Modeling data without any underlying causal theory can sometimes lead you down the wrong path, particularly if you are interested in understanding the *way* things work rather than making *predictions.* A while back, I described what can go wrong when you control for a mediator when you are interested in an exposure and an outcome. Here, I describe the potential biases that are introduced when you inadvertently control for a variable that turns out to be a ***collider***.

Be Careful not to control for post -exposure covariate

A researcher was presenting an analysis of the impact various types of childhood trauma might have on subsequent substance abuse in adulthood. Obviously, a very interesting and challenging research question. The statistical model included adjustments for several factors that are plausible confounders of the relationship between trauma and substance use, such as childhood poverty. However, the model also include a measurement for poverty in adulthood - believing it was somehow confounding the relationship of trauma and substance use. A confounder is a common cause of an exposure/treatment and an outcome; it is hard to conceive of adult poverty as a cause of childhood events, even though it might be related to adult substance use (or maybe not). At best, controlling for adult poverty has no impact on the conclusions of the research; less good, though, is the possibility that it will lead to the conclusion that the effect of trauma is less than it actually is.

Using a highly contrived simulation of data and the abstract concept of *potential outcomes*, I am hoping to illuminate some of the issues raised by this type of analysis.

**Potential outcomes and causal effects**

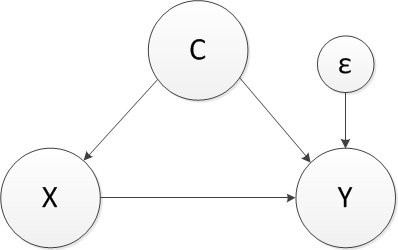
The field of causal inference is a rich one, and I won’t even scratch the surface here. My goal is to present the concepts of potential outcomes so that we can articulate at least one clear way to think about what a causal effect can be defined. Under this framework, we generate data where we can find out the “true” measure of causal effect. And then we can use simple regression models to see how well (or not) they recapture these “known” causal effects.

If an individual i*i* experiences a traumatic effect as a child, we say that the exposure X\_i = 1*Xi*​=1. Otherwise X\_i = 0*Xi*​=0, there was no traumatic event. (I am going to assume binary exposures just to keep things simple - exposed vs. not exposed.) In the potential outcomes world we say that every individual has possible outcomes Y\_{1i}*Y*1*i*​ (the outcome we would observe *if* the individual had experienced trauma) and Y\_{0i}*Y*0*i*​ (the outcome we would observe *if* the individual had not. Quite simply, we define the causal effect of X*X* on Y*Y* as the difference in potential outcomes, CE\_i = Y\_{1i} - Y\_{0i}*CEi*​=*Y*1*i*​−*Y*0*i*​. If Y\_{1i} = Y\_{0i}*Y*1*i*​=*Y*0*i*​ (i.e. the potential outcomes are the same), we would say that X*X* does not cause Y*Y*, at least for individual i*i*.

In the real world, we only observe one potential outcome - the one associated with the actual exposure. The field of causal inference has lots to say about the assumptions and conditions that are required for us to use observed data to estimate average causal effects; many would say that unless we use a randomized controlled study, those assumptions will never be reasonable. But in the world of simulation, we can generate potential outcomes and observed outcomes, so we know the causal effect both at the individual level and the average population level. And we can see how well our models do.

**Simple confounding**

Here’s a relatively straightforward example. Let’s say we are interested in understanding if some measure X*X* causes an outcome Y*Y*, where there is a common cause C*C* (the diagram is called a DAG - a directed acyclic graph - and is useful for many things, including laying out data generating process):



**library**(broom)

**library**(data.table)

**library**(simstudy)

def <- defData(varname = "C", formula = 0.4, dist = "binary")

def <- defData(def, "X", formula = "0.3 + 0.4 \* C", dist = "binary")

def <- defData(def, "e", formula = 0, variance = 2, dist = "normal")

def <- defData(def, "Y0", formula = "2 \* C + e", dist="nonrandom")

def <- defData(def, "Y1", formula = "0.5 + 2 \* C + e", dist="nonrandom")

def <- defData(def, "Y\_obs", formula = "Y0 + (Y1 - Y0) \* X", dist = "nonrandom")

def

## varname formula variance dist link

## 1: C 0.4 0 binary identity

## 2: X 0.3 + 0.4 \* C 0 binary identity

## 3: e 0 2 normal identity

## 4: Y0 2 \* C + e 0 nonrandom identity

## 5: Y1 0.5 + 2 \* C + e 0 nonrandom identity

## 6: Y\_obs Y0 + (Y1 - Y0) \* X 0 nonrandom identity

In this example, X*X* does have an effect on Y*Y*, but so does C*C*. If we ignore C*C* in assessing the size of the effect of X*X* on Y*Y*, we will overestimate that effect, which is 0.5. We can generate data and see that this is the case:

set.seed(5)

dt <- genData(1000, def)

We see that the true causal effect is easily recovered if we have access to the potential outcomes Y\_1*Y*1​ and Y\_0*Y*0​, but of course we don’t:

dt[, mean(Y1 - Y0)] # True causal effect

## [1] 0.5

If we compare the average *observed* outcomes for each exposure group ignoring the confounder, we overestimate the effect of the exposure:

dt[X == 1, mean(Y\_obs)] - dt[X == 0, mean(Y\_obs)]

## [1] 1.285009

We can estimate the same effect using simple linear regression:

lm1 <- lm(Y\_obs ~ X, data = dt)

tidy(lm1)

## # A tibble: 2 x 5

## term estimate std.error statistic p.value

## <chr> <dbl> <dbl> <dbl> <dbl>

## 1 (Intercept) 0.552 0.0733 7.53 1.14e-13

## 2 X 1.29 0.107 12.0 2.92e-31

And finally, if we adjust for the confounder C*C*, we recover the true causal effect of X*X* on Y*Y*, or at least get very close to it:

lm2 <- lm(Y\_obs ~ X + C, data = dt)

tidy(lm2)

## # A tibble: 3 x 5

## term estimate std.error statistic p.value

## <chr> <dbl> <dbl> <dbl> <dbl>

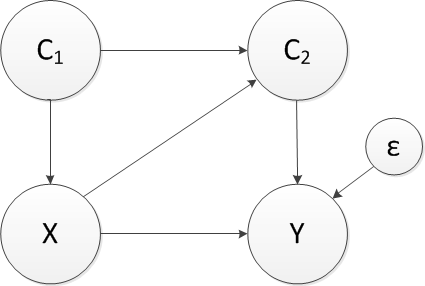
## 1 (Intercept) 0.0849 0.0650 1.31 1.92e- 1

## 2 X 0.489 0.0968 5.06 5.08e- 7

## 3 C 2.06 0.0983 20.9 5.77e-81

**Adjusting for a post-exposure covariate**

Now, we are ready to see what happens in a slightly more complicated setting that is defined by this DAG:



In this example C*C* is measured in two time periods, and exposure in period 1 relates to exposure in period 2. (For example, if a child is poor, he is more likely to be poor as an adult.) We are primarily interested in whether or not X*X* (trauma) causes Y*Y* (substance use). The difficulty is that X*X* and C\_2*C*2​ are related, as are C\_2*C*2​ and Y*Y*.

I suggest that in order to fully understand the effect of X*X* on Y*Y*, we cannot control for C\_2*C*2​, as tempting as it might be. The intuition is that part of the effect of X*X* on Y*Y* is due to the fact that X*X* has an effect on C\_2*C*2​, at least for some individuals. *If we control for C\_2C2​, we are actually removing a key component of the causal mechanism.* Below in is the data generating process - a few things to note: (1) C\_2*C*2​ has potential outcomes based on the exposure X*X*. (2) We have restricted the potential outcome C\_{21}*C*21​ to be set to 1 if C\_{20}*C*20​ is 1. For example, if someone would have been poor in adulthood *without* exposure to trauma, we assume that they also would have been poor in adulthood had they been exposed to trauma. (3) The potential outcome for Y*Y* is dependent on the relevant potential outcome for C\_2*C*2​. That is Y\_0*Y*0​ depends on C\_{20}*C*20​ and Y\_1*Y*1​ depends on C\_{21}*C*21​.

def2 <- defData(varname = "C1", formula = .25, dist = "binary")

def2 <- defData(def2, "X", formula = "-2 + 0.8 \* C1", dist = "binary", link = "logit")

def2 <- defData(def2, "C2.0", formula = "-2.0 + 1 \* C1", dist = "binary", link = "logit")

def2 <- defData(def2, "C2.1x", formula = "-1.5 + 1 \* C1", dist = "binary", link = "logit")

def2 <- defData(def2, "C2.1", formula = "pmax(C2.0, C2.1x)", dist = "nonrandom")

def2 <- defData(def2, "e", formula = 0, variance = 4, dist = "normal")

def2 <- defData(def2, "Y0", formula = "-3 + 5\*C2.0 + e", dist = "nonrandom")

def2 <- defData(def2, "Y1", formula = "0 + 5\*C2.1 + e", dist = "nonrandom")

def2 <- defData(def2, "C2\_obs", formula = "C2.0 + (C2.1 - C2.0) \* X", dist = "nonrandom")

def2 <- defData(def2, "Y\_obs", formula = "Y0 + (Y1 - Y0) \* X", dist = "nonrandom")

set.seed(25)

dt <- genData(5000, def2)

Here is the true average causal effect, based on information we will never know:

dt[, mean(Y1 - Y0)]

## [1] 3.903

When we control for C\_2*C*2​, we are essentially estimating the effect of X*X* at each level C\_2*C*2​ (and C\_1*C*1​, since we are controlling for that as well), and then averaging across the sub-samples to arrive at an estimate for the entire sample. We can see that, based on the specification of the potential outcomes in the data generation process, the effect at each level of C\_2*C*2​ will be centered around 3.0, which is different from the true causal effect of 3.9. The discrepancy is due to the fact each approach is effectively collecting different sub-samples (one defines groups based on set levels of X*X* and C\_2*C*2​, and the other defines groups based on set levels of X*X* alone) and estimating average effects based on weights determined by the sizes of those two sets of sub-samples.

Here is the inappropriate model that adjusts for C\_2*C*2​:

lm2a <- lm( Y\_obs ~ C1 + C2\_obs + X , data = dt)

tidy(lm2a)

## # A tibble: 4 x 5

## term estimate std.error statistic p.value

## <chr> <dbl> <dbl> <dbl> <dbl>

## 1 (Intercept) -3.01 0.0348 -86.6 0.

## 2 C1 -0.0208 0.0677 -0.307 7.59e- 1

## 3 C2\_obs 4.93 0.0763 64.6 0.

## 4 X 3.05 0.0811 37.5 6.68e-272

The estimate for the coefficient of X*X* is 3.0, just as anticipated. Here now is the correct model, and you will see that we recover the true causal effect in the coefficient estimate of X*X* (or at least, we get much, much closer):

lm2b <- lm( Y\_obs ~ C1 + X , data = dt)

tidy(lm2b)

## # A tibble: 3 x 5

## term estimate std.error statistic p.value

## <chr> <dbl> <dbl> <dbl> <dbl>

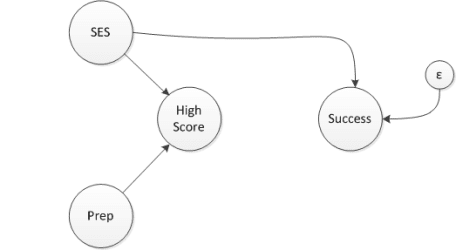
## 1 (Intercept) -2.49 0.0459 -54.3 0.

## 2 C1 0.967 0.0893 10.8 5.32e- 27

## 3 X 3.94 0.108 36.3 7.87e-257

Of course, in the real world, we don’t know the underlying data generating process or the true DAG. And what I have described here is a gross oversimplification of the underlying relationships, and have indeed left out many other factors that likely affect the relationship between childhood trauma and adult substance use. Other measures, such as parental substance use, may be related to both childhood trauma and adult substance use, and may affect poverty in the two time periods in different, complicated ways.

A collider, like a mediator, is a post-exposure/post-intervention outcome. Unlike a mediator, a collider is not necessarily causally related to the outcome of interest. (This is not to say that it cannot be, which is why this concept came up in a talk I gave about marginal structural models, described [here](https://www.rdatagen.net/post/potential-outcomes-confounding/), [here](https://www.rdatagen.net/post/inverse-probability-weighting-when-the-outcome-is-binary/), and [here](https://www.rdatagen.net/post/when-a-covariate-is-a-confounder-and-a-mediator/).) The key distinction of a collider is that it is an outcome that has two causes. In a directed acyclic graph (or [DAG](http://www.dagitty.net/learn/index.html)), a collider is a variable with two arrows pointing towards it. This is easier to see visually:



Code Chunks - Visualizing how confounding biases estimates of population-wide (or marginal) average causal effects

Our goal is to compare the distribution of outcomes for the control group with the exposed group. We often simplify this comparison by looking at the means of each distribution. The average causal effect (across all individuals) can be written as E(Y\_1 - Y\_0)*E*(*Y*1​−*Y*0​), where E()*E*() is the expectation or average. In reality, we cannot directly measure this since only one potential outcome is observed for each individual.

Using the following logic, we might be able to convince ourselves that we can use observed measurements to estimate unobservable average causal effects. First, we can say E(Y\_1 - Y\_0) = E(Y\_1) - E(Y\_0)*E*(*Y*1​−*Y*0​)=*E*(*Y*1​)−*E*(*Y*0​), because expectation is linear. Next, it seems fairly reasonable to say that E(Y\_1 | A = 1) = E(Y | A = 1)*E*(*Y*1​∣*A*=1)=*E*(*Y*∣*A*=1), where A=1*A*=1 for exposure, A=0*A*=0 for control. In words, this states that the average **potential outcome of exposure** for the **exposed group** is the same as what we actually **observe** for the **exposed group** (this is the consistency assumption in causal inference theory). Along the same lines, E(Y\_0 | A = 0) = E(Y | A = 0)*E*(*Y*0​∣*A*=0)=*E*(*Y*∣*A*=0). Finally, if we can say that E(Y\_1) = E(Y\_1 | A = 1)*E*(*Y*1​)=*E*(*Y*1​∣*A*=1) - the potential outcome of exposure for **everyone** is equal to the potential outcome of exposure for those **exposed** - then we can say that E(Y\_1) = E(Y | A = 1)*E*(*Y*1​)=*E*(*Y*∣*A*=1) (the potential outcome with exposure for **everyone** is the same as the observed outcome for **the exposed**. Similarly, we can make the same argument to conclude that E(Y\_0) = E(Y | A = 0)*E*(*Y*0​)=*E*(*Y*∣*A*=0). At the end of this train of logic, we conclude that we can estimate E(Y\_1 - Y\_0)*E*(*Y*1​−*Y*0​) using observed data only: E(Y | A = 1) - E(Y | A = 0)*E*(*Y*∣*A*=1)−*E*(*Y*∣*A*=0).

This nice logic fails if E(Y\_1) \ne E(Y | A = 1)*E*(*Y*1​)̸​=*E*(*Y*∣*A*=1) and/or E(Y\_0) \ne E(Y | A = 0)*E*(*Y*0​)̸​=*E*(*Y*∣*A*=0). That is, this nice logic fails when there is confounding.

This is all a very long-winded way of saying that confounding arises when the distributions of potential outcomes **for the population** are different from those distributions for **the subgroups** we are using for analysis. For example, if the potential outcome under exposure for the population as a whole (Y\_1*Y*1​) differs from the observed outcome for the subgroup that was exposed (Y|A=1*Y*∣*A*=1), or the potential outcome without exposure for the entire population (Y\_0*Y*0​) differs from the observed outcome for the subgroup that was not exposed (Y|A=0*Y*∣*A*=0), any estimates of population level causal effects using observed data will be biased.

However, if we can find a factor L*L* (or factors) where

\begin{aligned} P(Y\_1 | L=l) &amp;= P(Y | A = 1 \text{ and } L=l) \\ P(Y\_0 | L=l) &amp;= P(Y | A = 0 \text{ and } L=l) \end{aligned}*P*(*Y*1​∣*L*=*l*)*P*(*Y*0​∣*L*=*l*)​=*P*(*Y*∣*A*=1 and *L*=*l*)=*P*(*Y*∣*A*=0 and *L*=*l*)​both hold for all levels or values of L*L*, we can remove confounding (and get unbiased estimates of the causal effect) by “controlling” for L*L*. In some cases, the causal effect we measure will be conditional on L*L*, sometimes it will be a population-wide average (or marginal) causal effect, and sometimes it will be both.

### What confounding looks like …

The easiest way to illustrate the population/subgroup contrast is to generate data from a process that includes confounding. In this first example, the outcome is continuous, and is a function of both the exposure (A*A*) and a covariate (L*L*). For each individual, we can generate both potential outcomes Y\_0*Y*0​ and Y\_1*Y*1​. (Note that both potential outcomes share the same individual level noise term e*e* - this is not a necessary assumption.) This way, we can “know” the true population, or marginal causal effect of exposure. The observed outcome Y*Y* is determined by the exposure status. For the purposes of plotting a smooth density curve, we generate a very large sample - 2 million.

**library**(simstudy)

defC <- defData(varname = "e", formula = 0, variance = 2,

dist = "normal")

defC <- defData(defC, varname = "L", formula = 0.4,

dist = "binary")

defC <- defData(defC, varname = "Y0", formula = "1 + 4\*L + e",

dist = "nonrandom")

defC <- defData(defC, varname = "Y1", formula = "5 + 4\*L + e",

dist = "nonrandom")

defC <- defData(defC, varname = "A", formula = "0.3 + 0.3 \* L",

dist = "binary")

defC <- defData(defC, varname = "Y", formula = "1 + 4\*A + 4\*L + e",

dist = "nonrandom")

set.seed(2017)

dtC <- genData(n = 2000000, defC)

dtC[1:5]

## id e L Y0 Y1 A Y

## 1: 1 2.02826718 1 7.0282672 11.028267 1 11.0282672

## 2: 2 -0.10930734 0 0.8906927 4.890693 0 0.8906927

## 3: 3 1.04529790 0 2.0452979 6.045298 0 2.0452979

## 4: 4 -2.48704266 1 2.5129573 6.512957 1 6.5129573

## 5: 5 -0.09874778 0 0.9012522 4.901252 0 0.9012522

Feel free to skip over this code - I am just including in case anyone finds it useful to see how I generated the following series of annotated density curves:

**library**(ggplot2)

getDensity <- **function**(vector, weights = NULL) {

**if** (!is.vector(vector)) **stop**("Not a vector!")

**if** (is.null(weights)) {

avg <- mean(vector)

} **else** {

avg <- weighted.mean(vector, weights)

}

close <- min(which(avg < density(vector)$x))

x <- density(vector)$x[close]

**if** (is.null(weights)) {

y = density(vector)$y[close]

} **else** {

y = density(vector, weights = weights)$y[close]

}

**return**(data.table(x = x, y = y))

}

plotDens <- **function**(dtx, var, xPrefix, title, textL = NULL, weighted = FALSE) {

dt <- copy(dtx)

**if** (weighted) {

dt[, nIPW := IPW/sum(IPW)]

dMarginal <- getDensity(dt[, get(var)], weights = dt$nIPW)

} **else** {

dMarginal <- getDensity(dt[, get(var)])

}

d0 <- getDensity(dt[L==0, get(var)])

d1 <- getDensity(dt[L==1, get(var)])

dline <- rbind(d0, dMarginal, d1)

brk <- round(dline$x, 1)

p <- ggplot(aes(x=get(var)), data=dt) +

geom\_density(data=dt[L==0], fill = "#ce682f", alpha = .4) +

geom\_density(data=dt[L==1], fill = "#96ce2f", alpha = .4)

**if** (weighted) {

p <- p + geom\_density(aes(weight = nIPW),

fill = "#2f46ce", alpha = .8)

} **else** p <- p + geom\_density(fill = "#2f46ce", alpha = .8)

p <- p + geom\_segment(data = dline, aes(x = x, xend = x,

y = 0, yend = y),

size = .7, color = "white", lty=3) +

annotate(geom="text", x = 12.5, y = .24,

label = title, size = 5, fontface = 2) +

scale\_x\_continuous(limits = c(-2, 15),

breaks = brk,

name = paste(xPrefix, var)) +

theme(panel.grid = element\_blank(),

axis.text.x = element\_text(size = 12),

axis.title.x = element\_text(size = 13)

)

**if** (!is.null(textL)) {

p <- p +

annotate(geom = "text", x = textL[1], y = textL[2],

label = "L=0", size = 4, fontface = 2) +

annotate(geom = "text", x = textL[3], y = textL[4],

label="L=1", size = 4, fontface = 2) +

annotate(geom = "text", x = textL[5], y = textL[6],

label="Population distribution", size = 4, fontface = 2)

}

**return**(p)

}

**library**(gridExtra)

grid.arrange(plotDens(dtC, "Y0", "Potential outcome", "Full\npopulation",

c(1, .24, 5, .22, 2.6, .06)),

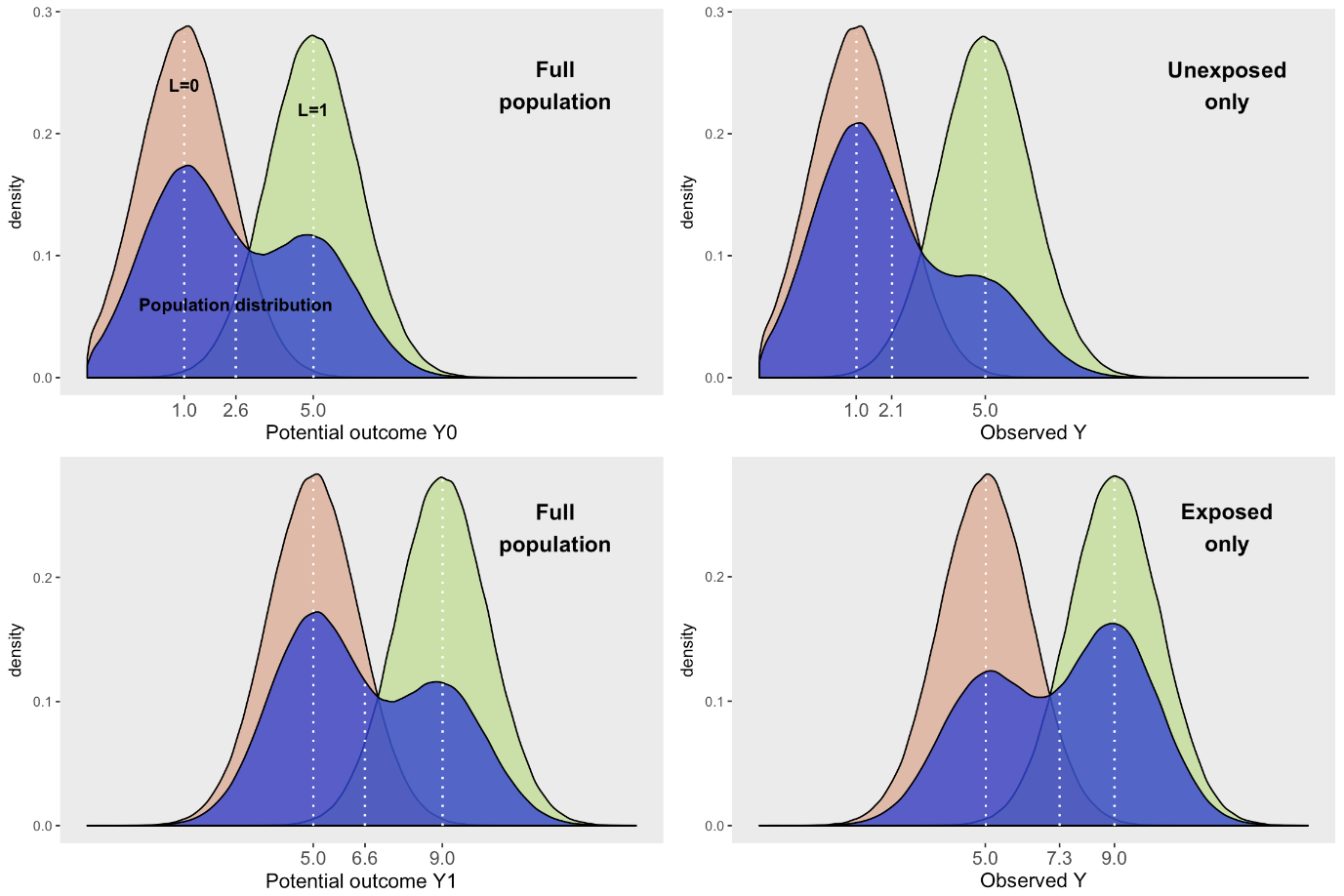
plotDens(dtC[A==0], "Y", "Observed", "Unexposed\nonly"),

plotDens(dtC, "Y1", "Potential outcome", "Full\npopulation"),

plotDens(dtC[A==1], "Y", "Observed", "Exposed\nonly"),

nrow = 2

)



Looking at the various plots, we can see a few interesting things. The density curves on the left represent the entire population. The conditional distributions of the potential outcomes at the population level are all normally distributed, with means that depend on the exposure and covariate L*L*. We can also see that the population-wide distribution of Y\_0*Y*0​ and Y\_1*Y*1​ (in blue) are non-symmetrically shaped, because they are a mixture of the conditional normal distributions, weighted by the proportion of each level of L*L*. Since the proportions for the top and bottom plots are in fact the population proportion, the population-level density curves for Y\_0*Y*0​ and Y\_1*Y*1​ are similarly shaped, with less mass on the higher end, because individuals are less likely to have an L*L* value of 1:

dtC[, .(propLis1 = mean(L))]

## propLis1

## 1: 0.399822

The shape of the marginal distribution of Y\_1*Y*1​ is identical to Y\_0*Y*0​ (in this case, because that is the way I generated the data), but shifted to the right by an amount equal to the causal effect. The conditional effect sizes are 4, as is the population or marginal effect size.

The subgroup plots on the right are a different story. In this case, the distributions of L*L* vary across the exposed and unexposed groups:

dtC[, .(propLis1 = mean(L)), keyby = A]

## A propLis1

## 1: 0 0.2757937

## 2: 1 0.5711685

So, even though the distributions of (observed) Y*Y* conditional on L*L* are identical to their potential outcome counterparts in the whole population - for example, P(Y | A=0 \text{ and } L = 1) = P(Y\_0 | L = 1)*P*(*Y*∣*A*=0 and *L*=1)=*P*(*Y*0​∣*L*=1) - the marginal distributions of Y*Y* are quite different for the exposed and unexposed. For example, P(Y | A = 0) \ne P(Y\_0)*P*(*Y*∣*A*=0)̸​=*P*(*Y*0​). This is directly due to the fact that the mixing weights (the proportions of L*L*) are different for each of the groups. In the unexposed group, about 28% have L=1*L*=1, but for the exposed group, about 57% do. Using the subgroup data only, the conditional effect sizes are still 4 (comparing mean outcomes Y*Y* within each level of L*L*). However the difference in means between the marginal distributions of each subgroup is about 5.2 (calculated by 7.3 - 2.1). This is confounding.

### No confounding

Just so we can see that when the covariate L*L* has nothing to do with the probability of exposure, the marginal distributions of the subgroups do in fact look like their population-level potential outcome marginal distributions:

defC <- updateDef(defC, "A", newformula = 0.5) # change data generation

dtC <- genData(n = 2000000, defC)

dtC[, .(propLis1 = mean(L)), keyby = A] # subgroup proportions

## A propLis1

## 1: 0 0.4006499

## 2: 1 0.3987437

dtC[, .(propLis1 = mean(L))] # population/marginal props

## propLis1

## 1: 0.3996975

grid.arrange(plotDens(dtC, "Y0", "Potential outcome", "Population",

c(1, .24, 5, .22, 2.6, .06)),

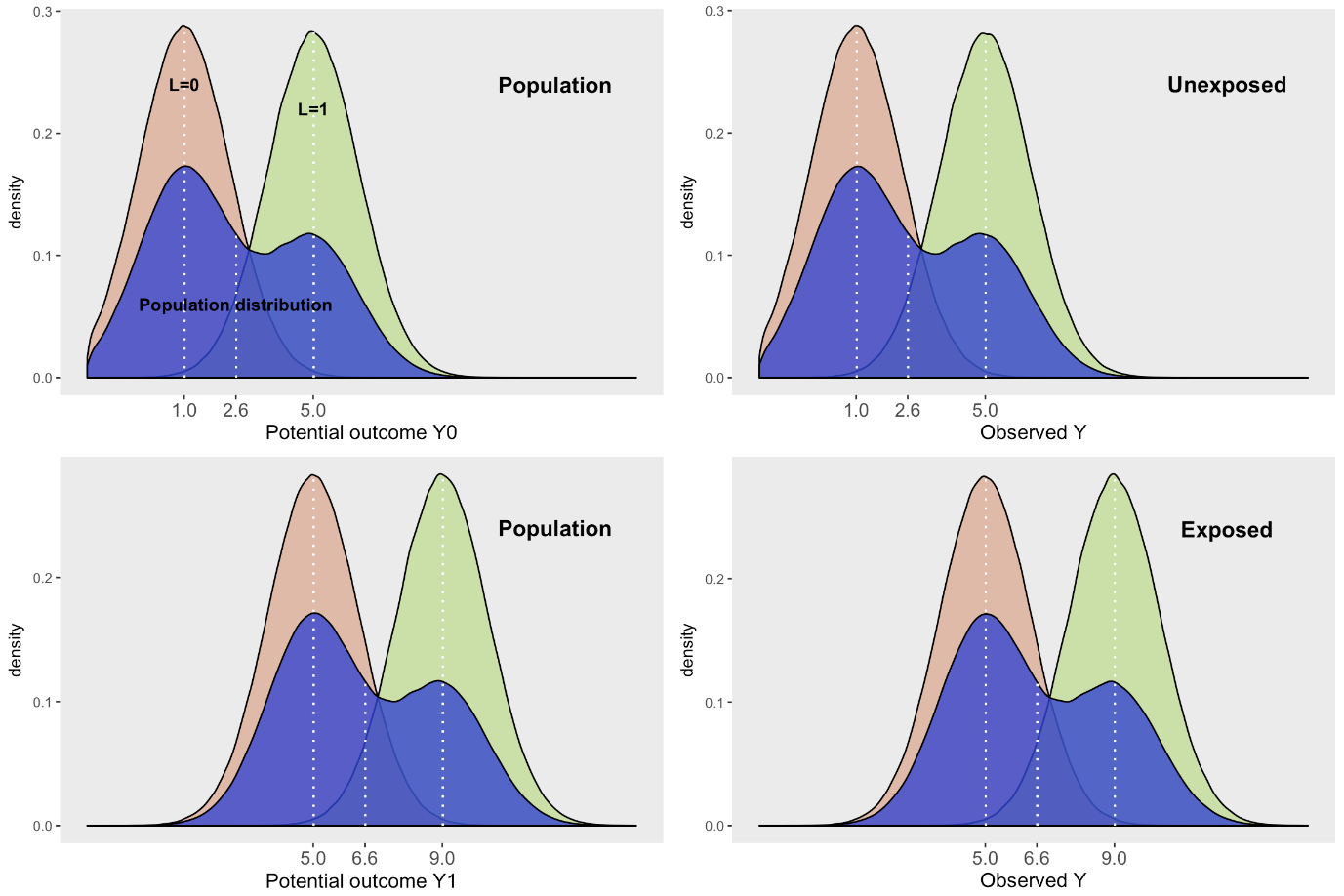
plotDens(dtC[A==0], "Y", "Observed", "Unexposed"),

plotDens(dtC, "Y1", "Potential outcome", "Population"),

plotDens(dtC[A==1], "Y", "Observed", "Exposed"),

nrow = 2

)



### Estimation of causal effects (now with confounding)

Generating a smaller data set, we estimate the causal effects using simple calculations and linear regression:

**library**(broom)

# change back to confounding

defC <- updateDef(defC, "A", newformula = ".3 + .3 \* L")

dtC <- genData(2500, defC)

The true average (marginal) causal effect from the average difference in potential outcomes for the entire population:

dtC[, mean(Y1 - Y0)]

## [1] 4

And the true average causal effects conditional on the covariate L*L*:

dtC[, mean(Y1 - Y0), keyby = L]

## L V1

## 1: 0 4

## 2: 1 4

If we try to estimate the marginal causal effect by using a regression model that does not include L*L*, we run into problems. The estimate of 5.2 we see below is the same biased estimate we saw in the plot above. This model is reporting the differences of the means (across both levels of L*L*) for the two subgroups, which we know (because we saw) are not the same as the potential outcome distributions in the population due to different proportions of L*L* in each subgroup:

tidy(lm(Y ~ A, data = dtC))

## term estimate std.error statistic p.value

## 1 (Intercept) 2.027132 0.06012997 33.71251 1.116211e-205

## 2 A 5.241004 0.09386145 55.83766 0.000000e+00

If we estimate a model that conditions on L*L*, the estimates are on target because in the context of normal linear regression without interaction terms, conditional effects are the same as marginal effects (when confounding has been removed, or think of the comparisons being made within the orange groups and green groups in the fist set of plots above, not within the purple groups):

tidy(lm(Y ~ A + L , data = dtC))

## term estimate std.error statistic p.value

## 1 (Intercept) 0.9178849 0.03936553 23.31697 5.809202e-109

## 2 A 4.0968358 0.05835709 70.20288 0.000000e+00

## 3 L 3.9589109 0.05862583 67.52844 0.000000e+00

### Inverse probability weighting (IPW)

What follows briefly here is just a sneak preview of IPW (without any real explanation), which is one way to recover the marginal mean using observed data with confounding. For now, I am ignoring the question of why you might be interested in knowing the marginal effect when the conditional effect estimate provides the same information. Suffice it to say that the conditional effect is not always the same as the marginal effect (think of data generating processes that include interactions or non-linear relationships), and sometimes the marginal effect estimate may the best that we can do, or at least that we can do easily.

If we weight each individual observation by the inverse probability of exposure, we can remove confounding and estimate the marginal effect of exposure on the outcome. Here is a quick simulation example.

After generating the dataset (the same large one we started out with so you can compare) we estimate the probability of exposure P(A=1 | L)*P*(*A*=1∣*L*), assuming that we know the correct exposure model. This is definitely a questionable assumption, but in this case, we actually do. Once the model has been fit, we assign the predicted probability to each individual based on her value of L*L*.

set.seed(2017)

dtC <- genData(2000000, defC)

exposureModel <- glm(A ~ L, data = dtC, family = "binomial")

tidy(exposureModel)

## term estimate std.error statistic p.value

## 1 (Intercept) -0.847190 0.001991708 -425.3584 0

## 2 L 1.252043 0.003029343 413.3053 0

dtC[, pA := predict(exposureModel, type = "response")]

The IPW is not based exactly on P(A=1 | L)*P*(*A*=1∣*L*) (which is commonly used in propensity score analysis), but rather, the probability of the actual exposure at each level of L*L*: P(A=a | L)*P*(*A*=*a*∣*L*), where a\in(0,1)*a*∈(0,1):

# Define two new columns

defC2 <- defDataAdd(varname = "pA\_actual",

formula = "A \* pA + (1-A) \* (1-pA)",

dist = "nonrandom")

defC2 <- defDataAdd(defC2, varname = "IPW",

formula = "1/pA\_actual",

dist = "nonrandom")

# Add weights

dtC <- addColumns(defC2, dtC)

round(dtC[1:5], 2)

## id e L Y0 Y1 A Y pA pA\_actual IPW

## 1: 1 2.03 1 7.03 11.03 1 11.03 0.6 0.6 1.67

## 2: 2 -0.11 0 0.89 4.89 0 0.89 0.3 0.7 1.43

## 3: 3 1.05 0 2.05 6.05 0 2.05 0.3 0.7 1.43

## 4: 4 -2.49 1 2.51 6.51 1 6.51 0.6 0.6 1.67

## 5: 5 -0.10 0 0.90 4.90 0 0.90 0.3 0.7 1.43

To estimate the marginal effect on the log-odds scale, we use function lm again, but with weights specified by IPW. The true value of the marginal effect of exposure (based on the population-wide potential outcomes) was 4.0. I know I am repeating myself here, but first I am providing the biased estimate that we get when we ignore covariate L*L* to convince you that the relationship between exposure and outcome is indeed confounded:

tidy(lm(Y ~ A , data = dtC))

## term estimate std.error statistic p.value

## 1 (Intercept) 2.101021 0.002176711 965.2275 0

## 2 A 5.184133 0.003359132 1543.2956 0

And now, with the simple addition of the weights but still not including L*L* in the model, our weighted estimate of the marginal effect is spot on (but with such a large sample size, this is not so surprising):

tidy(lm(Y ~ A , data = dtC, weights = IPW))

## term estimate std.error statistic p.value

## 1 (Intercept) 2.596769 0.002416072 1074.789 0

## 2 A 4.003122 0.003416842 1171.585 0

And finally, here is a plot of the IPW-adjusted density. You might think I am just showing you the plots for the unconfounded data again, but you can see from the code (and I haven’t hidden anything) that I am still using the data set with confounding. In particular, you can see that I am calling the routine plotDens with weights.

grid.arrange(plotDens(dtC, "Y0", "Potential outcome", "Population",

c(1, .24, 5, .22, 2.6, .06)),

plotDens(dtC[A==0], "Y", "Observed", "Unexposed",

weighted = TRUE),

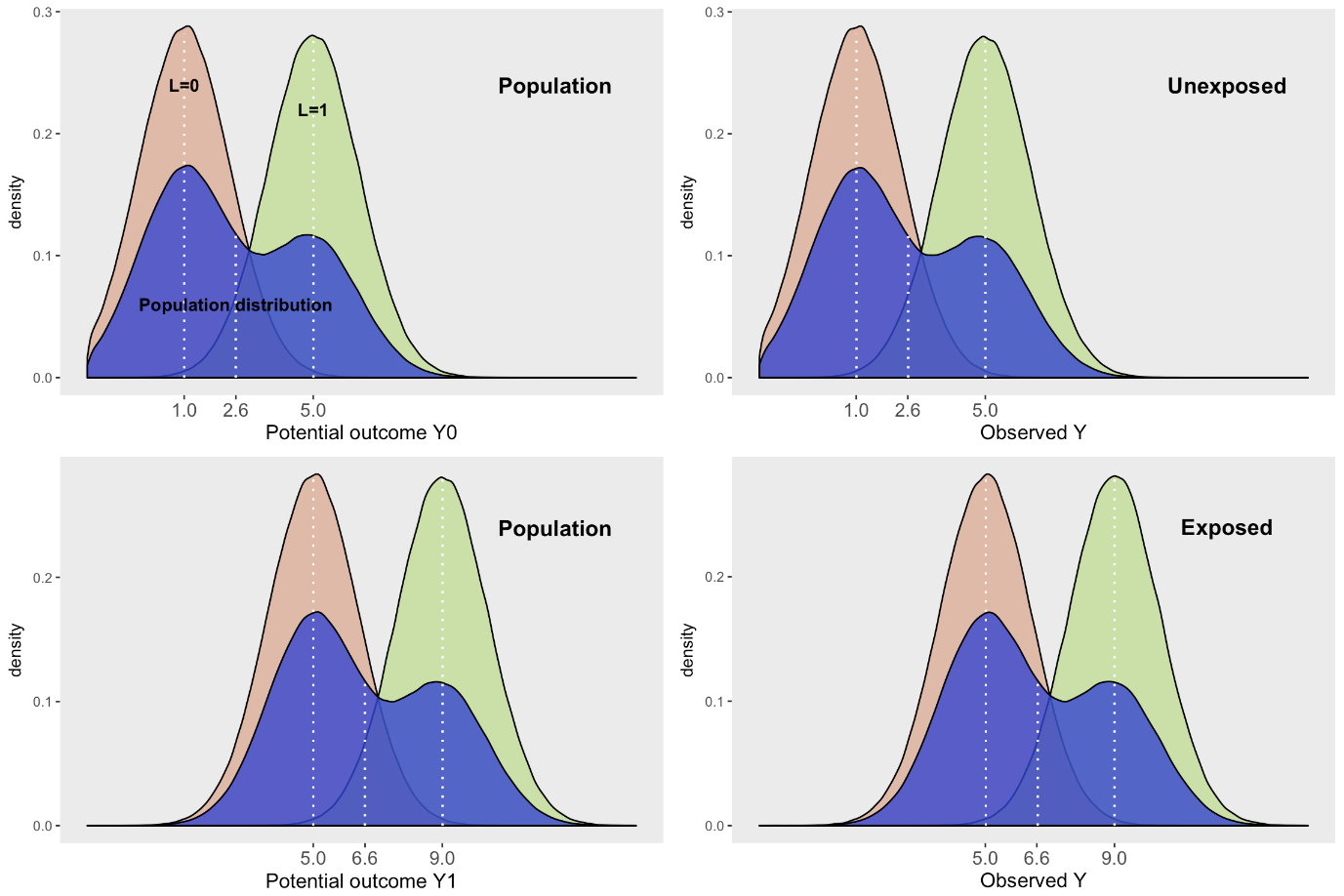
plotDens(dtC, "Y1", "Potential outcome", "Population"),

plotDens(dtC[A==1], "Y", "Observed", "Exposed",

weighted = TRUE),

nrow = 2

)



As I mentioned, I hope to write more on IPW, and marginal structural models, which make good use of this methodology to estimate effects that can be challenging to get a handle on.

Code Chunks - When you use inverse probability weighting for estimation, what are the weights actually doing?

### A simulation

Because binary outcomes are less amenable to visual illustration, I am going to stick with model estimation to see how this plays out:

**library**(simstudy)

# define the data

defB <- defData(varname = "L", formula =0.27,

dist = "binary")

defB <- defData(defB, varname = "Y0", formula = "-2.5 + 1.75\*L",

dist = "binary", link = "logit")

defB <- defData(defB, varname = "Y1", formula = "-1.5 + 1.75\*L",

dist = "binary", link = "logit")

defB <- defData(defB, varname = "A", formula = "0.315 + 0.352 \* L",

dist = "binary")

defB <- defData(defB, varname = "Y", formula = "Y0 + A \* (Y1 - Y0)",

dist = "nonrandom")

# generate the data

set.seed(2002)

dtB <- genData(200000, defB)

dtB[1:6]

## id L Y0 Y1 A Y

## 1: 1 0 0 0 0 0

## 2: 2 0 0 0 0 0

## 3: 3 1 0 1 1 1

## 4: 4 0 1 1 1 1

## 5: 5 1 0 0 1 0

## 6: 6 1 0 0 0 0

We can look directly at the potential outcomes to see the true causal effect, measured as a log odds ratio (LOR):

odds <- **function** (p) {

**return**((p/(1 - p)))

}

# log odds ratio for entire sample (marginal LOR)

dtB[, log( odds( mean(Y1) ) / odds( mean(Y0) ) )]

## [1] 0.8651611

In the linear regression context, the conditional effect measured using observed data from the exposed and unexposed subgroups was in fact a good estimate of the marginal effect in the population. Not the case here, as the conditional causal effect (LOR) of A is 1.0, which is greater than the true marginal effect of 0.86:

**library**(broom)

tidy(glm(Y ~ A + L , data = dtB, family="binomial"))

## term estimate std.error statistic p.value

## 1 (Intercept) -2.4895846 0.01053398 -236.33836 0

## 2 A 0.9947154 0.01268904 78.39167 0

## 3 L 1.7411358 0.01249180 139.38225 0

This regression estimate for the coefficient of A*A* is a good estimate of the conditional effect in the population (based on the potential outcomes at each level of L*L*):

dtB[, .(LOR = log( odds( mean(Y1) ) / odds( mean(Y0) ) ) ), keyby = L]

## L LOR

## 1: 0 0.9842565

## 2: 1 0.9865561

Of course, ignoring the confounder L*L* is not very useful if we are interested in recovering the marginal effect. The estimate of 1.4 is biased for both the conditional effect and the marginal effect - it is not really useful for anything:

tidy(glm(Y ~ A , data = dtB, family="binomial"))

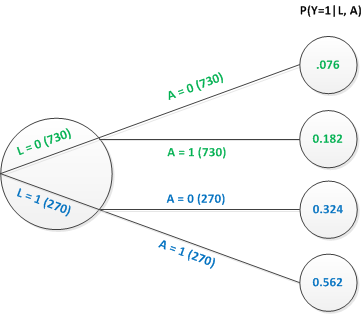
## term estimate std.error statistic p.value

## 1 (Intercept) -2.049994 0.009164085 -223.6987 0

## 2 A 1.433094 0.011723767 122.2384 0

### How weighting reshapes the data …

Here is a simple tree graph that shows the potential outcomes for 1000 individuals (based on the same distributions we’ve been using in our simulation). For 27% of the individuals, L=1*L*=1, for 73% L=0*L*=0. Each individual has a potential outcome under each level of treatment A*A*. So, that is why there are 730 individuals with L=0*L*=0 who are both with and without treatment. Likewise each treatment arm for those with L=0*L*=0 has 270 individuals. We are not double counting.



Both the marginal and conditional estimates that we estimated before using the simulated data can be calculated directly using information from this tree. The conditional effects on the log-odds scale can be calculated as …

LOR\_{A=1 \textbf{ vs } A=0|L = 0} = log \left (\frac{0.182/0.818}{0.076/0.924} \right)=log(2.705) = 0.995*LORA*=1**vs***A*=0∣*L*=0​=*log*(0.076/0.9240.182/0.818​)=*log*(2.705)=0.995

and

LOR\_{A=1 \textbf{ vs } A=0|L = 1} = log \left (\frac{0.562/0.438}{0.324/0.676} \right)=log(2.677) = 0.984*LORA*=1**vs***A*=0∣*L*=1​=*log*(0.324/0.6760.562/0.438​)=*log*(2.677)=0.984

The marginal effect on the log odds scale is estimated marginal probabilities: P(Y=1|A=0)*P*(*Y*=1∣*A*=0) and P(Y=1|A=1)*P*(*Y*=1∣*A*=1). Again, we can take this right from the tree …

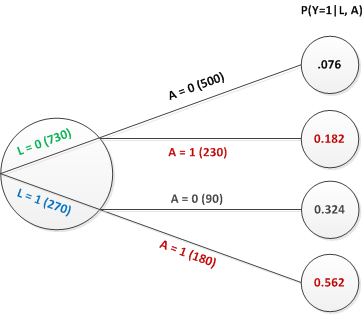
P(Y=1|A=0) = 0.73\times0.076 + 0.27\times0.324 = 0.143*P*(*Y*=1∣*A*=0)=0.73×0.076+0.27×0.324=0.143and

P(Y=1|A=1) = 0.73\times0.182 + 0.27\times0.562 = 0.285*P*(*Y*=1∣*A*=1)=0.73×0.182+0.27×0.562=0.285

Based on these average outcomes (probabilities) by exposure, the marginal log-odds for the sample is:

LOR\_{A=1 \textbf{ vs } A=0} = log \left (\frac{0.285/0.715}{0.143/0.857} \right)=log(2.389) = 0.871*LORA*=1**vs***A*=0​=*log*(0.143/0.8570.285/0.715​)=*log*(2.389)=0.871

Back in the real world of observed data, this is what the tree diagram looks like:



This tree tells us that the probability of exposure A=1*A*=1 is different depending upon that value of L*L*. For L=1*L*=1, P(A=1) = 230/730 = 0.315*P*(*A*=1)=230/730=0.315 and for L=0*L*=0, P(A=1) = 180/270 = 0.667*P*(*A*=1)=180/270=0.667. Because of this disparity, the crude estimate of the effect (ignoring L*L*) is biased for the marginal causal effect:

P(Y=1|A=0) = \frac{500\times0.076 + 90\times0.324}{500+90}=0.114*P*(*Y*=1∣*A*=0)=500+90500×0.076+90×0.324​=0.114

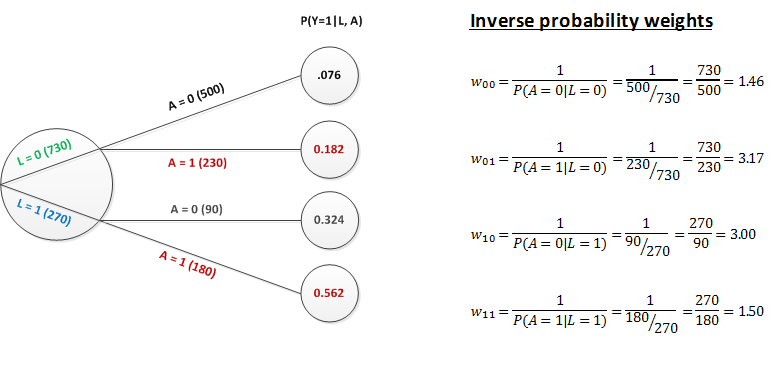
and

P(Y=1|A=1) = \frac{230\times0.182 + 180\times0.562}{230+180}=0.349*P*(*Y*=1∣*A*=1)=230+180230×0.182+180×0.562​=0.349

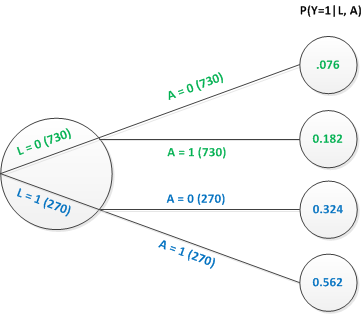
The crude log odds ratio is

LOR\_{A=1 \textbf{ vs } A=0} = log \left (\frac{0.349/0.651}{0.114/0.886} \right)=log(4.170) = 1.420*LORA*=1**vs***A*=0​=*log*(0.114/0.8860.349/0.651​)=*log*(4.170)=1.420

And now we finally get to the weights. As mentioned in the prior post, the IPW is based on the probability of the actual exposure at each level of L*L*: P(A=a | L)*P*(*A*=*a*∣*L*), where a\in(0,1)*a*∈(0,1) (and not on P(A=1|L)*P*(*A*=1∣*L*), the propensity score). Here are the simple weights for each group:



If we apply the weights to each of the respective groups, you can see that the number of individuals in each treatment arm is adjusted to the total number of individuals in the sub-group defined the level of L*L*. For example, if we apply the weight of 3.17 (730/230) to each person observed with L=0*L*=0 and A=1*A*=1, we end up with 230\times3.17=730230×3.17=730. Applying each of the respective weights to the subgroups of L*L* and A*A* results in a new sample of individuals that looks exactly like the one we started out with in the potential outcomes world:



This all works only if we make these two assumptions:P(Y=1|A=0, L=l) = P(Y\_0=1 | A=1, L=l)*P*(*Y*=1∣*A*=0,*L*=*l*)=*P*(*Y*0​=1∣*A*=1,*L*=*l*)andP(Y=1|A=1, L=l) = P(Y\_1=1 | A=0, L=l)*P*(*Y*=1∣*A*=1,*L*=*l*)=*P*(*Y*1​=1∣*A*=0,*L*=*l*)

That is, we can make this claim only under the assumption of no unmeasured confounding.

### Applying IPW to our data

We need to estimate the weights using logistic regression (though other, more flexible methods, can also be used). First, we estimate P(A=1|L)*P*(*A*=1∣*L*) …

exposureModel <- glm(A ~ L, data = dtB, family = "binomial")

dtB[, pA := predict(exposureModel, type = "response")]

Now we can derive an estimate for P(A=a|L=l)*P*(*A*=*a*∣*L*=*l*) and get the weight itself…

# Define two new columns

defB2 <- defDataAdd(varname = "pA\_actual",

formula = "(A \* pA) + ((1 - A) \* (1 - pA))",

dist = "nonrandom")

defB2 <- defDataAdd(defB2, varname = "IPW",

formula = "1/pA\_actual",

dist = "nonrandom")

# Add weights

dtB <- addColumns(defB2, dtB)

dtB[1:6]

## id L Y0 Y1 A Y pA pA\_actual IPW

## 1: 1 0 0 0 0 0 0.3146009 0.6853991 1.459004

## 2: 2 0 0 0 0 0 0.3146009 0.6853991 1.459004

## 3: 3 1 0 1 1 1 0.6682351 0.6682351 1.496479

## 4: 4 0 1 1 1 1 0.3146009 0.3146009 3.178631

## 5: 5 1 0 0 1 0 0.6682351 0.6682351 1.496479

## 6: 6 1 0 0 0 0 0.6682351 0.3317649 3.014183

To estimate the marginal effect on the log-odds scale, we use the function glm with weights specified by IPW. The true value of marginal effect (based on the population-wide potential outcomes) was 0.87 (as we estimated from the potential outcomes directly and from the tree graph). Our estimate here is spot on (but with such a large sample size, this is not so surprising):

tidy(glm(Y ~ A , data = dtB, family="binomial", weights = IPW))

## term estimate std.error statistic p.value

## 1 (Intercept) -1.7879512 0.006381189 -280.1909 0

## 2 A 0.8743154 0.008074115 108.2862 0

It may not seem like a big deal to be able to estimate the marginal effect - we may actually not be interested in it. However, in the next post, I will touch on the issue of estimating causal effects when there are repeated exposures (for example, administering a drug over time) and time dependent confounders that are both affected by prior exposures and affect future exposures and affect the outcome. Under this scenario, it is very difficult if not impossible to control for these confounders - the best we might be able to do is estimate a marginal, population-wide causal effect. That is where weighting will be really useful.

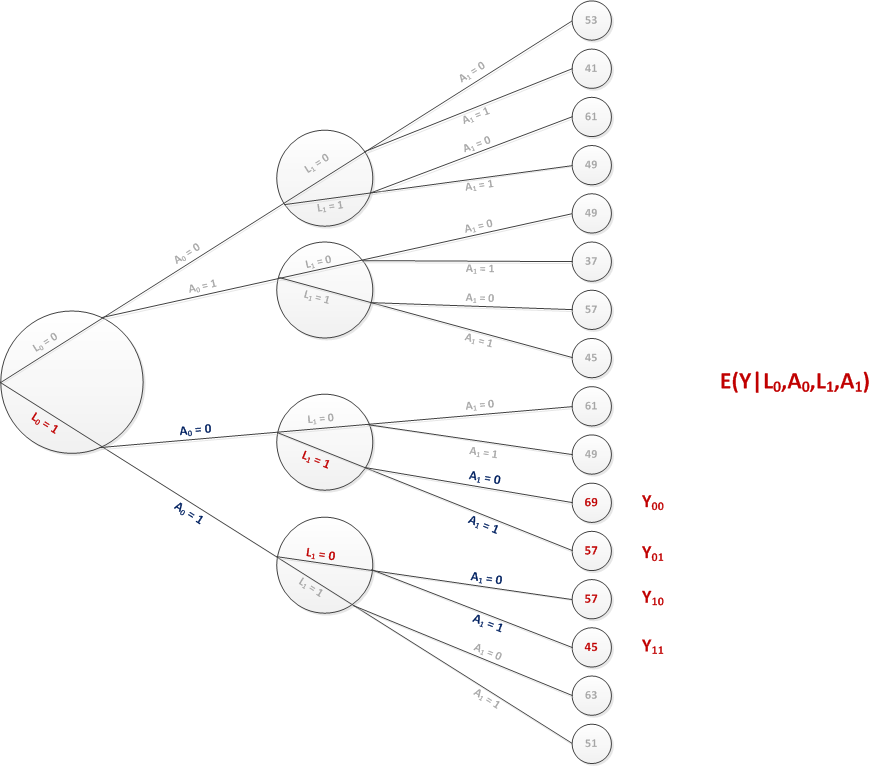
Code Chunks - When there's a fork in the road, take it. Or, taking a look at marginal structural models

### Imagine an experiment …

To understand the MSM, it is actually helpful to think about how a single individual fits into the picture. The tree diagram below literally shows that. The MSM posits a weird experiment where measurements (of L*L*) are collected and treatments (A*A*) are assigned repeatedly until a final outcome is measured. In this experiment, the patient is not just assigned to one treatment arm, but to both! Impossible of course, but that is the world of potential outcomes.

At the start of the experiment, a measurement of L\_0*L*0​ is collected. This sends the patient down the one of the branches of the tree. Since the patient is assigned to both A\_0=0*A*0​=0 and A\_0=1*A*0​=1, she actually heads down two different branches simultaneously. Following the completion of the first treatment period A\_0*A*0​, the second measurement (L\_1*L*1​) is collected. But, two measurements are taken for the patient - one for each branch. The results need not be the same. In fact, if the treatment has an individual-level effect on L\_1*L*1​, then the results will be different for this patient. In the example below, this is indeed the case. Following each of those measurements (in parallel universes), the patient is sent down the next treatment branches (A\_1*A*1​). At this point, the patient finds herself in four branches. At the end of each, the measurement of Y*Y* is taken, and we have four potential outcomes for individual {i}: Y^i\_{00}*Y*00*i*​, Y^i\_{10}*Y*10*i*​, Y^i\_{01}*Y*01*i*​, and Y^i\_{11}*Y*11*i*​.

A different patient will head down different branches based on his own values of L\_0*L*0​ and L\_1*L*1​, and will thus end up with different potential outcomes. (Note: the values represented in the figure are intended to be average values for that particular path.)



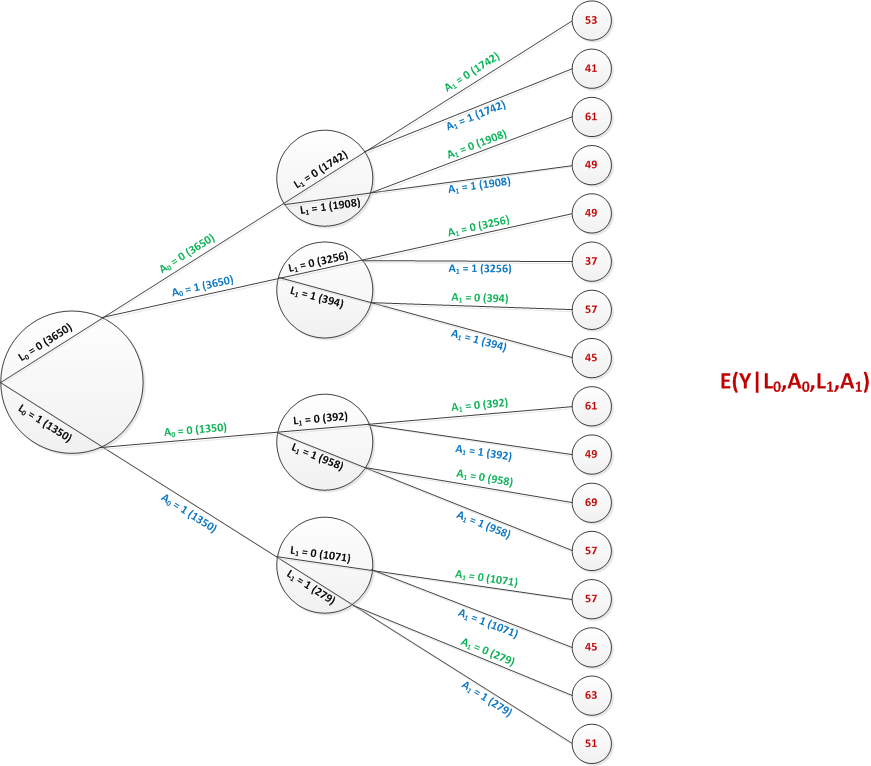
### How do we define the causal effect?

With four potential outcomes rather than two, it is less obvious how to define the causal effect. We could, for example, consider three separate causal effects by comparing each of the treatment “regimes” that include at least one exposure to the intervention to the single regime that leaves the patient entirely unexposed. That is, we could be interested in (at the individual i*i* level) E^i\_1 = Y^i\_{10}-Y^i\_{00}*E*1*i*​=*Y*10*i*​−*Y*00*i*​, E^i\_2 = Y^i\_{01}-Y^i\_{00}*E*2*i*​=*Y*01*i*​−*Y*00*i*​, and E^i\_3 = Y^i\_{11}-Y^i\_{00}*E*3*i*​=*Y*11*i*​−*Y*00*i*​. This is just one possibility; the effects of interest are driven entirely by the research question.

When we have three or four or more intervention periods, the potential outcomes can start to pile up rapidly (we will have 2^n2*n* potential outcomes for a sequence of n*n* treatments.) So, the researcher might want to be judicious in deciding which contrasts to be made. Maybe something like Y\_{1111} - Y\_{0000}*Y*1111​−*Y*0000​, Y\_{0111} - Y\_{0000}*Y*0111​−*Y*0000​, Y\_{0011} - Y\_{0000}*Y*0011​−*Y*0000​, and Y\_{0001} - Y\_{0000}*Y*0001​−*Y*0000​ for a four-period intervention. This would allow us to consider the effect of starting (and never stopping) the intervention in each period compared to never starting the intervention at all. By doing this, though, we would be using only 5 out of the 16 potential outcomes. If the remaining 11 paths are not so rare, we might be ignoring a lot of data.

### The marginal effect

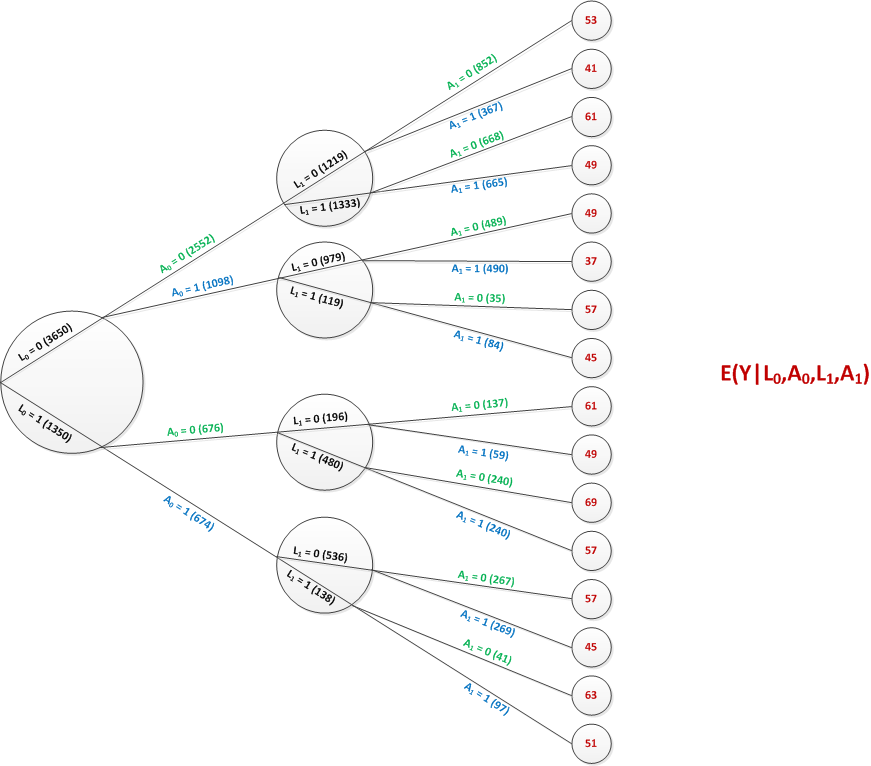
The tree below represents an aggregate set of branches for a sample of 5000 individuals. The sample is initially characterized only by the distribution of L\_0*L*0​. Each individual will go down her own set of four paths, which depend on the starting value of L\_0*L*0​ and how each value of L\_1*L*1​ responds in the context of each treatment arm.



Each individual i*i* (at least in theory) has four potential outcomes: Y^i\_{00}*Y*00*i*​, Y^i\_{10}*Y*10*i*​, Y^i\_{01}*Y*01*i*​, and Y^i\_{11}*Y*11*i*​. Averaging across the sample provides a marginal estimate of each of these potential outcomes. For example, E(Y\_{00})=\sum\_i{Y^i\_{00}}/5000*E*(*Y*00​)=∑*i*​*Y*00*i*​/5000. This can be calculated from the tree as(1742\*53 + 1908\*61 + 392\*61 + 958\*69)/5000 = 59.7(1742∗53+1908∗61+392∗61+958∗69)/5000=59.7Similarly, E(Y\_{11}) = 40.1*E*(*Y*11​)=40.1 The sample average causal effects are estimated using the respective averages of the potential outcomes. For example, E\_3*E*3​ at the sample level would be defined as E(Y\_{11}) - E(Y\_{00}) = 40.1 - 59.7 = -19.6*E*(*Y*11​)−*E*(*Y*00​)=40.1−59.7=−19.6.

### Back in the real world

In reality, there are no parallel universes. Maybe we could come up with an actual randomized experiment to mimic this, but it may be difficult. More likely, we’ll have observed data that looks like this:



Each individual heads down his or her own path, receiving a single treatment at each time point. Since this is not a randomized trial, the probability of treatment is different across different levels of L\_0*L*0​ and L\_1*L*1​ and that L\_0*L*0​ and L\_1*L*1​ are associated with the outcome (i.e. there is confounding).

### Estimating the marginal effects

In the previous posts in this series, I provided some insight as to how we might justify using observed data only to estimate these sample-wide average potential outcomes. The most important assumption is that when we have measured all confounders, we may be able to say, for example, E(Y\_{01}) = E(Y | A\_0 = 0 \ \&amp; \ A\_1 = 1 )*E*(*Y*01​)=*E*(*Y*∣*A*0​=0 & *A*1​=1). The potential outcome for everyone in the sample is equal to the observed outcome for the subgroup who actually followed the particular path that represents that potential outcome. We will make the same assumption here.

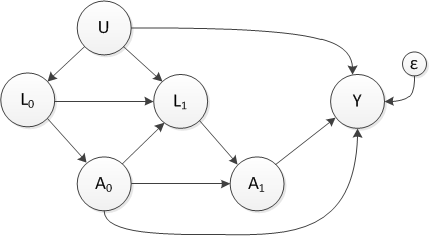
At the start of this post, I argued that given the complex nature of the data generating process (in particular given that L\_1*L*1​ is both a mediator and confounder), it is challenging to get unbiased estimates of the intervention effects. One way to do this with marginal structural models (another way is using [g-computation](https://academic.oup.com/aje/article/173/7/731/104142), but I won’t talk about that here). Inverse probability weighting converts the observed tree graph from the real world to the marginal tree graph so that we can estimate sample-wide average (marginal) potential outcomes as an estimate for some population causal effects.

In this case, the inverse probability weight is calculated asIPW = \frac{1}{P(A\_0=a\_0 | L\_0=l\_0) \times P(A\_1=a\_1 | L\_0=l\_0, A\_0=a\_0, L\_1=l\_1)}*IPW*=*P*(*A*0​=*a*0​∣*L*0​=*l*0​)×*P*(*A*1​=*a*1​∣*L*0​=*l*0​,*A*0​=*a*0​,*L*1​=*l*1​)1​In practice, we estimate both probabilities using logistic regression or some other modeling technique. But here, we can read the probabilities off the tree graph. For example, if we are interested in the weight associated with individuals observed with L\_0=1, A\_0=0, L\_1=0, \textbf{and } A\_1=1*L*0​=1,*A*0​=0,*L*1​=0,**and***A*1​=1, the probabilities areP(A\_0 = 0 | L\_0=1) = \frac{676}{1350}=0.50*P*(*A*0​=0∣*L*0​=1)=1350676​=0.50andP(A\_1=1 | L\_0=1, A\_0=0, L\_1=0) = \frac{59}{196} = 0.30*P*(*A*1​=1∣*L*0​=1,*A*0​=0,*L*1​=0)=19659​=0.30

So, the inverse probability weight for these individuals isIPW = \frac{1}{0.50 \times 0.30} = 6.67*IPW*=0.50×0.301​=6.67For the 59 individuals that followed this pathway, the weighted number is 59 \times 6.67 = 39359×6.67=393. In the marginal world of parallel universes, there were 394 individuals.

### Simulating data from an MSM

Before I jump into the simulation, I do want to reference a paper by [Havercroft and Didelez](http://onlinelibrary.wiley.com/doi/10.1002/sim.5472/full) that describes in great detail how to generate data from a MSM with time-dependent confounding. It turns out that the data can’t be generated exactly using the intial DAG (presented above), but rather needs to come from something like this:



where U*U* is an unmeasured, maybe latent, covariate. The observed data (that ignores U*U*) will indeed have a DAG that looks like the one that we started with.

When doing simulations with potential outcomes, we can generate all the potential outcomes for each individual using a parallel universe approach. The observed data (treatment choices and observed outcomes) are generated separately. The advantage of this is that we can confirm the true causal effects because we have actually generated potential outcomes. The disadvantage is that the code is considerably more complicated and the quantity of data generated grows. The situation is not so bad with just two treatment periods, but the size of the data increases exponentially with the number of treatments: as I mentioned earlier, there will be 2^n2*n* potential outcomes for each individual.

Alternatively, we can just generate the observed data directly. Since we know the true causal parameters we actually “know” the causal effects and can compare our estimates.

I will go through the convoluted approach because I think it clarifies (at least a bit) what is going on. As an addendum, I will show how all of this could be done in a few lines of code if we take the second approach …

**library**(broom)

**library**(simstudy)

# define U, e and L0

defA0 <- defData(varname = "U", formula = "0;1", dist = "uniform")

defA0 <- defData(defA0, varname = "e", formula = 0,

variance = 4, dist = "normal")

defA0<- defData(defA0, varname = "L0", formula = "-2.66+ 3\*U",

dist = "binary", link = "logit")

# generate the data

set.seed(1234)

dtA0 <- genData(n = 50000, defA0)

dtA0[1:6]

## id U e L0

## 1: 1 0.1137034 -3.5951796 0

## 2: 2 0.6222994 -0.5389197 0

## 3: 3 0.6092747 1.0675660 0

## 4: 4 0.6233794 -0.7226909 1

## 5: 5 0.8609154 0.8280401 0

## 6: 6 0.6403106 3.3532399 0

Now we need to create the two parallel universes - assigning each individual to both treatments. simstudy has a function addPeriods to generate longitudinal data. I am not doing that here, but can generate 2-period data and change the name of the “period” field to “A0”.

dtA0 <- addPeriods(dtA0, 2)

setnames(dtA0, "period", "A0")

dtA0[1:6]

## id A0 U e L0 timeID

## 1: 1 0 0.1137034 -3.5951796 0 1

## 2: 1 1 0.1137034 -3.5951796 0 2

## 3: 2 0 0.6222994 -0.5389197 0 3

## 4: 2 1 0.6222994 -0.5389197 0 4

## 5: 3 0 0.6092747 1.0675660 0 5

## 6: 3 1 0.6092747 1.0675660 0 6

Now we are ready to randomly assign a value of L\_1*L*1​. The probability is lower for cases where A\_0 = 1*A*0​=1, so individuals themselves may have different values of L\_1*L*1​ in the alternative paths.

# generate L1 as a function of U, L0, and A0

addA0 <- defDataAdd(varname = "L1",

formula = "-1.2 + 3\*U + 0.2\*L0 - 2.5\*A0",

dist= "binary", link="logit")

dtA0 <- addColumns(addA0, dtOld = dtA0)

dtA0[1:6]

## id A0 U e L0 timeID L1

## 1: 1 0 0.1137034 -3.5951796 0 1 0

## 2: 1 1 0.1137034 -3.5951796 0 2 0

## 3: 2 0 0.6222994 -0.5389197 0 3 1

## 4: 2 1 0.6222994 -0.5389197 0 4 0

## 5: 3 0 0.6092747 1.0675660 0 5 0

## 6: 3 1 0.6092747 1.0675660 0 6 0

# L1 is clearly a function of A0

dtA0[, .(prob\_L1 = mean(L1)), keyby = .(L0,A0)]

## L0 A0 prob\_L1

## 1: 0 0 0.5238369

## 2: 0 1 0.1080039

## 3: 1 0 0.7053957

## 4: 1 1 0.2078551

Now we create two additional parallel universes for treatment A\_1*A*1​ and the potential outcomes. This will result in four records per individual:

dtA1 <- addPeriods(dtA0, 2)

setnames(dtA1, "period", "A1")

addA1 <- defDataAdd(varname = "Y\_PO",

formula = "39.95 + U\*40 - A0 \* 8 - A1 \* 12 + e",

dist = "nonrandom")

dtA1 <- addColumns(addA1, dtA1)

dtA1[1:8]

## id A1 A0 U e L0 timeID L1 Y\_PO

## 1: 1 0 0 0.1137034 -3.5951796 0 1 0 40.90296

## 2: 1 0 1 0.1137034 -3.5951796 0 2 0 32.90296

## 3: 1 1 0 0.1137034 -3.5951796 0 3 0 28.90296

## 4: 1 1 1 0.1137034 -3.5951796 0 4 0 20.90296

## 5: 2 0 0 0.6222994 -0.5389197 0 5 1 64.30306

## 6: 2 0 1 0.6222994 -0.5389197 0 6 0 56.30306

## 7: 2 1 0 0.6222994 -0.5389197 0 7 1 52.30306

## 8: 2 1 1 0.6222994 -0.5389197 0 8 0 44.30306

Not surprisingly, the estimates for the causal effects mirror the parameters we used to generate the Y*Y*’s above …

# estimate for Y\_00 is close to what we estimated from the tree

Y\_00 <- dtA1[A0 == 0 & A1 == 0, mean(Y\_PO)]

Y\_00

## [1] 59.96619

Y\_10 <- dtA1[A0 == 1 & A1 == 0, mean(Y\_PO)]

Y\_01 <- dtA1[A0 == 0 & A1 == 1, mean(Y\_PO)]

Y\_11 <- dtA1[A0 == 1 & A1 == 1, mean(Y\_PO)]

# estimate 3 causal effects

c(Y\_10 - Y\_00, Y\_01 - Y\_00, Y\_11 - Y\_00)

## [1] -8 -12 -20

Now that we’ve generated the four parallel universes with four potential outcomes per individual, we will generate an observed treatment sequence and measurements of the L*L*’s and Y*Y* for each individual. The observed data set will have a single record for each individual:

dt <- dtA1[A0 == 0 & A1 == 0, .(id, L0)]

dt

## id L0

## 1: 1 0

## 2: 2 0

## 3: 3 0

## 4: 4 1

## 5: 5 0

## ---

## 49996: 49996 1

## 49997: 49997 0

## 49998: 49998 1

## 49999: 49999 0

## 50000: 50000 1

A\_0*A*0​ is a function of L\_0*L*0​:

dtAdd <- defDataAdd(varname = "A0",

formula = "0.3 + L0 \* 0.2", dist = "binary" )

dt <- addColumns(dtAdd, dt)

dt[, mean(A0), keyby= L0]

## L0 V1

## 1: 0 0.3015964

## 2: 1 0.4994783

Now, we need to pull the appropriate value of L\_1*L*1​ from the original data set that includes both possible values for each individual. The value that gets pulled will be based on the observed value of A\_0*A*0​:

setkeyv(dt, c("id", "A0"))

setkeyv(dtA1, c("id", "A0"))

dt <- merge(dt, dtA1[, .(id, A0, L1, A1) ], by = c("id", "A0"))

dt <- dt[A1 == 0, .(id, L0, A0, L1)]

dt

## id L0 A0 L1

## 1: 1 0 1 0

## 2: 2 0 1 0

## 3: 3 0 0 0

## 4: 4 1 1 1

## 5: 5 0 0 1

## ---

## 49996: 49996 1 1 0

## 49997: 49997 0 1 0

## 49998: 49998 1 1 0

## 49999: 49999 0 0 1

## 50000: 50000 1 0 0

Finally, we generate A\_1*A*1​ based on the observed values of A\_0*A*0​ and L\_1*L*1​, and select the appropriate value of Y*Y*:

dtAdd <- defDataAdd(varname = "A1",

formula = "0.3 + L1 \* 0.2 + A0 \* .2", dist = "binary")

dt <- addColumns(dtAdd, dt)

# merge to get potential outcome that matches actual path

setkey(dt, id, L0, A0, L1, A1)

setkey(dtA1, id, L0, A0, L1, A1)

dtObs <- merge(dt, dtA1[,.(id, L0, A0, L1, A1, Y = Y\_PO)])

dtObs

## id L0 A0 L1 A1 Y

## 1: 1 0 1 0 0 32.90296

## 2: 2 0 1 0 1 44.30306

## 3: 3 0 0 0 1 53.38856

## 4: 4 1 1 1 1 44.16249

## 5: 5 0 0 1 0 75.21466

## ---

## 49996: 49996 1 1 0 0 74.09161

## 49997: 49997 0 1 0 0 50.26162

## 49998: 49998 1 1 0 0 73.29376

## 49999: 49999 0 0 1 0 52.96703

## 50000: 50000 1 0 0 0 57.13109

If we do a crude estimate of the causal effects using the unadjusted observed data, we know we are going to get biased estimates (remember the true causal effects are -8, -12, and -20):

Y\_00 <- dtObs[A0 == 0 & A1 == 0, mean(Y)]

Y\_10 <- dtObs[A0 == 1 & A1 == 0, mean(Y)]

Y\_01 <- dtObs[A0 == 0 & A1 == 1, mean(Y)]

Y\_11 <- dtObs[A0 == 1 & A1 == 1, mean(Y)]

c(Y\_10 - Y\_00, Y\_01 - Y\_00, Y\_11 - Y\_00)

## [1] -6.272132 -10.091513 -17.208856

This biased result is confirmed using an unadjusted regression model:

lmfit <- lm(Y ~ A0 + A1, data = dtObs)

tidy(lmfit)

## term estimate std.error statistic p.value

## 1 (Intercept) 58.774695 0.07805319 753.00828 0

## 2 A0 -6.681213 0.10968055 -60.91520 0

## 3 A1 -10.397080 0.10544448 -98.60241 0

Now, shouldn’t we do better if we adjust for the confounders? I don’t think so - the parameter estimate for A\_0*A*0​ should be close to 88; the estimate for A\_1*A*1​ should be approximately 1212, but this is not the case, at least not for both of the estimates:

lmfit <- lm(Y ~ L0 + L1 + A0 + A1, data = dtObs)

tidy(lmfit)

## term estimate std.error statistic p.value

## 1 (Intercept) 53.250244 0.08782653 606.31157 0

## 2 L0 7.659460 0.10798594 70.93016 0

## 3 L1 8.203983 0.10644683 77.07119 0

## 4 A0 -4.369547 0.11096204 -39.37875 0

## 5 A1 -12.037274 0.09592735 -125.48323 0

Maybe if we just adjust for L\_0*L*0​ or L\_1*L*1​?

lmfit <- lm(Y ~ L1 + A0 + A1, data = dtObs)

tidy(lmfit)

## term estimate std.error statistic p.value

## 1 (Intercept) 54.247394 0.09095074 596.44808 0.000000e+00

## 2 L1 9.252919 0.11059038 83.66839 0.000000e+00

## 3 A0 -2.633981 0.11354466 -23.19775 2.031018e-118

## 4 A1 -12.016545 0.10063687 -119.40499 0.000000e+00

lmfit <- lm(Y ~ L0 + A0 + A1, data = dtObs)

tidy(lmfit)

## term estimate std.error statistic p.value

## 1 (Intercept) 57.036320 0.07700591 740.67459 0

## 2 L0 8.815691 0.11311215 77.93761 0

## 3 A0 -8.150706 0.10527255 -77.42480 0

## 4 A1 -10.632238 0.09961593 -106.73231 0

So, none of these approaches seem to work. This is where IPW can provide a solution. First we estimate the treatment/exposure models, then we estimate the IPW, and finally we use weighted regression or just estimate weighted average outcomes directly (we’d have to bootstrap here if we want standard errors for the simple average approach):

# estimate P(A0|L0) and P(A1|L0, A0, L1)

fitA0 <- glm(A0 ~ L0, data = dtObs, family=binomial)

fitA1 <- glm(A1 ~ L0 + A0 + L1, data = dtObs, family=binomial)

dtObs[, predA0 := predict(fitA0, type = "response")]

dtObs[, predA1 := predict(fitA1, type = "response")]

# function to convert propenisty scores to IPW

getWeight <- **function**(predA0, actA0, predA1, actA1) {

predActA0 <- actA0\*predA0 + (1-actA0)\*(1-predA0)

predActA1 <- actA1\*predA1 + (1-actA1)\*(1-predA1)

p <- predActA0 \* predActA1

**return**(1/p)

}

dtObs[, wgt := getWeight(predA0, A0, predA1, A1)]

# fit weighted model

lmfit <- lm(Y ~ A0 + A1, weights = wgt, data = dtObs)

tidy(lmfit)

## term estimate std.error statistic p.value

## 1 (Intercept) 59.982379 0.09059652 662.08257 0

## 2 A0 -7.986486 0.10464257 -76.32157 0

## 3 A1 -12.051805 0.10464258 -115.17114 0

# non-parametric estimation

Y\_00 <- dtObs[A0 == 0 & A1 == 0, weighted.mean(Y, wgt)]

Y\_10 <- dtObs[A0 == 1 & A1 == 0, weighted.mean(Y, wgt)]

Y\_01 <- dtObs[A0 == 0 & A1 == 1, weighted.mean(Y, wgt)]

Y\_11 <- dtObs[A0 == 1 & A1 == 1, weighted.mean(Y, wgt)]

round(c(Y\_10 - Y\_00, Y\_01 - Y\_00, Y\_11 - Y\_00), 2)

## [1] -8.04 -12.10 -20.04

## Addendum

This post has been quite long, so I probably shouldn’t go on. But, I wanted to show that we can do the data generation in a much less convoluted way that avoids generating all possible forking paths for each individual. As always in simstudy the data generation process needs us to create a data definition table. In this example, I’ve created that table in an external file named msmDef.csv. In the end, this simpler approach has reduced necessary code by about 95%.

defMSM <- defRead("msmDef.csv")

defMSM

## varname formula variance dist link

## 1: U 0;1 0 uniform identity

## 2: e 0 9 normal identity

## 3: L0 -2.66+ 3\*U 0 binary logit

## 4: A0 0.3 + L0 \* 0.2 0 binary identity

## 5: L1 -1.2 + 3\*U + 0.2\*L0 - 2.5\*A0 0 binary logit

## 6: A1 0.3 + L1\*0.2 + A0\*0.2 0 binary identity

## 7: Y 39.95 + U\*40 - A0\*8 - A1\*12 + e 0 nonrandom identity

dt <- genData(50000, defMSM)

fitA0 <- glm(A0 ~ L0, data = dt, family=binomial)

fitA1 <- glm(A1 ~ L0 + A0 + L1, data = dt, family=binomial)

dt[, predA0 := predict(fitA0, type = "response")]

dt[, predA1 := predict(fitA1, type = "response")]

dt[, wgt := getWeight(predA0, A0, predA1, A1)]

tidy(lm(Y ~ A0 + A1, weights = wgt, data = dt))

## term estimate std.error statistic p.value

## 1 (Intercept) 60.061609 0.09284532 646.89967 0

## 2 A0 -7.931715 0.10715916 -74.01808 0

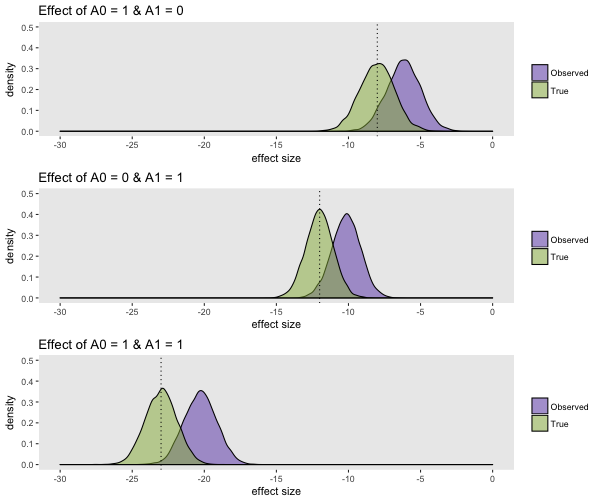
## 3 A1 -12.131829 0.10715900 -113.21335 0

### Does the MSM still work with more complicated effects?

In conclusion, I wanted to show that MSMs still function well even when the causal effects do not follow a simple linear pattern. (And I wanted to be able to end with a figure.) I generated 10000 datasets of 900 observations each, and calculated the crude and marginal causal effects after each iteration. The true treatment effects are described by an “interaction” between A\_0*A*0​ and A\_1*A*1​. If treatment is received in both periods (i.e. A\_0=1*A*0​=1 and A\_1=1*A*1​=1), there is an extra additive effect:

Y = 39.95 + U\*40 - A0\*8 - A1\*12 - A0\*A1\*3 + e*Y*=39.95+*U*∗40−*A*0∗8−*A*1∗12−*A*0∗*A*1∗3+*e*

The purple density is the (biased) observed estimates and the green density is the (unbiased) IPW-based estimate. Again the true causal effects are -8, -12, and -23:



In this (admittedly thoroughly made-up though not entirely implausible) network diagram, the *test score* outcome is a collider, influenced by a *test preparation* class and *socio-economic status* (SES). In particular, both the test prep course and high SES are related to the probability of having a high test score. One might expect an arrow of some sort to connect SES and the test prep class; in this case, participation in test prep is randomized so there is no causal link

Code Chunks – Complier average causal effect? Exploring what we learn from an RCT with participants who don't do what they are told

### Simulating data

The data simulation will be based on generating potential outcomes. Observed outcomes will be a function of randomization group and complier status.

options(digits = 3)

**library**(data.table)

**library**(simstudy)

**library**(ggplot2)

# Status :

# 1 = A(lways taker)

# 2 = N(ever taker)

# 3 = C(omplier)

def <- defDataAdd(varname = "Status",

formula = "0.20; 0.40; 0.40", dist = "categorical")

# potential outcomes (PO) for intervention

def <- defDataAdd(def, varname = "M0",

formula = "(Status == 1) \* 1", dist = "nonrandom")

def <- defDataAdd(def, varname = "M1",

formula = "(Status != 2) \* 1", dist = "nonrandom")

# observed intervention status based on randomization and PO

def <- defDataAdd(def, varname = "m",

formula = "(z==0) \* M0 + (z==1) \* M1", dist = "nonrandom")

# potential outcome for Y (depends on potential outcome for M)

set.seed(888)

dt <- genData(2000)

dt <- trtAssign(dt, n=2, grpName = "z")

dt <- addColumns(def, dt)

# using data functions here, not simstudy - I need add

# this functionality to simstudy

dt[, AStatus := factor(Status,

labels = c("Always-taker","Never-taker", "Complier"))]

# potential outcomes depend on group status - A, N, or C

dt[Status == 1, Y0 := rnorm(.N, 1.0, sqrt(0.25))]

dt[Status == 2, Y0 := rnorm(.N, 0.0, sqrt(0.36))]

dt[Status == 3, Y0 := rnorm(.N, 0.1, sqrt(0.16))]

dt[Status == 1, Y1 := rnorm(.N, 1.0, sqrt(0.25))]

dt[Status == 2, Y1 := rnorm(.N, 0.0, sqrt(0.36))]

dt[Status == 3, Y1 := rnorm(.N, 0.9, sqrt(0.49))]

# observed outcome function of actual treatment

dt[, y := (m == 0) \* Y0 + (m == 1) \* Y1]

dt

## id z Status M0 M1 m AStatus Y0 Y1 y

## 1: 1 1 3 0 1 1 Complier 0.5088 0.650 0.6500

## 2: 2 1 3 0 1 1 Complier 0.1503 0.729 0.7292

## 3: 3 1 2 0 0 0 Never-taker 1.4277 0.454 1.4277

## 4: 4 0 3 0 1 0 Complier 0.6393 0.998 0.6393

## 5: 5 0 1 1 1 1 Always-taker 0.6506 1.927 1.9267

## ---

## 1996: 1996 0 3 0 1 0 Complier -0.9554 0.114 -0.9554

## 1997: 1997 0 3 0 1 0 Complier 0.0366 0.903 0.0366

## 1998: 1998 1 3 0 1 1 Complier 0.3606 1.098 1.0982

## 1999: 1999 1 3 0 1 1 Complier 0.6651 1.708 1.7082

## 2000: 2000 0 3 0 1 0 Complier 0.2207 0.531 0.2207

The plot shows outcomes y*y* for the two randomization groups. The ITT estimate would be based on an average of all the points in group, regardless of color or shape. The difference between the average of the black circles in the two groups represents the CACE.

ggplot(data=dt, aes(y=y, x = factor(z, labels = c("Assigned to control",

"Assigned to treatment")))) +

geom\_jitter(aes(shape=factor(m, labels = c("No treatment", "Treatment")),

color=AStatus),

width = 0.35) +

scale\_shape\_manual(values = c(1,19)) +

scale\_color\_manual(values = c("#e1d07d", "#7d8ee1", "grey25")) +

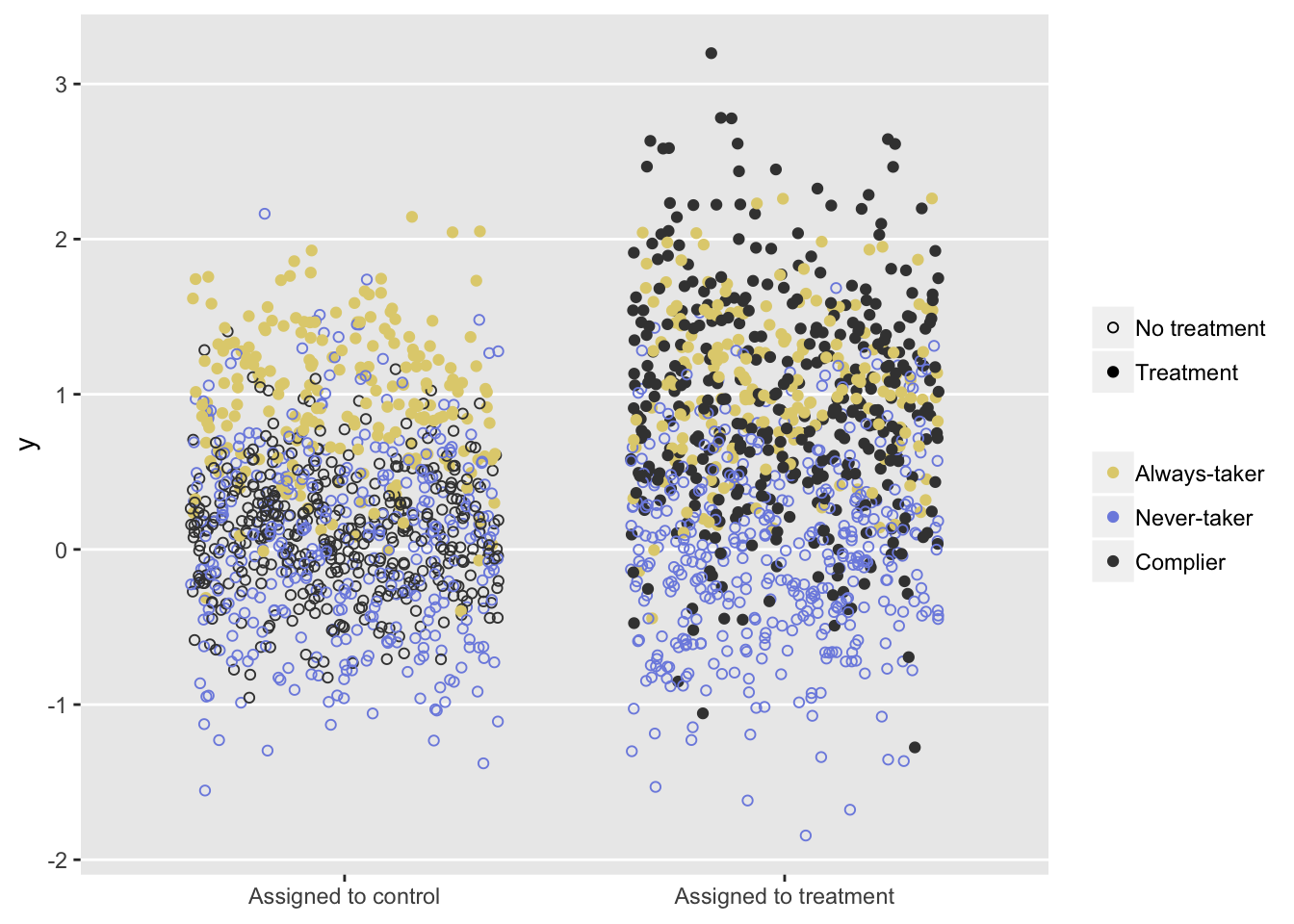
scale\_y\_continuous(breaks = seq(-3, 3, 1), labels = seq(-3, 3, 1)) +

theme(legend.title = element\_blank(),

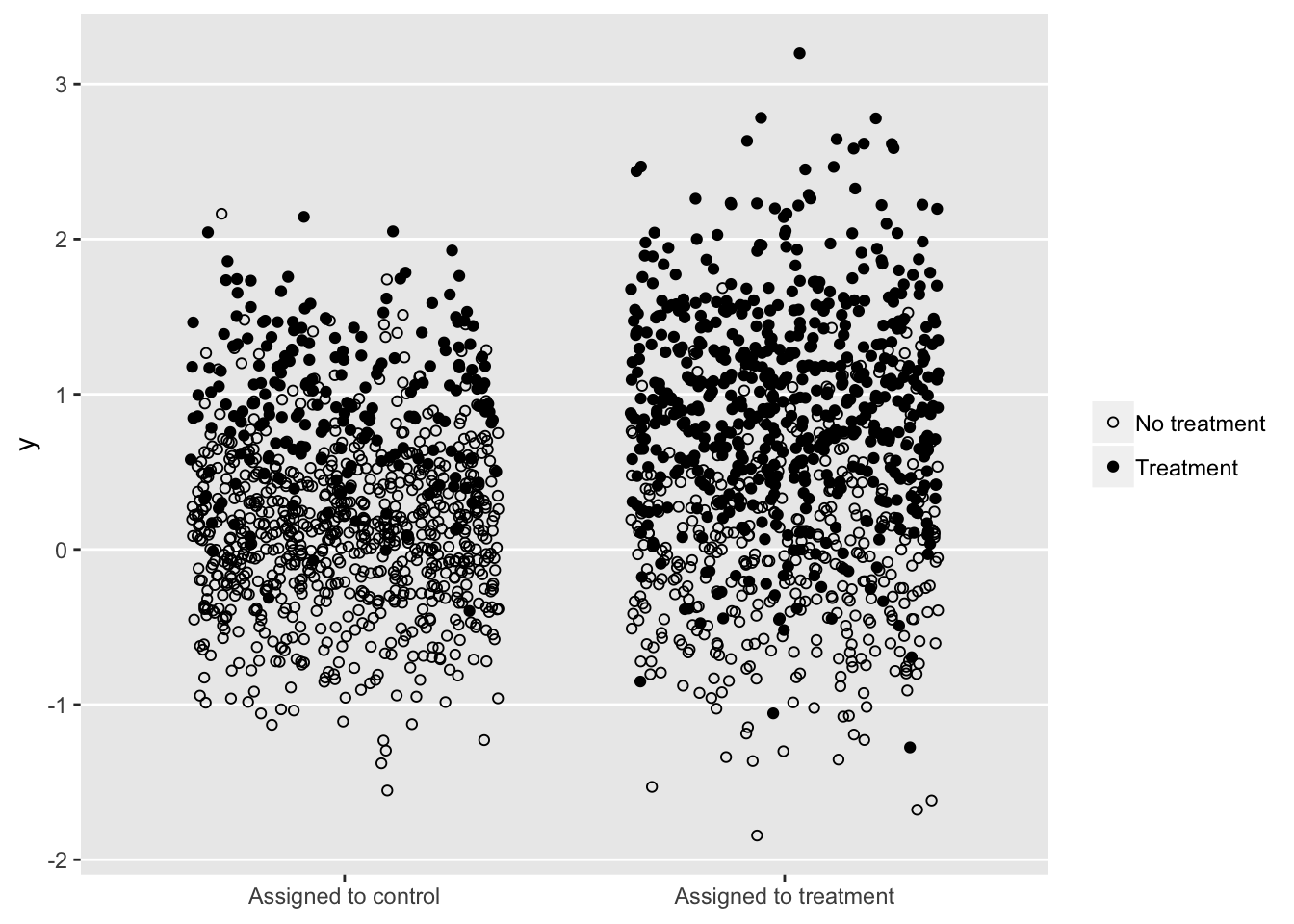
axis.title.x = element\_blank(),

panel.grid.minor.y = element\_blank(),

panel.grid.major.x = element\_blank())



In the real world, we cannot see the colors, yet we need to estimate as if we do, or at least use a method to bypasses that need:



### Estimating CACE using observed data

The challenge is to estimate the CACE using observed data only, since that is all we have (along with a couple of key assumptions). We start of by claiming that the average causal effect of treatment **assignment** (ACE*ACE*) is a weighted average of the three sub-populations of compliers, never-takers, and always-takers:

ACE = \pi\_C \times CACE + \pi\_N \times NACE + \pi\_A \times AACE,*ACE*=*πC*​×*CACE*+*πN*​×*NACE*+*πA*​×*AACE*,where CACE*CACE* is the average causal effect of treatment assignment for the subset of those in the sample who are compliers, NACE*NACE* is the average causal effect of treatment assignment for the subset who are never-takers, and AACE*AACE* is the average causal effect for those who are always-takers. \pi\_C*πC*​, \pi\_N*πN*​, and \pi\_A*πA*​ represent the sample proportions of compliers, never-takers, and always-takers, respectively.

A key assumption often made to estimate CACE*CACE* is known as the exclusion restriction: treatment assignment has an effect on the outcome only if it changes the actual treatment taken. (A second key assumption is that there are no deniers, or folks who do the opposite of what they are told. This is called the monotonicity assumption.) This exclusion restriction implies that both NACE=0*NACE*=0 and AACE=0*AACE*=0, since in both cases the treatment received is the same regardless of treatment assignment. In that case, we can re-write the equality as

ACE = \pi\_C \times CACE,*ACE*=*πC*​×*CACE*,

and finally with a little re-arranging,

CACE = \frac{ACE}{\pi\_C}.*CACE*=*πC*​*ACE*​.So, in order estimate CACE*CACE*, we need to be able to estimate ACE*ACE* and \pi\_C*πC*​. Fortunately, we are in a position to do this. Since this is a randomized trial, the average causal effect of treatment assignment is just the difference in observed outcomes for the two treatment assignment groups:

ACE = E[Y | Z = 1] - E[Y | Z = 0]*ACE*=*E*[*Y*∣*Z*=1]−*E*[*Y*∣*Z*=0]This also happens to be the intention-to-treat ) (ITT*ITT*) estimate.

\pi\_C*πC*​ is a little harder, but in this simplified scenario, not that hard. We just need to follow a little logic: for the control group, we can identify the always-takers (they’re the ones who actually receive the treatment), so we know \pi\_A*πA*​ for the the control group. This can be estimated as P(M=1|Z=0)*P*(*M*=1∣*Z*=0). And, since the study was randomized, the distribution of always-takers in the treatment group must be the same. So, we can use \pi\_A*πA*​ estimated from the control group as an estimate for the treatment group.

For the treatment group, we know that \pi\_C + \pi\_A = P(M = 1 | Z = 1)*πC*​+*πA*​=*P*(*M*=1∣*Z*=1). That is everyone who receives treatment in the treatment group is either a complier or always-taker. With this, we can say

\pi\_C = P(M=1 | Z = 1) - \pi\_A.*πC*​=*P*(*M*=1∣*Z*=1)−*πA*​.

But, of course, we argued above that we can estimate \pi\_A*πA*​ as P(M=1|Z=0)*P*(*M*=1∣*Z*=0). So, finally, we have

\pi\_C = P(M=1 | Z = 1) - P(M=1|Z=0).*πC*​=*P*(*M*=1∣*Z*=1)−*P*(*M*=1∣*Z*=0).This gives us a method of moments estimator for CACE*CACE* from observed data:

CACE = \frac{ACE}{\pi\_C} = \frac{E[Y | Z = 1] - E[Y | Z = 0]}{P(M=1 | Z = 1) - P(M=1|Z=0)}.*CACE*=*πC*​*ACE*​=*P*(*M*=1∣*Z*=1)−*P*(*M*=1∣*Z*=0)*E*[*Y*∣*Z*=1]−*E*[*Y*∣*Z*=0]​.

## The simulated estimate

ACE <- dt[z==1, mean(y)] - dt[z==0, mean(y)] # Also ITT

ACE

## [1] 0.307

pi\_C <- dt[z==1, mean(m)] - dt[z==0, mean(m)] # strength of instrument

pi\_C

## [1] 0.372

truth <- dt[AStatus == "Complier", mean(Y1 - Y0)]

truth

## [1] 0.81

ACE/pi\_C

## [1] 0.826

A method quite commonly used to analyze non-compliance is the instrumental variable model estimated with two-staged least squares regression. The R package ivpack is one of several that facilitates this type of analysis. A discussion of this methodology far exceeds the scope of this post. In any case, we can see that in this simple example, the IV estimate is the same as the method of moments estimator (by looking at the coefficient estimate of m).

**library**(ivpack)

ivmodel <- ivreg(formula = y ~ m | z, data = dt, x = TRUE)

summary(ivmodel)

##

## Call:

## ivreg(formula = y ~ m | z, data = dt, x = TRUE)

##

## Residuals:

## Min 1Q Median 3Q Max

## -2.19539 -0.36249 0.00248 0.35859 2.27902

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 0.0932 0.0302 3.09 0.002 \*\*

## m 0.8262 0.0684 12.08 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 0.569 on 1998 degrees of freedom

## Multiple R-Squared: 0.383, Adjusted R-squared: 0.383

## Wald test: 146 on 1 and 1998 DF, p-value: <2e-16

The researcher who carried out the randomization had a hypothesis that test prep actually is detrimental to college success down the road, because it de-emphasizes deep thinking in favor of wrote memorization. In reality, it turns out that the course and subsequent college success are not related, indicated by an *absence* of a connection between the course and the long term outcome.

**Simulate data**

We can simulate data from this hypothetical world (using functions from package simstudy):

# define data

library(simstudy)

defCollide <- defData(varname = "SES",

formula = "0;1",

dist = "uniform")

defCollide <- defData(defCollide, varname = "testPrep",

formula = 0.5,

dist = "binary")

defCollide <- defData(defCollide, varname = "highScore",

formula = "-1.2 + 3\*SES + 3\*testPrep",

dist = "binary", link="logit")

defCollide <- defData(defCollide, varname = "successMeasure",

formula = "20 + SES\*40", variance = 9,

dist = "normal")

defCollide

## varname formula variance dist link

## 1: SES 0;1 0 uniform identity

## 2: testPrep 0.5 0 binary identity

## 3: highScore -1.2 + 3\*SES + 3\*testPrep 0 binary logit

## 4: successMeasure 20 + SES\*40 9 normal identity

# generate data

set.seed(139)

dt <- genData(1500, defCollide)

dt[1:6]

## id SES testPrep highScore successMeasure

## 1: 1 0.52510665 1 1 40.89440

## 2: 2 0.31565690 0 1 34.72037

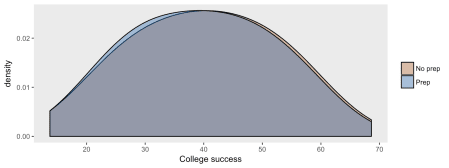
## 3: 3 0.47978492 1 1 41.79532

## 4: 4 0.19114934 0 0 30.05569

## 5: 5 0.06889896 0 0 21.28575

## 6: 6 0.10139604 0 0 21.30306

We can see that the distribution of the long-term (continuous) success outcome is the same for those who are randomized to test prep compared to those who are not, indicating there is no causal relationship between the test and the college outcome:



An unadjusted linear model leads us to the same conclusion, since the parameter estimate representing the treatment effect is quite small (and the hypothesis test is not statistically significant):

library(broom)

rndtidy( lm(successMeasure ~ testPrep, data = dt))

## term estimate std.error statistic p.value

## 1 (Intercept) 40.112 0.44 91.209 0.000

## 2 testPrep -0.495 0.61 -0.811 0.418

**But, don’t we need to adjust for some measure of intellectual ability?**

Or so the researcher might ask after looking at the initial results, questioning the model. He believes that differences in ability could be related to future outcomes. While this may be the case, the question isn’t about ability but the impact of test prep. Based on his faulty logic, the researcher decides to fit a second model and control for the test score that followed the experiment. And this is where things go awry. Take a look at the following model where there appears to be a relationship between test prep and college success after controlling for the test score:

# adjusted model

rndtidy( lm(successMeasure ~ highScore + testPrep, data = dt))

## term estimate std.error statistic p.value

## 1 (Intercept) 35.525 0.619 57.409 0

## 2 highScore 8.027 0.786 10.207 0

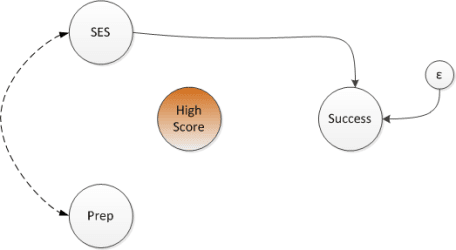
## 3 testPrep -3.564 0.662 -5.380 0

It does indeed appear that the test prep course is causing problems for real learning in college later on!

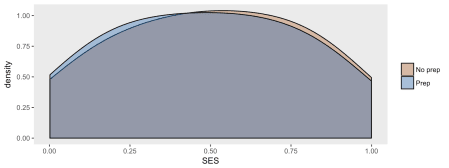
**What is going on?**

Because the test score (here I am treating it as binary – either a high score or not), is related to both SES and test prep, the fact that someone does well on the test is due either to the fact that the student took the course, has high SES, or both. But, let’s consider the students who are possibly high SES or maybe took the course, but not not both, ***and*** who had a high test score. If a student is low SES, she probably took the course, or if she did not take the course, she is probably high SES. So, within the group that scored well, SES and the probability of taking the course are slightly negatively correlated.

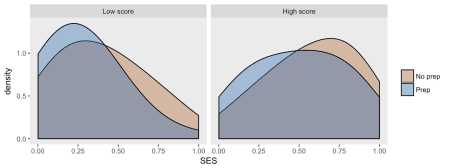
If we “control” for test scores in the model, we are essentially comparing students within two distinct groups – those who scored well and those who did not. The updated network diagram shows a relationship between SES and test prep that didn’t exist before. This is the induced relationship we get by controlling a collider. (Control is shown in the diagram by removing the connection of SES and test prep to the test score.)



If we look at the entire sample and compare the SES distribution (which is a continuous measure uniformly distributed between 0 and 1) for each test prep group, we see that both groups have the same distribution (i.e. there is no relationship):

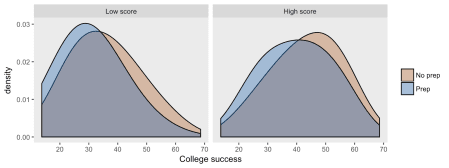


But if we look at the relationship between SES and test prep within each test score group, the distributions no longer completely overlap – within each test score group, there is a relationship between SES and test prep.



**Why does this matter?**

If the researcher has no good measure for SES or no measure at all, he cannot control for SES in the model. And now, because of the induced relationship between test prep and (unmeasured) SES, there is unmeasured confounding. This confounding leads to the biased estimate that we saw in the second model. And we see this bias in the densities shown for each test score group:



If it turns out that we *can* control for SES as well, because we have an adequate measure for it, then the artificial link between SES and test prep is severed, and so is the relationship between test prep and the long term college outcome.

rndtidy( lm(successMeasure ~ SES + highScore + testPrep, data = dt))

## term estimate std.error statistic p.value

## 1 (Intercept) 19.922 0.194 102.519 0.000

## 2 SES 40.091 0.279 143.528 0.000

## 3 highScore -0.098 0.212 -0.462 0.644

## 4 testPrep 0.137 0.174 0.788 0.431

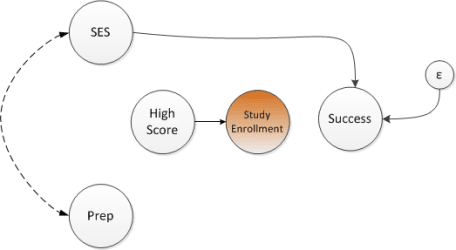
The researcher can create problems by controlling for all the variables he has and not controlling for the variables he doesn’t have. Of course, if there are no colliders and mediators, then there is no harm. And unfortunately, without theory, it may be hard to know the structure of the DAG, particularly if there are important unmeasured variables. But, the researcher needs to proceed with a bit of caution.

**Addendum: selection bias**

“Selection bias” is used in a myriad of ways to characterize the improper assessment of an exposure-outcome relationship. For example, unmeasured confounding (where there is an unmeasured factor that influences both an exposure and an outcome) is often called selection bias, in the sense that the exposure is “selected” based on that particular characteristic.

Epidemiologists talk about selection bias in a very specific way, related to how individuals are selected or self-select into a study. In particular, if selection into a study depends on the exposure of interest and some other factor that is associated with the outcome, we can have selection bias.

How is this relevant to this post? Selection bias results from controlling a collider. In this case, however, control is done on through the study design, rather than through statistical modeling. Let’s say we have the same scenario with a randomized trial of a test prep course and we are primarily interested in the impact on the near-term test score. But, later on, we decide to explore the relationship of the course with the long-term college outcome and we send out a survey to collect the college outcome data. It turns out that those who did well on the near-term test were much more likely to respond to the survey – so those who have been selected (or in this case self-selected) will have an induced relationship between the test prep course and SES, just as before. Here is the new DAG:



**Simulate new study selection variable**

The study response or selection variable is dependent on the near-term test score. The selected group is explicitly defined by the value of inStudy

# selection bias

defS <- defDataAdd(varname = "inStudy",

formula = "-2.0 + 2.2 \* highScore",

dist = "binary", link = "logit")

dt <- addColumns(defS, dt)

dSelect <- dt[inStudy == 1]

We can see that a large proportion of the the selected group has a high probability of having scored high on the test score:

dSelect[, mean(highScore)]

## [1] 0.9339207

**Selection bias is a muted version of full-on collider bias**

Within this group of selected students, there is an (incorrectly) estimated relationship between the test prep course and subsequent college success. This bias is what epidemiologists are talking about when they talk about selection bias:

rndtidy( lm(successMeasure ~ testPrep, data = dSelect))

## term estimate std.error statistic p.value

## 1 (Intercept) 41.759 0.718 58.154 0.000

## 2 testPrep -2.164 0.908 -2.383 0.017