Crash course on confounding, bias, and deconfounding remedies using R

Andy Wilson and Aimee Harrison

Jun 25th, 2024

Contents

Introduction

> In theory there is no difference between theory and practice. In practice there is.
> Yogi Berra

Confounding bias is one of the most ubiquitous challenges in estimating effects from observational (real-world data) studies. Confounding happens when the relationship between a treatment (or other exposure) and an outcome is distorted by the presence of other variables. These confounding variables can create misleading associations, or biases, making it difficult to infer causal relationships accurately. Fortunately, study design and statistical methods have been evolving to help us mitigate biases and recover the true effect estimate.

In this article, we will explore the concepts of confounding, bias, causal identifiability, and target trial emulation and also provide an overview of causal graphs. We will then introduce traditional and modern (or emerging) applied methods for estimating effects from data - including targeted learning and targeted maximum likelihood estimation (TMLE) and double machine learning (DML). At the end, you should have a general understanding of the benefits and shortcomings of traditional and emerging methods for estimating treatment effect as well as how we (can) draw causal conclusions from data.

How to read this article

In order to offer a meaningful experience for readers across multiple fields, we have made use of tabs to separate conceptual overviews, notation, and a worked example in R. If you would like to learn the concepts but do not have a need to apply these methods, proceed by reading linearly. If you would like to see how these methods work in practice, use the "Notation" and "Worked Example" tabs.

We have also provided additional resources for key terms from the field that not all readers will be familiar with. Hover over a hyperlink to see a definition, and use the hyperlinks to navigate to more in-depth background materials.[1]

Introduction to the worked example

In this demonstration, we'll:

1) create (plausible) synthetic data so that we can know the ground truth effect we're trying to estimate;
2) estimate the effect without any attempt to remedy confounding (to start with our baseline **biased** estimate); and
3) apply and compare traditional (propensity scores, g-computation) and emerging (TMLE, DML) statistical methods to recover the unbiased effect.

We will be using an example scenario where the protective effect of statins (treatment) against high cholestoral (outcome) is obscured by confounding. Though we will be working with synthetic data in our example, this is a common and reproducible type of problem in public health.

Libraries for today's R session

All the following libraries are available via CRAN and can be installed either through the `tools` menu or using `install.packages(" ")` command for each.

[1]Note on accessibility: This is the authors' first time using these formatting conventions in RMarkdown. If you are experiencing difficulty reading the article due to compatibility with assistive technologies, or for any other reasons, please reach out to the authors. We would be happy to provide you with an alternate format, and will incorporate feedback into future iterations of the work.

```r
library(simcausal)
library(ranger)
library(tidyverse)
library(ggdag)
library(MatchIt)
library(WeightIt)
library(survey)
library(tableone)
library(cobalt)
library(PSweight)
library(tmle)
library(DoubleML)
library(mlr3)
library(mlr3learners)
set.seed(12345)
nn = 5000 #Simulation(s) sample size
```

About the authors

Andy Wilson received his PhD in Public Health and MStat in biostatistics from the University of Utah School of Medicine. He is currently adjunct faculty in the Department of Family and Preventive Medicine at the University of Utah and Head of Innovative RWD Analytics at Parexel, a large CRO. He is interested in emerging causal methods and how they help advance real-world data analytics.

Aimee Harrison

The target trial and causal interpretations

Overview

Methods to estimate treatment effects in a population depend on exchangability, or approximating equivalent conditions and baseline characteristics between the treated and untreated groups. Because we can never actually know what the outcomes for a treated group would have been had all the individuals in the group remained untreated (the counterfactual), we depend on our untreated and treated groups being sufficiently similar, so that we can more more confidently assess causal effects of a treatment.

Unlike in randomized controlled trials (RCTs), where random assignment of treatment ensures that treatment groups are comparable, observational studies rely on study design and statistical techniques to account for differences between treated and untreated groups. One approach to estimating causal effects in observational studies is to emulate the conditions of a target randomized trial as closely as possible. This process, often referred to as "target trial emulation," involves designing the observational study to mimic the hypothetical RCT that would answer the causal question of interest.

But whether causal effects are estimated from randomized data or observational data, the **legitimacy** of the causal claims comes from satisfying what we refer to as the `Causal Idenifiability assumptions.` (Hernan and Robins, *What IF*) These are, briefly:

1) *Exchangeability:* The treated and untreated groups are comparable.
2) *Positivity:* Each subject has a positive probability of receiving each treatment.

3) *Consistency:* The potential outcomes are consistent with the observed treatment.
4) No Interference: The treatment of one subject does not affect the outcome of another. {Also referred to as the single unit treatment }

Identifiability assumptions

1. **Exchangeability**:
   - *Description*: The treated and untreated groups are comparable in terms of both observed and unobserved confounders.
   - *Mathematical Notation*: $Y_t \perp\!\!\!\perp T \mid W$
   - *Explanation*: The potential outcome Y_t is independent of the treatment T given the covariates W.
2. **Positivity**:
   - *Description*: Each subject has a positive probability of receiving each treatment level given their covariates.
   - *Mathematical Notation*: $P(T = t \mid W = w) > 0$ for all treatment levels t and covariates w.
   - *Explanation*: There should be no combination of covariates for which a treatment probability is zero.
3. **Consistency**:
   - *Description*: The potential outcomes under the treatment observed are the same as the outcomes that would be observed if the treatment were assigned.
   - *Mathematical Notation*: $Y = Y_t$ if $T = t$
   - *Explanation*: If a subject received treatment t, their observed outcome is Y_t.
4. **No Interference**:
   - *Description*: The treatment of one subject does not affect the outcome of another subject.
   - *Mathematical Notation*: $Y_{i,t}$ is not affected by T_j for $i \neq j$.
   - *Explanation*: The potential outcome $Y_{i,t}$ of subject i is only affected by their own treatment T_i, not by the treatment of any other subject T_j.

By satisfying these assumptions, we strengthen the validity of our causal claims, whether derived from randomized controlled trials or observational studies.

Directed acyclic graphs

Overview

Causal graphs, specifically directed acyclic graphs (DAGs), are powerful tools used in epidemiology (and other fields) to visualize relationships between variables in a study. These graphs help researchers understand the causal pathways and identify potential confounders that could bias study results.

A DAG is a graphical representation where nodes represent variables, and directed edges (unidirectional arrows) indicate causal relationships between these variables. The "acyclic" aspect means that there are no loops or cycles, ensuring a unidirectional flow of causality. Figure 1 shows a simple DAG where the only variables modeled are treatment (statins) and outcome (high cholestoral).

DAGs are particularly useful for identifying confounding because they clearly depict the pathways through which variables are connected. Confounders (or *confounding variables*) are variables that influence both the treatment and the outcome, potentially creating a spurious association. By mapping out all relevant

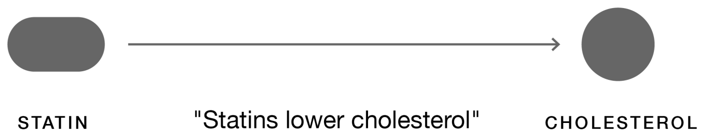


Figure 1: Figure 1: Example of a simple DAG where treatment (statins) affects (causes or influences) outcome (cholestoral).

variables and their relationships, a DAG helps researchers see which variables need to be controlled for to obtain an unbiased estimate of the treatment effect.

To better understand the relationship between treatment and outcome, lets consider a more complex model (see figure 2). Say we know patients with previously high cholesterol are much more likely to be treated with statins and that this prior cholestoral level strongly influences subsequent levels. Additionally, age and gender are suspected to be related to both treatment and outcome in our example. We say that these additional variables (confounders) influence both the likelihood of receiving treatment and the risk of having high cholestoral. We incorporate these additional variables (confounders) into our diagram, adding corresponding nodes and directed edges.

DAGs like this can guide our analysis by identifying where we need to adjust for confounders, so we can better isolate the true effect of treatment (through methods like propensity scores or TMLE).

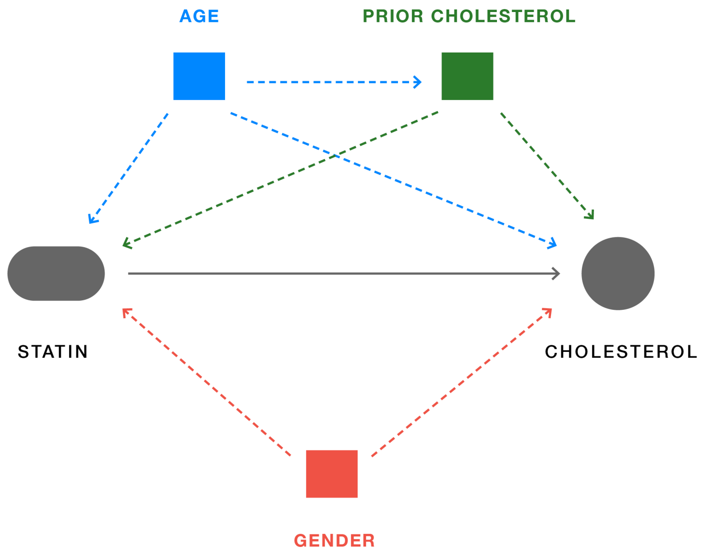


Figure 2: Figure 2: Example of a DAG where treatment (statins) affects outcome (cholestoral), and three confounding variables (age, prior cholestoral, gender) affect both treatment and outcome.

Worked example: Generating synthetic data

Synthetic Data and Ground Truth When we generate synthetic data, we have complete control over the data-generating process. This means we know the true causal effect of the treatment, allowing us to directly assess the accuracy and bias of our estimates. By applying causal inference methods to synthetic data, we can evaluate their performance in recovering the known causal effect (as well as identifying and quantifying bias).

DAG-based data generation process We're going to use a package called `simcausal` to create a hypothetical DAG and simulate data (directly) from that. (A more traditional approach is to build up the nodes sequentially, see box 1 in the linked tutorial.)

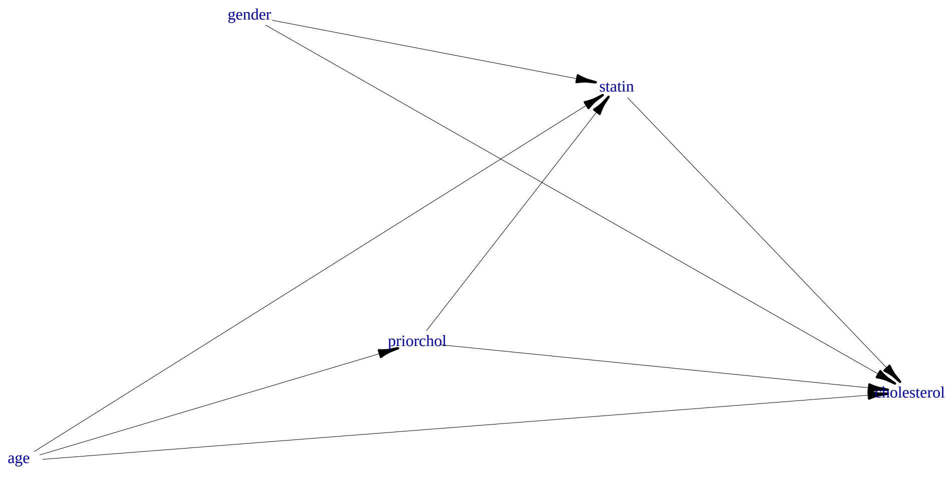
```
D_AW <- DAG.empty() +
  node("age", distr = "rnorm", mean = 50, sd = 10) +
        node("gender", distr = "rbern", prob = 0.5) +
       node("priorchol", distr = "rnorm", mean = 200 +
              0.5 * age, sd = 30) +
        node("statin", distr = "rbern",
             prob = plogis(-2 + 0.02 * priorchol +
                             0.01 * age - 0.5 * gender)) +
        node("cholesterol", distr = "rnorm",
             mean = 180 - 20 * statin + 0.8 * priorchol + 0.5 *
               age + 5 * gender, sd = 15)

D_AW_set <- set.DAG(D_AW)
```

Notice the value we put in above: `- 20 * statin` → this is where we seed in our causal effect we'll recover later.

Using `plotDAG`, let's take a look at the resulting DAG. Notice the interesting configuration of treatment, outcome, and confounders here.

```
plotDAG(D_AW_set, xjitter = 0.3, yjitter = 0.3)
```



Generate synthetic data Using this DAG, let's now generate some data, starting with n = 5000.

```
ObsData <- sim(DAG = D_AW_set, n = nn, rndseed = 12345)
head(ObsData)
```

```
##   ID      age gender priorchol statin cholesterol
## 1  1 55.85529      0  164.8389      1    315.7124
## 2  2 57.09466      1  253.0668      1    399.0876
## 3   3 48.90697       1   254.1566       1     376.5909
## 4   4 45.46503       1   207.4650       1     361.7573
## 5   5 56.05887       1   235.9930       1     368.5381
## 6   6 31.82044       0   188.3088       1     316.1860
```

```
A1 <- node("statin", distr = "rbern", prob = 1)
D_AW_set <- D_AW_set + action("A1", nodes = A1)
A0 <- node("statin", distr = "rbern", prob = 0)
D_AW_set <- D_AW_set + action("A0", nodes = A0)

Xdat1 <- sim(DAG = D_AW_set, actions = c("A1", "A0"), n = nn, rndseed = 12345)

Y1 <- Xdat1[["A1"]][,"cholesterol" ]
Y0 <- Xdat1[["A0"]][,"cholesterol" ]

(True_ATE <- mean(Y1 - Y0))
```

```
## [1] -20
```

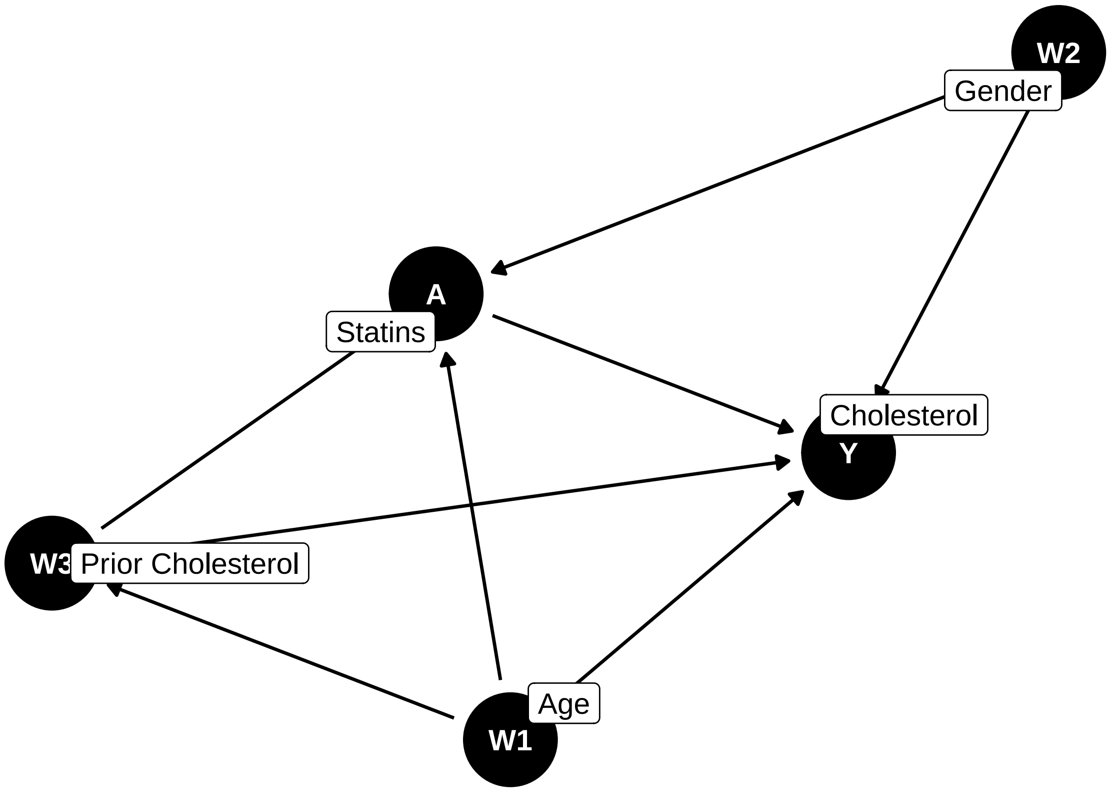
Notice we have created the *full* dataset! We have the (usually unavailable) Y(1) and Y(0) as well as the observed Y.

Bonus: Prettier DAGs with ggdag

Notice that the DAG produced in our worked example included lines crashing on the labels, and compressed type. For more stylish (and accessible) DAGs, consider using the `ggdag` package:

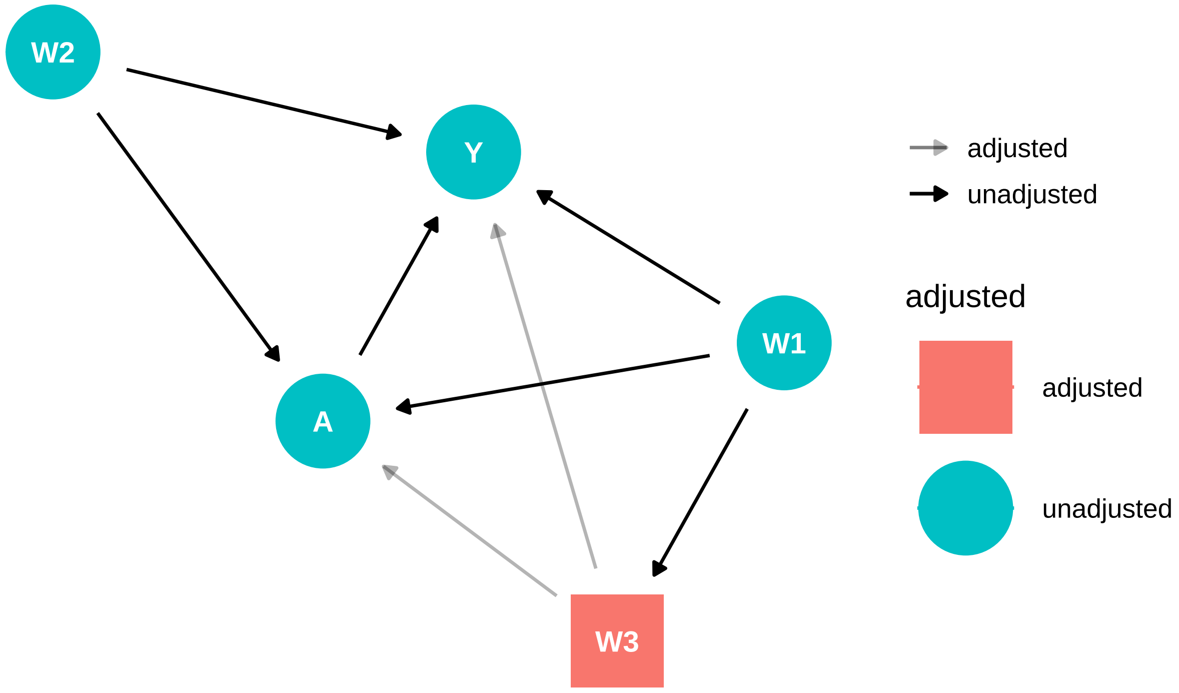
```
theme_set(theme_dag())

D_AW_DAG <- dagify(Y ~  A + W1 + W2 + W3 ,
                   A ~ W1 + W2 + W3,
                   W3 ~ W1,
                   labels = c("Y" = "Cholesterol",
                              "A" = "Statins",
                              "W1" = "Age",
                              "W2" = "Gender",
                              "W3" = "Prior Cholesterol"),
                   exposure = "A",
                   outcome = "Y")
ggdag(D_AW_DAG, text = TRUE, use_labels = "label")
```



The package offers a range of styles and controls, available in the reference PDF. We can also use `ddag` to control for partcular variables. In the example below, we adjust for “W1”.

```
# control_for(D_AW_DAG, var = "W1")
ggdag_adjust(D_AW_DAG, var = "W3")
```



Bias and average treatment effect

Overview

Bias refers to systematic errors (e.g., confounders) that distort relationship between variables, leading to incorrect conclusions about causality. Here we will define it as `bias = true - estimated`. An estimator is said to be unbiased if its bias is zero.

The *apparent* (naive) estimate of average treatment effect (ATE) is calculated by directly taking the difference between the observed risk of outcome in the treated group and the observed risk of outcome in the untreated group (without correcting for bias). The *true* ATE is calculated by taking the difference between risk of counterfactual outcomes for the treated versus the risk of counterfactual outcomes for the untreated groups. By considering counterfactuals (calculated) rather than observed (actual) outcomes, we correct for bias. (Think of "*conditional* A" vs. "*do* A.")

In an actual study setting, you would not be able to so easily calculate *true* ATE (or bias) because you would not know the counterfactual outcomes. In our worked example, whichwe will be able to calculate both naive and true ATE because we are using synthetic data that generates counterfactual outcomes. This will allow us to compare results and see how good our traditional and emerging methods are at correcting for bias.

DAGs to identify confounding variables

DAGitty (https://www.dagitty.net/) is an excellent online tool that helps researchers create and analyze DAGs. It provides a user-friendly interface to draw DAGs and automatically identifies the necessary adjustments to control for confounding variables. DAGitty can also check for d-separation, a criterion used to determine if a set of variables blocks all backdoor paths, ensuring valid causal inference.

Notation

The *apparent (naive) estimate of average treatment effect* is defined as:

$$\widehat{ATE} = \widehat{\psi} = E[Y|A=1] - E[Y|A=0]$$

where $E[Y|A]$ represents the risk of outcome Y given a treatment value A (1 = treated, 0 = untreated).

The *true ATE* is defined ad:

$$ATE = \psi = E[Y(1)] - E[Y(0)]$$

where $E[Y(i)]$ represents the risk of outcome Y given counterfactual treatment $A = i$.

From the apparent (naive) estimate of the ATE and the true ATE, *bias* can be defined as:

$$bias = ATE - \widehat{ATE} = \psi - \widehat{\psi}$$

where ψ is the target parameter (in this case ATE).

Percent bias is defined here as:

$$\%bias = \frac{\widehat{\psi} - \psi}{\psi} * 100\%$$

Worked example

Let's take a look at the apparent (naive) estimate of average treatment effect in our cholesterol example. We will compare this to the *true* ATE (a value we can only calculate because our synthetic data generated Y(1) and Y(0)).

```
treat <- ObsData %>% filter(statin == 1) %>% select(cholesterol)
controls <- ObsData %>% filter(statin == 0) %>% select(cholesterol)

(naive_est <- mean(treat$cholesterol) - mean(controls$cholesterol))
```

```
## [1] -5.117758
```

```
(True_ATE <- mean(Y1 - Y0))
```

```
## [1] -20
```

```
# Absolute bias
(bias <- naive_est - True_ATE)
```

```
## [1] 14.88224
```

```
# Percent bias
(naive_est-True_ATE)/True_ATE*100
```

```
## [1] -74.41121
```

So our bias in the above example is -74.4112%.

Traditional methods for effects estimates

Crude (naive) estimates

Overview

Crude (naive) estimates provide a basic understanding of the association between treatment and outcome (without correcting for bias) by evaluating the apparent difference in means between treatment groups, ignoring all else. You can also think of this as the conditional mean difference, without adjustments for covariates.

Due to the synthetic nature of our data, in the worked example, this will look in the same as our naive estimate of ATE calculated in the last section. However, in a real world setting this would differ because . . .

Notation

The **conditional mean difference** for our worked example is defined as:

$$CMD = E[cholesterol|statin = 1] - E[cholesterol|statin = 0]$$

Note that this is different from the apparent (naive) estimate of ATE because …

Worked example

All right. Let's get down to business removing this bias and getting the estimate we *would have gotten* from a (target) randomized trial. To begin, we will calculate the crude (naive) estimate, without adjustment, before applying four methods to remove bias:

1) Propensity score (matching and weighting)
2) G-computation
3) TMLE
4) DML

It's good to plant our flag and see how bad it is if we don't do anything about bias. (You'll see we just recover the initial biased estimate, but it should help level-set.)

```r
naive <- glm(cholesterol ~ statin, data = ObsData)
summary(naive)
```

```
##
## Call:
## glm(formula = cholesterol ~ statin, data = ObsData)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  374.013      1.573 237.839  < 2e-16 ***
## statin        -5.118      1.632  -3.137  0.00172 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 877.8762)
##
##     Null deviance: 4396263  on 4999  degrees of freedom
## Residual deviance: 4387625  on 4998  degrees of freedom
## AIC: 48081
##
## Number of Fisher Scoring iterations: 2
```

As we can see, by regression, we get get the same biased estimate (statin coefficient = -5.1178) that we calculated earlier via apparent (naive) average treatment effect. So, our initial percent bias from simple (unadjusted) regression is -74.41% (the same value we calculated using ATE and $\widehat{ATE}$.

Propensity Scores

Overview Introduced by Rosenbaum and Rubin in 1983, propensity scores evaluate the conditional probability that someone (e.g., a patient) receives a treatment given a set of observed covariates. Historically, propensity scores have been a powerful tool used in observational studies to control for confounding variables. The concept provides an elegant and intuitive way to balance the distribution of covariates between treated and untreated (control) groups.

Notation The propensity score is defined as:

$$e(X) = P(T = 1|X)$$

where T represents the treatment indicator (1 for treated, 0 for untreated), and X represents the set of observed covariates. *Note that by convention both A and T can be used to represent treatment, though A is typically used for continuous or categorical treatment, and T is often used to indicate a binary treatment.*

Propensity Scores for Matching

Overview Propensity score matching involves pairing units (individuals) between treated and control (untreated) groups with similar propensity scores. This technique aims to create a pseudo-randomized experiment by ensuring a similar distribution of covariates between the two matched groups. Given the appropriate selection of covariates, matching can allow for more accurate estimation of treatment effects by significantly reducing selection bias and *cohort imbalance* (unequal distribution of covariates between treated and control groups).

Although propensity score matching has been historically important, contemporary debates question whether propensity scores should be used for matching (see King et. al. 2019). We tend to lean away from recommending this method, although it is very intuitive and tractable.

Notation In our worked example for propensity score matching, we will use a variant of the average treatment effect: the **average treatment effect among the treated (ATT)**, defined as:

$$ATT = E[Y(1) - Y(0)|T = 1]$$

where the difference between the counterfactual outcomes $Y(1)$ and $Y(0)$ are only compared within the treatment group ($T = 1$).

To quantify mean values of covariates across treatment groups, we will also use the **standard mean difference (SMD)**, defined as:

$$SMD = \frac{\widehat{X(1)} - \widehat{X(0)}}{s_p}$$

where $\widehat{X(t)}$ is the mean value of the covariate in the group $T = t$ and s_p is the pooled standard deviation.

Worked example Recall that when drawing our initial DAG, we identified three potential confounders: age, gender, and prior cholesterol. Let's use the `tableone` package to evaluate the standard mean difference (SMD) across treatment groups for each of these covariates:

```r
xvars<-c("age","gender","priorchol")
```

```r
table1<- CreateTableOne(vars=xvars, strata="statin", data=ObsData )
## include standardized mean difference (SMD)
print(table1, smd=TRUE)
```

```
##                        Stratified by statin
##                         0               1               p      test SMD
##   n                       355            4645
##   age (mean (SD))        47.73 (9.84)   50.18 (9.87)  <0.001       0.248
##   gender (mean (SD))      0.61 (0.49)    0.49 (0.50)  <0.001       0.243
##   priorchol (mean (SD)) 209.52 (29.85) 226.80 (29.95) <0.001       0.578
```

Wow - look at the difference/imbalance! (For reference, when evaluating distribution of covariates across treatment groups, a heuristic for balance is $SMD < 0.1$.)

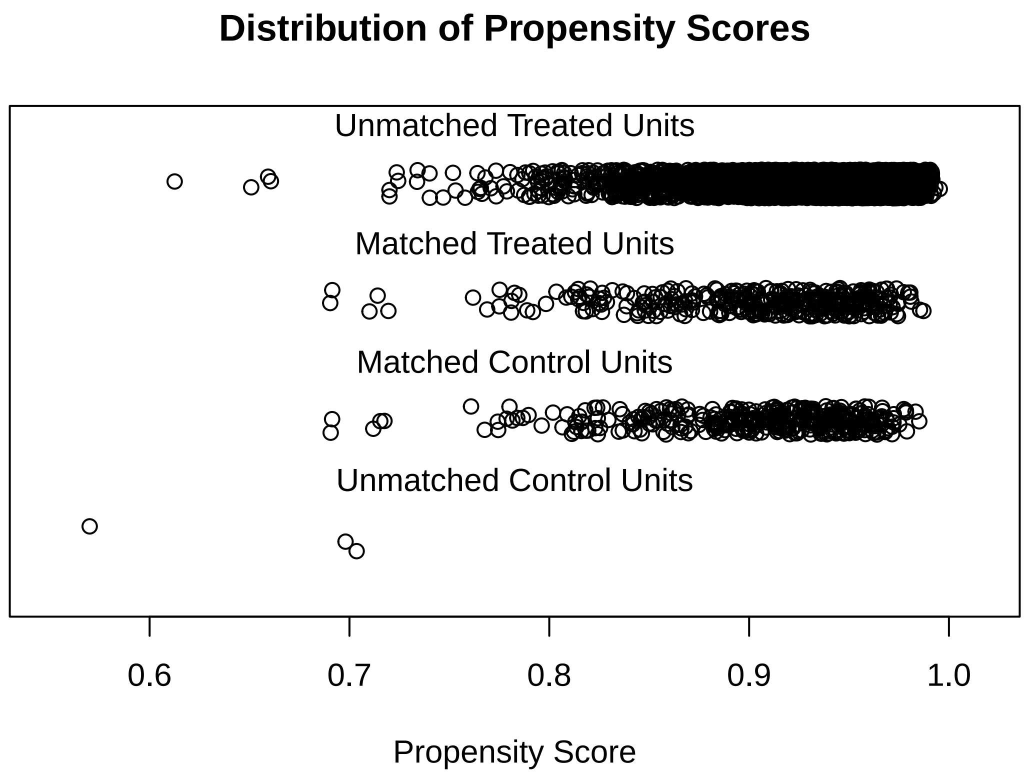
Now, let's use `matchit` to apply propensity score matching:

```r
propensity_model <- matchit(statin ~ age + gender + priorchol,
                    data = ObsData,
                    method = "nearest", # computationally efficient
                    ratio = 1,
                    caliper = 0.05)
bal.tab(propensity_model, m.threshold=0.1)
```

```
## Balance Measures
##               Type Diff.Adj    M.Threshold
## distance  Distance   0.0491 Balanced, <0.1
## age         Contin.  -0.0181 Balanced, <0.1
## gender       Binary  -0.0028 Balanced, <0.1
## priorchol  Contin.   0.0635 Balanced, <0.1
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         4
## Not Balanced, >0.1     0
##
## Variable with the greatest mean difference
##   Variable Diff.Adj    M.Threshold
##  priorchol   0.0635 Balanced, <0.1
##
## Sample sizes
##           Control Treated
## All           355    4645
## Matched       352     352
## Unmatched       3    4293
```

And we'll use `propensity model` to inspect the results:

```r
print(plot(propensity_model, type = 'jitter'))
```



```
## To identify the units, use first mouse button; to stop, use second.
## A matchit object
##  - method: 1:1 nearest neighbor matching without replacement
##  - distance: Propensity score [caliper]
##              - estimated with logistic regression
##  - caliper: <distance> (0.002)
##  - number of obs.: 5000 (original), 704 (matched)
##  - target estimand: ATT
##  - covariates: age, gender, priorchol
```

You’ll notice that our groups are now pretty well balanced, although we lost some friends along the way. (For this analysis, we dropped all the *unmatched* people.)

Let’s extract the matches, so we can re-evaluate treatment effect in matched groups (using ATT):

```
md <- match.data(propensity_model)
```

```
adjusted.PSmatch <- glm(cholesterol ~ statin,
                        data = md,
                        weights = weights)

summary(adjusted.PSmatch)
```

```
##
## Call:
## glm(formula = cholesterol ~ statin, data = md, weights = weights)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  374.496      1.580 237.063  < 2e-16 ***
## statin       -18.607      2.234  -8.329 4.27e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 878.4353)
##
##     Null deviance: 677599  on 703  degrees of freedom
## Residual deviance: 616662  on 702  degrees of freedom
## AIC: 6773.7
##
## Number of Fisher Scoring iterations: 2
```

You'll notice that the effects estimate we calculated here is much to the *true* treatment effect we initially calculated (-18.6074 vs -20).

Propensity Scores for Weighting

Overview Propensity score weighting involves assigning weights to each individual based on their propensity score to create a synthetic (standardized) sample in which the distribution of covariates is balanced across treatment groups. The effect is that the apparent sample size of each group and the impact of each individual's outcome on the average treatment effect will be weighted depending on propensity score.

There are two methods commonly used for calculating propensity score weights: **inverse probability of treatment weighting (IPTW)** and **stabilized weights.** For both methods, each unit of treated group is weighted by the inverse of their propensity score; each unit of the control population is weighted by the inverse of one minus their propensity score. If there are extreme propensity scores, and therefore extreme weights, inverse probability weighting leads can lead to high variance. In such cases, stabilized weights can be used instead. Stabilized weights are calculated with an additional factor: the marginal probability of treatment.

Notation Non-stabilized weight for traditional **inverse probability of treatment weighting (IPTW)** is expressed as:

$$w_i = \frac{T_i}{e(X_i)} + \frac{1-T_i}{1-e(X_i)}$$

where w_i represents weight, T_i represents the treated individuals, $1-T_i$ represents the untreated individuals, and $e(X_i)$ represents propensity score.

Stabilized weight is expressed as:

$$w_i = \frac{T_i}{e(X_i)} \times \frac{P(T=1)}{P(T=1|X)} + \frac{1-T_i}{1-e(X_i)} \times \frac{P(T=0)}{P(T=0|X)}$$

where $P(T=t)$ represents the marginal probability of treatment.

Worked example Returning to our original data set, let's use the `WeightIt` package to recalculate weights. First we will calculate non-stabilized and apply them to the data:

```r
# Calculating non-stabilized weights
w.out.ns <- weightit(cholesterol ~ age + gender + priorchol, data = ObsData,
                     method = "ps", estimand = "ATE", stabilize = FALSE)
summary(w.out.ns)
```

```
##                      Summary of weights
##
## - Weight ranges:
##
##         Min                                    Max
## all 0.0022 |---------------------------| 120.0368
##
## - Units with the 5 most extreme weights:
##
##           869    3706     117     2270     3852
##  all 79.1375 89.2385 94.5719 102.3759 120.0368
##
## - Weight statistics:
##
##     Coef of Var   MAD Entropy # Zeros
## all       2.871 0.794   0.862       0
##
## - Effective Sample Sizes:
##
##            Total
## Unweighted  5000
## Weighted     541
```

```r
# Adding weights to data
ObsData$weights_ns <- w.out.ns$weights
```

And then again for stabilized weights:

```r
# Calculating stabilized weights
w.out.st <- weightit(cholesterol ~ age + gender + priorchol, data = ObsData,
                     method = "ps", estimand = "ATE", stabilize = TRUE)
summary(w.out.st)
```

```
##                      Summary of weights
##
## - Weight ranges:
##
##         Min                                    Max
## all 0.0022 |---------------------------| 120.0368
##
## - Units with the 5 most extreme weights:
##
##           869    3706     117     2270     3852
##  all 79.1375 89.2385 94.5719 102.3759 120.0368
##
## - Weight statistics:
##
##     Coef of Var   MAD Entropy # Zeros
## all       2.871 0.794   0.862       0
##
```

```
## - Effective Sample Sizes:
##
##            Total
## Unweighted  5000
## Weighted     541
```

```r
# Adding weights to data
ObsData$weights_st <- w.out.st$weights
```

Notice that at a glance these results appear the same. This is because in our scenario we didn't really introduce pathological weighting - results may be different and pull apart if we introduced extreme propensity scores (e.g., near 0) which can lead to enormous inverse treatment weights. This is also a consideration when addressing positivity assumption.

Now we will use `svydesign` to generate data for treated and control groups based on our weighted data, and from those groups, we will calculate ATE. First for non-stabilized weights:

```r
# Creating survey design objects
design_ns <- svydesign(ids = ~1, weights = ~weights_ns, data = ObsData)

# Estimating ATE using non-stabilized weights
ate_ns <- svyglm(cholesterol ~ statin, design = design_ns)
summary(ate_ns)
```

```
##
## Call:
## svyglm(formula = cholesterol ~ statin, design = design_ns)
##
## Survey design:
## svydesign(ids = ~1, weights = ~weights_ns, data = ObsData)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  388.720      3.132 124.093  < 2e-16 ***
## statin       -20.979      3.344  -6.273 3.83e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 624.499)
##
## Number of Fisher Scoring iterations: 2
```

And again for stabilized weights:

```r
# Creating survey design objects
design_st <- svydesign(ids = ~1, weights = ~weights_st, data = ObsData)

# Estimating ATE using stabilized weights
ate_st <- svyglm(cholesterol ~ statin, design = design_st)
summary(ate_st)
```

```
##
## Call:
## svyglm(formula = cholesterol ~ statin, design = design_st)
##
## Survey design:
## svydesign(ids = ~1, weights = ~weights_st, data = ObsData)
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  388.720      3.132 124.093  < 2e-16 ***
## statin       -20.979      3.344  -6.273 3.83e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 624.499)
##
## Number of Fisher Scoring iterations: 2
```

Remember that the *true* treatment effect we calculated using $Y(0)$ and $Y(1)$ was -20%. Both stabilized and non-stabilized weights get us very close to this ATE (-20.9792% and -20.9792% respectively) – even closer than propensity score matching got us.

G-computation

Overview

Introduced by James Robins in 1986 to address limitations of traditional causal effects estimates, particularly in longitudinal studies, g-computation (also known as the g-formula or g-estimation) is a very promising method used to estimate true treatment effect in complex settings where variables may vary over time.

G-computation involves using a statistical model to predict outcomes under different treatment scenarios. The key idea is to estimate what each individual's outcome would be under each possible treatment condition (even though we only actually observe them under one condition). These estimated outcomes are called counterfactual outcomes.

The essence of g-computation lies in its ability to estimate the "unseen" (counterfactual) version of each individual (ie, what an individual's outcome would have been had they received a different treatment than the one they actually received), accomplished by using a statistical model trained on the observed data. This approach accounts for confounding factors and provides a very good estimate of the average treatment effect.

There are three steps to g-computation:

1. **Modeling the outcome:** First, a model is fitted to predict the outcome based on the treatment and a set of covariates. This model can be a regression model or it can be any machine learning model that can generate predicted outcomes.
2. **Predicting counterfactuals:** Using the fitted model, predict the outcomes for each individual under different treatment scenarios. Specifically, predict the outcome for each individual if they had received the treatment and if they had not received the treatment.
3. **Averaging the predictions:** Calculate the average of the predicted outcomes for the treated and control scenarios across all individuals in the sample to determine an (unbiased) ATE.

Notation

Actual outcome and counterfactual outcome for treated individuals ($T = 1$) are represented by:

$$Y, Y(0)$$

where Y represents the actual outcome and Y(0) represents the counterfactual treatment had they not received treatment ($T = 0$).

Actual outcome and counterfactual outcome for individuals who did not receive treatment ($T = 1$) are represented by:

$$Y, Y(1)$$

where $Y(1)$ represents the counterfactual treatment had they received treatment ($T = 1$).

Mathematically, the three steps of g-computation can be written as:

1. **Modeling the outcome** (where f represents the fitted model, $\widehat{Y}$ represents estimated outcome, T represents actual treatment, and X represents a set of covariates)

$$\widehat{Y} = f(T, X)$$

2. **Predicting counterfactuals**

$$Q_1 = \widehat{Y}(T = 1, X = x) \text{ and } Q_0 = \widehat{Y}(T = 0, X = x)$$

3. **Averaging the predictions**

$$\text{ATE} = \frac{1}{N}\sum_{i=1}^{N}\left[\widehat{Y_i}(T = 1) - \widehat{Y_i}(T = 0)\right]$$

Worked example

Returning to our original data set, we want to generate $\widehat{Y_i}(T = 0)$ and $\widehat{Y_i}(T = 1)$ (which we'll call Q_0 and Q_1). Let's start by using the `SuperLearner` package to calculate expected outcomes.

```r
# Define Super Learner libraries
sl_libs <- c("SL.glmnet", "SL.ranger", "SL.glm")
# Select relevant variables from the dataset
Obs <- ObsData %>% dplyr::select(statin, cholesterol,age, gender, priorchol)
# Extract outcome and covariates
Y <- Obs$cholesterol
W_A <- Obs %>% dplyr::select(-cholesterol)
```

Now we will begin g-computation by *modeling the outcomes (step 1)*:

```r
# Outcome regression using Super Learner
Q <- SuperLearner(Y = Y,
                  X = W_A,
                  SL.library = sl_libs)


Q_A <- as.vector(predict(Q)$pred)
```

Next we'll set `treatment` to 0 and then 1 for each individual and *predict counterfactuals (step 2)*:

```r
# Predict Q for treated (A = 1) and untreated (A = 0)
W_A1 <- W_A %>% mutate(statin = 1)  # Set A = 1
Q_1 <- as.vector(predict(Q, newdata = W_A1)$pred)
```

```r
W_A0 <- W_A %>% mutate(statin = 0) # Set A = 0
Q_0 <- as.vector(predict(Q, newdata = W_A0)$pred)
```

Finally we will *average the predictions (step 3)*:

```r
# Create dataset for g-computation
dat_g.comp <- as.data.frame(cbind(Y = Obs$cholesterol, A = Obs$statin, Q_A, Q_0, Q_1))
```

The g-computation estimate of the ATE is simply the average difference of Q_1 and Q_0:

```r
(ate_gcomp <- mean(dat_g.comp$Q_1 - dat_g.comp$Q_0))
```

```
## [1] -19.80528
```

Nice! This is *very* close to the *true* ATE. let's pause and take a look at how it compares to the four other treatment estimates we have calculated thus far:

Method	ATE (or ATT)
true (baseline ref)	-20
crude (naive) estimate	-5.1178
propensity score matching	-18.6074
propensity score weighting (stabilized weight)	-20.9792
propensity score weighting (non-stabilized weight)	-20.9792
g-computation	-19.8053

Summary

Emerging methods for effects estimates

Targeted Learning emerged as a framework in the early-mid 2000s, led by the work of Mark van der Laan and Jamie Robins. The crux of Targeted Learning is to start with a target parameter of interest and set up the problem such that we can use data-adaptive substitution estimators to produce the most unbiased estimate of the parameter possible for a given study design. It does this by leveraging the flexibility of machine learning algorithms to model complex relationships within the data while simultaneously incorporating robust statistical principles to ensure valid inference. Specifically, Targeted Learning:

1) Defines the Target Parameter: Clearly specifies the causal parameter of interest, such as the average treatment effect.
2) Utilizes Machine Learning Models: Applies machine learning techniques to estimate nuisance parameters, like propensity scores and outcome regressions, which are crucial for adjusting for confounders.
3) Applies Targeted Maximum Likelihood Estimation (TMLE): Combines the initial machine learning estimates with a targeted update step to refine the estimation and reduce bias.
4) Ensures Double Robustness: Guarantees that the estimator remains consistent if either the model for the treatment assignment or the model for the outcome is correctly specified.
5) Incorporates Cross-Validation: Uses cross-validation to avoid overfitting and ensure that the estimates are generalizable.

Targeted Learning aims to achieve a balance between flexibility and statistical rigor, providing the most accurate and unbiased estimates possible from the available data. Additionally, because the TMLE estimator is asymptotically linear, the distribution of the estimator can be approximated by a normal distribution in large samples. This property enables straightforward construction of confidence intervals and hypothesis tests.

To date, (in our opinion) targeted learning and TMLE offers the most promising method to debias target estimates and get to the true (treatment) effect from a given study design.

See van der Laan and Rose (2011) for an excellent full course on Targeted Learning.

Targeted maximum likelihood estimation

Overview

Like g-computation, TMLE uses a *substitution estimator approach* to approximate counterfactual outcomes in order to evaluate ATE. However, TMLE takes the estimations further by using flexible machine learning algorithms to derive *a fluctuation parameter* from the propensity scores and calculate a *clever covariate* from the initial estimates. These tools are then used to update the initial estimates, aligning them more closely to the target parameter (eg, ATE) - "a second chance to get it right" using the components we've introduced!

There are four steps to TMLE:

1. **Select a target parameter:** Define the specific causal effect of interest (e.g. ATE).
2. **Calculate the initial estimates:** Using machine learning, create two initial estimates:
   - an outcome model (similar to step one of g-computation)
   - a propensity score (using IPTW)
3. **Construct the clever covariate:** Using the propensity score, derive a *clever covariate* to direct the estimate towards the target parameter.
4. **Choose a fluctuation parameter:** Select a fluctuation parameter that can be used to minimize loss between the observed and updated models.
5. **Compute the targeted estimate:** Using a process called targeting, which makes use of the fluctuation parameter, update the initial outcome model to reduce bias and improve efficiency. Combine the updated outcome model and the clever covariate to obtain the targeted estimate of the treatment effect that, by design, is be both unbiased and efficient.

Though the resulting ATE may appear similar to our traditional methods, there are three key advantages to using TMLE that make it an invaluable tool:

1. **Double robustness:** TMLE provides valid estimates even if either the outcome model or the propensity score model is misspecified (but not both).
2. **Efficiency:** By incorporating both the clever covariate and the fluctuation parameter, TMLE achieves higher efficiency compared to traditional methods.
3. **Robustness to model misspecification:** TMLE's targeted update step helps mitigate bias due to model misspecification, leading to more reliable causal inference.

Notation

The **estimated outcome regression** is represented as:

$$Q(T, X) = E[Y \mid T, X]$$

where E represents the original outcome regression model, T is treatment value, and X is a set of covariates. The **estimated propensity score** is represented by:

$$e(X) = P[T \mid X]$$

where P is the original propensity score and $e(X)$ is estimated propensity score.

The **clever covariate** is defined as:

$$H(T, X) = \frac{T}{e(X)} - \frac{1-T}{1-e(X)}$$

The *updated outcome estimate* is defined as:

$$Q^{\star}(T, X) = Q(T, X) + \epsilon H(A, X)$$

where ϵ is the fluctuating parameter.
Recall that we represented **g-computation** as:

$$\Psi_n = \frac{1}{n}\sum_{i=1}^{n} Q_n(1, w_i) - \frac{1}{n}\sum_{i=1}^{n} Q_n(0, w_i)$$

where Ψ is ATE, w_i is weight, and 1 or 0 represent the treated and untreated groups.
Similarly, **TMLE** can be represented by:

$$\Psi_n^{tmle} = \frac{1}{n}\sum_{i=1}^{n} Q_n^{\star}(1, w_i) - \frac{1}{n}\sum_{i=1}^{n} Q_n^{\star}(0, w_i)$$

Notice that this takes the same form as g-computation, but the $\star$ symbols indicate the updated counterfactual outcomes $Q(1)$ and $Q(0)$.

Worked example

Let's turn back again to our initial data to model TMLE from start to finish. We'll start by identifying our target parameter: ATE. (Step 1 done!)

For step 2, we need to calculate an outcome model and a propensity score. For both of these, we can follow the steps from g-computation, using the ensemble machine learning approach `SuperLearning` to calculate propensity score.

```r
# Propensity score estimation using Super Learner
A <- Obs$statin
W <- Obs %>% dplyr::select(-cholesterol, -statin)
g <- SuperLearner(Y = A,
                  X = W,
                  family=binomial(),
                  SL.library=sl_libs)
g_w <- as.vector(predict(g)$pred)
```

Now let's calculate the *clever covariate* (step 3) and define a *fluctuation parameter* (step 4):

```r
H_1 <- 1/g_w

H_0 <- -1/(1-g_w)

# add clever covariate data to previous dataset we made
dat_tmle <-
  dat_g.comp %>%
  bind_cols(
    H_1 = H_1,
    H_0 = H_0) %>%
  mutate(H_A = case_when(A == 1 ~ H_1,   # if A is 1 (treated), assign H_1
                         A == 0 ~ H_0))  # if A is 0 (not treated), assign H_0
```

```r
glm_fit <- glm(Y ~ -1 + offset(Q_A) + H_A,
               data=dat_tmle)

(eps <- coef(glm_fit))
```

```
##          H_A
## 0.0004389115
```

With *clever covariate* and *fluctuation parameter* in hand, we can update the initial estimates of the expected outcome (step 5):

```r
dat_tmle<- dat_tmle %>%
 mutate(Q_A_update = Q_A + eps*H_A,
        Q_1_update = Q_1 + eps*H_1,
        Q_0_update = Q_0 + eps*H_0)
```

and finally, compute the target estimate (ATE):

```r
(tmle_ate <- mean(dat_tmle$Q_1_update - dat_tmle$Q_0_update))
```

```
## [1] -19.79597
```

Chef's Kiss!

Now that you've seen how the machine works, we'll let you in on a secret: there is (of course) a one-stop-shop package to calculate `tmle`:

```r
tmle_fit <- tmle(Y = Y, A = A, W = W,
                 Q.SL.library = sl_libs,
                 g.SL.library = sl_libs)

tmle_fit
```

```
##  Marginal mean under treatment (EY1)
##    Parameter Estimate:  367.85
##    Estimated Variance:  0.18218
##               p-value:  <2e-16
##     95% Conf Interval:  (367.01, 368.68)
##
##  Marginal mean under comparator (EY0)
##    Parameter Estimate:  387.65
##    Estimated Variance:  0.97811
##               p-value:  <2e-16
##     95% Conf Interval:  (385.71, 389.59)
##
##  Additive Effect
##    Parameter Estimate:  -19.806
##    Estimated Variance:  0.89669
##               p-value:  <2e-16
##     95% Conf Interval:  (-21.662, -17.95)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -19.782
##    Estimated Variance:  0.93682
##               p-value:  <2e-16
##     95% Conf Interval:  (-21.679, -17.885)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -19.845
##    Estimated Variance:  0.65932
##               p-value:  <2e-16
##     95% Conf Interval:  (-21.437, -18.254)
```

```
tmle_fit$estimates$ATE$psi
```

```
## [1] -19.80578
```

Look how close that is to our *true* ATE! Now let's return to our estimation table and see how TMLE compares:

Method	ATE (or ATT)
true (baseline ref)	-20
crude (naive) estimate	-5.1178
propensity score matching	-18.6074
propensity score weighting (stabilized weight)	-20.9792
propensity score weighting (non-stabilized weight)	-20.9792
g-computation	-19.8053
TMLE	-19.8058.

One thing to note is how tmle and g-comp estimates are nearly identical. This is a good chance to review that what tmle does is updates the g-comp estimates of $Q(1)$ and $Q(0)$ with this small amount ϵ - and you'll note in the above example $\epsilon \approx 0$. Again, this may be an artifact of how we generated the synthetic data and we invite readers to innovative data creation methods that tackle this limitation such as the CREDENCE framework of Eric Tchetgen Thetgen, et. al.

Double machine learning (a method on the horizon)

And on our radar as an exciting and recent addition to the causal machine learning toolkit that debiases estimates and provides valid (causal) statistical inference.

Overview

Double Machine Learning (DML) is another emerging (and promising) technique that combines traditional treatment effect estimators with machine learning to control for confounding. The core idea of DML is to leverage high-quality machine learning algorithms in order to more flexibly model and control for confounders while isolating the effect of the treatment variable on the outcome. This model is particularly good at inferring treatment effects in large data sets with high numbers of covariates.

DML uses orthogonalization, in particular Neyman orthogonality, to generate estimators which are insensitive to small errors in the nuisance parameters (auxiliary parameters needs to adjust the main parameter estimation process to reduce bias, for example outcome model and propensity score). It also makes use of *sample splitting* (or dividing the data into multiple parts that are used to estimate different parameters, in order to reduce the risk of overfitting) and *cross-fitting* (or doing multiple iterations of sample splitting and averaging the results).

The three primary steps of the DML algorithm are:

1) **Prediction:** Predict the outcome and treatment variable given a set of covariates, using high-performing machine learning tools (such as random forest.

2) **Residualization (orthogonalization):** Compute the residuals (difference between the predicted values and the actual values). See the *Bonus* tab in this section for an overview of a similar method, the *Frisch-Waugh-Lovell procedure*.
3) **Regression:** Estimate the target parameter (e.g. ATE) by regressing the residual of the outcome variables onto the residuals of the treatment variables. Regress W on V to estimate the target parameter ψ.

Notation

In DML, the outcome and treatment variables are estimated using machine learning methods. These estimations are represented by:

$$E[Y|Z] \text{ and } E[D|Z]$$

where Y is outcome, A is treatment, and X is the set of covariates.

Residuals are defined as:

$$W = Y - E[Y|X] \text{ and } V = A - E[A|X]$$

where W is the residuals of the outcome variables and V is the residuals of the treatment variable.

The regression of W onto V can be represented as:

$$W = \alpha + \psi + V + \epsilon$$

Worked example

Returning to our original data set, we will apply the `DoubleML` package:

```r
obj_dml_data = DoubleMLData$new(Obs, y_col = "cholesterol", d_cols = "statin")


lgr::get_logger("mlr3")$set_threshold("warn")
# Initialize a random forests learner with specified parameters
ml_l = lrn("regr.ranger", num.trees = 100, mtry = 2, min.node.size = 2,
           max.depth = 5)
ml_m = lrn("regr.ranger", num.trees = 100, mtry = 2, min.node.size = 2,
           max.depth = 5)
ml_g = lrn("regr.ranger", num.trees = 100, mtry = 2, min.node.size = 2,
           max.depth = 5)


doubleml_plr = DoubleMLPLR$new(obj_dml_data,
                               ml_l, ml_m, ml_g,
                               n_folds = 2,
                               score = "IV-type")


doubleml_plr$fit()
doubleml_plr$summary()
```

```
## Estimates and significance testing of the effect of target variables
##        Estimate. Std. Error t value Pr(>|t|)
## statin  -19.7834     0.8694  -22.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Again that's pretty close to the *true* ATE. Next, we'll look at how the three methods compare and consider how these results might vary more in across different data sets.

Bonus technique: Frisch-Waugh-Lovell

Established in the early twentieth century, the Frisch-Waugh-Lovell (FWL) theorem is a result in regression analysis used to estimate coefficients in multiple linear regression models. It provides a method for obtaining the coefficient of a single regressor in the presence of other regressors, and is similar to the method used in DML to calculate residualization. Below we will walk through an example using FWL to calculate the residuals of a treatment and outcome.

First, we will regress A(T) on W(X)

```
a.hat <- glm(statin ~  age + gender + priorchol, data = ObsData)
```

Then, we will regress Y on W (X):

```
y.hat <- glm(cholesterol ~  age + gender + priorchol, data = ObsData)
```

Now, let's compute residuals from the first two regressions:

```
delta.a <- a.hat$residuals
delta.y <- y.hat$residuals
```

Last, from these results, we can regress residuals delta.y on residuals delta.a:

```
Delta <- glm(delta.y ~ delta.a)

Delta$coefficients["delta.a"]
```

```
##   delta.a
## -19.80528
```

Voila!

In closing...

We've been through a lot today. We introduced several concepts and methods and worked though the estimation of a (fixed) causal parameter from a synthetic dataset. We seeded in the effect **-20** and saw how all methods (aside from the crude estimate) do a good job of recovering the effect by accounting for confounding variables. Each method has unique strengths and weaknesses, but they all share the same goal of providing an unbiased estimate of the causal effect and take the same inputs for this example.

Benefits and shortcomings of each method

0) Crude Estimate:

- Strengths: Simple and easy to compute.
- Weaknesses: Prone to significant bias due to confounding variables, leading to misleading results.

1) Propensity Score Matching:

- Strengths: Most intuitive and straightforward of adjustment methods. It matches treated and untreated units based on their propensity scores to balance the distribution of confounders.
- Weaknesses: Can be inefficient if many units are unmatched. Matching quality can be poor if propensity scores are poorly estimated.

2) Propensity Score Weighting:

- Strengths: Uses weights to standardize to a population without confounding (removing counfounding/bias).
- Weaknesses: Sensitive to extreme weights, which can increase variance and lead to instability in estimates.

3) G-Computation:

- Strengths: Provides a clear framework for causal inference by modeling the outcome directly.
- Weaknesses: Relies heavily (entirely) on the correct specification of the outcome model. Misspecification can lead to biased estimates.

4) Targeted Maximum Likelihood Estimation (TMLE):

- Strengths: Robust to model misspecification, provides double robustness, and leverages machine learning for nuisance parameter estimation. (Our favorite approach)
- Weaknesses: More complex and less intuitive than propensity score methods. Relatively new and still gaining adoption/acceptance

5) Double Machine Learning (DML):

- Strengths: Combines machine learning with causal methods for flexible, robust estimation. Can handle high-dimensional data.
- Weaknesses: More complex and less intuitive than propensity score methods. Relatively new and still gaining adoption/acceptance

Unified goal and inputs

Despite their differences, all these methods aim to estimate the causal effect as accurately and *unbiasedly* as possible. They start with the same input data — observations of treatments, outcomes, and DAG-identified confounding variables — and apply their different techniques to adjust for/remove confounding and recover the causal parameter.

The Role of Machine Learning

Machine learning plays a ever-increasing role in modern causal inference. It also enhances traditional methods by providing flexible, data-adaptive models that can better capture complex relationships in the data. In **all** these applications, machine learning helps improve the accuracy and robustness of causal estimates.

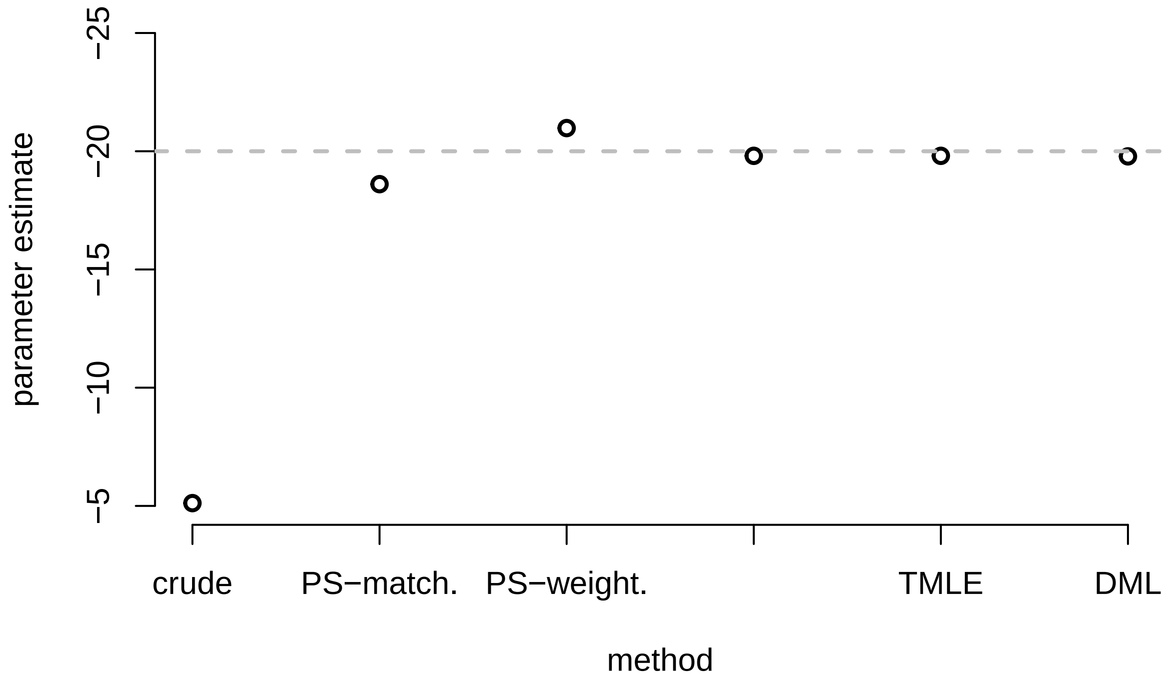
Worked example results summary

Let's take a look at all the ATE values we have thus far calculated:

Method	ATE (or ATT)
true (baseline ref)	-20
crude (naive) estimate	-5.1178
propensity score matching	-18.6074

Method	ATE (or ATT)
propensity score weighting (stabilized weight)	-20.9792
propensity score weighting (non-stabilized weight)	-20.9792
g-computation	-19.8053
TMLE	-19.8058
DML	-19.7834

```r
plot(c(1,2,3,4,5,6), c(naive$coefficients["statin"],
                 adjusted.PSmatch$coefficients["statin"],
                 ate_ns$coefficients["statin"],
                 ate_gcomp,
                 tmle_fit$estimates$ATE$psi,
                 doubleml_plr$coef),
     ylim = c(-5,-25),
     ylab = "parameter estimate",
     xlab = "method",
     axes = F, lwd = 2)
axis(1, at = c(1,2,3,4, 5,6),
     labels = c("crude", "PS-match.", "PS-weight.","
                G-comp", "TMLE", "DML"))
axis(2)
abline(h = True_ATE, lty = 2, lwd = 2, col = "grey")
text(2, .26, "grey line = 'true value'")
```



References

1. Rosenbaum, P. R., & Rubin, D. B. (1983). "The central role of the propensity score in observational studies for causal effects." *Biometrika*, 70(1), 41-55.

2. Austin, P. C. (2011). "An introduction to propensity score methods for reducing the effects of confounding in observational studies." *Multivariate Behavioral Research*, 46(3), 399-424.
3. van der Laan, M. J., & Rose, S. (2011). "Targeted Learning: Causal Inference for Observational and Experimental Data." Springer.
4. Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., & Robins, J. (2018). "Double/debiased machine learning for treatment and structural parameters." The Econometrics Journal, 21(1), C1-C68., https://doi.org/10.1111/ectj.12097