited knowledge of the data generating system. We did not place any assumptions on the joint distribution of the exogenous nodes. We made only one exclusion restriction. Finally, we did not make any assumptions about the functional form of the structural equations.

Now, we consider a particular data generating process $P_{U,X}$, one of many compatible with $\mathcal{M}^F$.

- Each of the exogenous factors U is drawn independently from the following distributions:

$$\begin{aligned}U_{W1} &\sim Uniform(min = 0, max = 1) \\ U_{W2} &\sim Bernoulli(p = 0.5) \\ U_A &\sim Normal(\mu = -3, \sigma^2 = 1^2) \\ U_Y &\sim Normal(\mu = 0, \sigma^2 = 0.3^2)\end{aligned}$$

- Let us also specify the structural equations F:

$$\begin{aligned} W1 &= f_{W1}(U_{W1}) = \mathbb{I}\big[U_{W1} < 0.35\big] \\ W2 &= f_{W2}(W1, U_{W2}) = W1 + 2^{*}U_{W2} \\ A &= f_A(W1, W2, U_A) = \mathbb{I}\big[(1 + W1 + 2^{*}W2 + U_A) > 0\big] \\ Y &= f_Y(W1, A, U_Y) = 1 + 2.5^{*}A + 3^{*}W1 - 0.25^{*}A^{*}W1 + U_Y \end{aligned}$$

  where $\mathbb{I}[\cdot]$ is the indicator function and equal to 1 if the statement in the brackets is true.
- The distribution of exogenous terms U and the set of structural equations F give us the joint distribution of (U, X), denoted $P_{U,X}$. This distribution is an element of (i.e. is one possible distribution compatible with) the SCM: $P_{U,X} \in \mathcal{M}^F$.

1. **Evaluate $\Psi^F(P_{U,X})$ for this data generating process.**
2. **Interpret $\Psi^F(P_{U,X})$.**

2.1 Translating this data generating process for $P_{U,X}$ into simulations

1. **First, set the seed to 252. Type `set.seed(252)`.** The `set.seed` function ensures the same values are generated each time each time we run all of our code. More information about this function can be accessed with `?set.seed`. Briefly, R generates the exogenous input U by calling a pseudorandom number generator to simulate a $Uniform(0, 1)$ variable and then transforming it to correspond with a draw from the specified distribution. For example, a Bernoulli random variable with probability p is generated as an indicator that this $Uniform(0, 1)$ is $< p$.
2. **Set `n=5000` as the the number of draws from $P_{U,X}$.**
3. **Simulate the background factors U.** To simulate from the uniform distribution, use `runif()` function, setting the minimum and maximum values. To simulate from a Bernoulli with probability p, use the `rbinom()` function, setting the size=1. To simulate from a normal distribution, use the `rnorm()` function, setting the mean and *standard deviation.* More information on these functions can be found by typing `?runif`, `?rnorm` or `?rbinom`.
4. **Evaluate the structural equations F to deterministically generate the endogenous nodes X.** There are several ways to code indicator functions. One way is with logical variables and the `as.numeric` function. Consider the following example.

```
> # assign x=20 and x2=30
> x <- 20
> x2 <- 30
> # create a logical variable x3 from the result of the test x> x2
> x3<- (x > x2)
> x3

[1] FALSE

> # we can convert the logical variable into binary with the as.numeric function
> # R converts TRUE to 1 and FALSE to 0
> x3<- as.numeric(x > x2)
> x3

[1] 0
```

5. **Create a data frame `X` to hold these values.** The rows are the n repetitions of the experiment and the columns are the random variables. Use the `data.frame` function. Then use the `head` and `summary` functions to get a better understanding of the data. (More information on these functions can be obtained with `?data.frame`, `?head` and `?summary`.)

2.2 Generate the counterfactual outcomes & obtain the value of the target causal parameter

1. **Intervene on the above data generating system to set butterbeer consumption $A = a$ and generate counterfactual outcomes Y_a.**
2. **Add the resulting columns to the `X` matrix using either the `data.frame` or `cbind` function.**
3. **Does the counterfactual value Y_a equal the observed Y when $A = a$?**
4. Above, we evaluated the average treatment effect in closed form. **Estimate $\Psi^F(P_{U,X})$ with the simulated counterfactuals.** Given a large sample of i.i.d. draws of (Y_0, Y_1), we can closely approximate $E_{U,X}[Y_1 - Y_0]$ with the sample average of the observed values of $Y_1 - Y_0$.

3 Target causal parameters defined with a MSM

Suppose we are now interested in summarizing how the expectation of counterfactual happiness Y_a varies as a function of cups of butterbeer consumed a. We will assume that the latest wizarding technology allows precise measurement of the amount of butterbeer consumed so that a may take on decimal values. For simplicity, ignore baseline covariates.

We could use a marginal structural model (MSM) to summarize how the expected counterfactual outcome changes as function of the exposure. Suppose, however, that we do not know the exact shape of the butterbeer-happiness (dose-response) curve. Therefore, we use the following *working* MSM to define the target parameter β as the projection of the true causal curve $E_{U,X}(Y_a)$ onto a summary model $m(a|\beta)$:

$$\beta(P_{U,X}|m) = argmin_\beta \; E_{U,X}\left[\sum_{a\in\mathcal{A}} \left(Y_a - m(a|\beta)\right)^2\right]$$
$$m(a|\beta) = \beta_0 + \beta_1 a$$

The causal parameter is then the value of the β coefficients that minimize the sum of squared residuals between the counterfactual outcomes Y_a and the predicted $m(a|\beta)$ for all possible exposure levels $a \in \mathcal{A}$. In this case, we are using a linear summary.

3.1 A specific data generating process

Consider the following data generating process for $P_{U,X}$:

$$U_A \sim Uniform(min = 0, max = 2)$$
$$U_Y \sim Normal(\mu = 0, \sigma^2 = 0.3^2)$$
$$A = f_A(U_A) = 2^* U_A$$
$$Y = f_Y(A, U_Y) = 4 + 9^* A - 2.25^* A^2 + U_Y$$

where $U_A \perp\!\!\!\perp U_Y$. Suppose we are interested in the counterfactual happiness Y_a under butterbeer consumption $a \in \mathcal{A} = \{0, 1, 2, 3, 4\}$ cups.

1. **For $n = 5$ wizards, generate the exogenous factors $U = (U_A, U_Y)$. Then set $A = 0$ to simulate the counterfactual happiness under 0 cups of butterbeer `Y.0`. Repeat for each of the 4 other doses of butterbeer to generate counterfactual outcomes `Y.1`, `Y.2`, `Y.3` and `Y.4`.**
2. **Use the `c()` function to combine (stack) all the counterfactual outcomes (`Y.0`, `Y.1`, `Y.2`, `Y.3`, `Y.4`) into a single vector `Y.a`.**

```
> Y.a<- c(Y.0, Y.1, Y.2, Y.3, Y.4)
```

3. **Use the `rep()` function to create vector `a` with the corresponding exposures.**

```
> a<- c( rep(1, 5), rep(2, 5), rep(3, 5), rep(4, 5), rep(5,5))
> a

 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5
```

4. **Create data frame $X = (a, Y_a)$, consisting of the set treatment levels and corresponding counterfactual outcomes.**

5. **Plot the counterfactual outcomes Y_a as a function of a.** More information can be accessed with `?plot` and more plotting options with `?par`.

```
> # Hint: this code would plot x2 as a function of x2 and add a title
> plot(x, x2, main='Plot of x2 by x')
```

6. **Based on your knowledge of the data generating system as well as the graph, do you think a working MSM with a linear form is a good summary?**

3.2 Obtain the value of the target causal parameter

1. **We have the counterfactuals Y_a for $a \in \mathcal{A} = \{0, 1, 2, 3, 4\}$ and have defined the target parameter using the least squares projection (i.e. with the L2 loss function). Use `glm()` function to fit the coefficients of the $m(a|\beta)$. Interpret the results.**

2. **Add the projected causal curve to the graph. See `abline()`.**

3. ***Bonus:* Add the true causal curve to the graph. Use the `seq()` function to generate lots of doses between $A = 0$ and $A = 4$. Then evaluate the mean counterfactual $E_{U,X}(Y_a) = 4 + 9a - 2.25a^2$ at these points. Finally, use the `lines()` function to add the line.**

Appendix - More about this document

This document is written in Sweave, a programming language in `R` for reproducible research and to embed `R` code in a LaTex document. (See www.statistik.lmu.de/~leisch/Sweave for more information.) The following is a code "chunk" to assign the value 3 to random variable x.

```
> # comments in R are denoted by the # symbol
> x<- 3
> x

[1] 3
```

Code that is longer than one line will have `+` signs at line breaks. For example, consider the following code for the *logit* and *expit* functions.

```
> # logit function: given input x, returns log(x/(1-x))
> logit<- function(x){
+   log(x/(1-x))
+ }
```

```
> # expit function: given input x, returns 1/(1+e^{-x})
> expit<- function(x){
+   1/(1+ exp(-x))
+ }
> # let's try it out for a probability x=0.25
> x<- 0.25
> x2<- logit(x)
> x2


[1] -1.098612


> expit(x2)


[1] 0.25
```

Help for functions can be accessed with ? and then the function name. For example, the `qlogis` function is equivalent to the `logit` function specified above. Likewise the `plogis` function is equivalent to the `expit` function.

```
> # get information about the qlogis() function
> ?qlogis


> qlogis(x)== logit(x)


[1] TRUE


> plogis(x)== expit(x)


[1] TRUE
```