L9 Boosted propensity score estimation

Greg Ridgeway

2025-04-08

Table of contents

1 Introduction to estimating treatment effects from observational studies 2
1.1 Example: Florida murderers 2
1.2 Example: Simulated data ($\mathbf{x}$ correlated with t) 7
1.3 Example: Simulated data ($\mathbf{x}$ uncorrelated with t) 10

2 Weights to uncorrelate t and $\mathbf{x}$ 12

3 Average Treatment Effect (ATE) and the Average Treatment Effect on the Treated (ATT) 19

4 Neyman-Rubin-Holland causal model 20

5 Estimation of ATT with propensity scores 22

6 Example: Florida murderers revisited 24

7 Boosted propensity score estimation with NSDUH 27
7.1 Propensity score estimation with logistic regression 31
7.2 Outcome analysis 35
7.3 Propensity score estimation with boosting 37
7.4 `fastDR` to estimate propensity score estimates (and doubly robust estimates) . 41

8 Summary 47

1 Introduction to estimating treatment effects from observational studies

A common aim in social science is to estimate the causal effect of a treatment or exposure or condition on an outcome of interest.

1. Effect of a medical treatment on a disease outcome (e.g. survival, time to recovery)
2. Effect of race on a criminal justice outcome (e.g. arrest, sentence)
3. Effect of an educational program on reading or math scores

Ideally, we would conduct a randomized controlled trial (RCT) in which we randomize a sample of individuals to either receive the treatment condition or to a control condition, which could be nothing, business as usual, or some placebo treatment. This is the gold standard approach in science. However, we have numerous questions that are difficult to assess with an RCT for reasons such as ethical considerations, cost, and rarity of outcomes. Particularly in the last several decades, we have access to large administrative datasets with numerous features on individuals and numerous outcomes. Even if we could technically conduct an RCT, it is worth exploring these datasets to assess which questions might have interesting findings.

"Observational studies" aim to draw causal conclusions from data in which the treatment or exposure was *not* under the control of the researcher. The most common form of "observational data" are data that were not collected for the purposes of analysis, but for some basic record keeping purpose, such as health insurance claims, arrest records, mortgage applications, or school attendance and grades.

The section will focus on one particular method of causal estimation in which machine learning plays a starring role, propensity score estimation.

1.1 Example: Florida murderers

Radelet (1981) collected data on convicted in murderers in Florida between 1976-1977, including their race, their victim's race, and whether or not they received the death penalty. Our primary interest here is to assess whether there is a racial disparity in the imposition of the death penalty.

Here are the Florida murderers data.

```
library(dplyr)
library(tidyr)

dFLmurder <- rbind(matrix(rep(c("W","W",1),each= 19),ncol=3),
                   matrix(rep(c("W","W",0),each=132),ncol=3),
                  #matrix(rep(c("W","B",1),each=  0),ncol=3),
```

```
              matrix(rep(c("W","B",0),each=  9),ncol=3),
              matrix(rep(c("B","W",1),each= 11),ncol=3),
              matrix(rep(c("B","W",0),each= 52),ncol=3),
              matrix(rep(c("B","B",1),each=  6),ncol=3),
              matrix(rep(c("B","B",0),each= 97),ncol=3))
dFLmurder <- data.frame(dFLmurder) |>
  setNames(c("murderer","victim","death")) |>
  mutate(death = as.numeric(death))

dFLmurder |> group_by(murderer,victim,death) |> count()
```

```
# A tibble: 7 x 4
# Groups:   murderer, victim, death [7]
  murderer victim death     n
  <chr>    <chr>  <dbl> <int>
1 B        B          0    97
2 B        B          1     6
3 B        W          0    52
4 B        W          1    11
5 W        B          0     9
6 W        W          0   132
7 W        W          1    19
```

The simplest comparison would be to contrast the fraction of black murderers receiving the death penalty with the fraction of white murderers receiving the death penalty.

```
dFLmurder |> group_by(murderer) |> summarize(deathPenalty=mean(death))
```

```
# A tibble: 2 x 2
  murderer deathPenalty
  <chr>           <dbl>
1 B               0.102
2 W               0.119
```

So, 10.2% of black murderers received the death compared with 11.9% of white murderers. It appears that white murderers are actually 1.6 percentage points *more* likely to receive the death penalty than black murderers.

By far the most common way social scientists have traditionally explored such questions is to use a regression model. With regression we can formally test whether the outcome differs by "treatment," which in this case is the race of the murderer.

```r
# compare death sentence for B vs. W murderers
lm1 <- lm(death ~ murderer,data=dFLmurder)
summary(lm1)
```

```
Call:
lm(formula = death ~ murderer, data = dFLmurder)

Residuals:
    Min      1Q  Median      3Q     Max
-0.1187 -0.1187 -0.1024 -0.1024  0.8976

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.10241    0.02439   4.198 3.48e-05 ***
murdererW    0.01634    0.03482   0.469    0.639
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3143 on 324 degrees of freedom
Multiple R-squared:  0.0006793, Adjusted R-squared:  -0.002405
F-statistic: 0.2202 on 1 and 324 DF,  p-value: 0.6392
```

At first glance, the race of the murderer does not seem to matter. However, there is another variable in the dataset, the race of the victim. Let's explore

```r
# do B & W murderers differ by the victim's race?
dFLmurder |>
  group_by(murderer) |>
  summarize(raceVictim = mean(victim=="B"))
```

```
# A tibble: 2 x 2
  murderer raceVictim
  <chr>         <dbl>
1 B             0.620
2 W             0.0562
```

It does seem that there are large differences in the race of the victims. The majority (62%) of black murderers had black victims. White murderers rarely had black victims (5.6%). Maybe this should not matter since we are really only interested in the race of the murderer, but let's try "controlling" for the race of the victim. I put "controlling" in quotes because social

scientists often say "I controlled for the race of the victim," but what they really mean is that they put `+victim race` in a regression model, which is not the same and, as we will see, does not offer complete adjustment for the race of the victim.

```
# include race of victim in regression
lm3 <- lm(death ~ murderer+victim,data=dFLmurder)
summary(lm3)
```

```
Call:
lm(formula = death ~ murderer + victim, data = dFLmurder)

Residuals:
     Min       1Q   Median       3Q      Max
-0.17565 -0.12539 -0.12539 -0.05761  0.94239

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.05761    0.02963   1.944  0.05276 .
murdererW   -0.05026    0.04290  -1.172  0.24217
victimW      0.11804    0.04516   2.614  0.00937 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3115 on 323 degrees of freedom
Multiple R-squared:  0.02138,   Adjusted R-squared:  0.01532
F-statistic: 3.529 on 2 and 323 DF,  p-value: 0.03047
```

Now that we have included race of the victim, the sign on the race of the murderer has changed from positive to negative. Suddenly it seems that black murderers are *more* likely to get the death penalty by 5 percentage points. Why would include the race of the victim suddenly change our understanding of the relationship between the race of the murderer and receiving the death penalty?

Let's examine some simple tables.

```
dFLmurder |>
  group_by(murderer) |>
  summarize(deathPenalty = mean(death) )
```

```
# A tibble: 2 x 2
  murderer deathPenalty
  <chr>           <dbl>
```

```
1 B               0.102
2 W               0.119
```

Here we observe the 1.6 percentage point difference in the rate of death penalty imposition with white murderers being more likely to receive the death penalty.

```
dFLmurder |>
  group_by(murderer,victim) |>
  summarize(deathPenalty = mean(death) )
```

```
`summarise()` has grouped output by 'murderer'. You can override using the
`.groups` argument.
```

```
# A tibble: 4 x 3
# Groups:   murderer [2]
  murderer victim deathPenalty
  <chr>    <chr>         <dbl>
1 B        B            0.0583
2 B        W            0.175
3 W        B            0
4 W        W            0.126
```

Once we break it down by race of the victim, we see that black murderers are more likely to receive the death penalty when their victim is black (by 5.8 percentage points) *and* when their victim is white (by 4.9 percentage points). Our conclusions have completely changed after we introduced the race of the victim.

> **Simpson's Paradox**
>
> In some datasets, after incorporating a third variable in the analysis, the conclusions can completely reverse. This is sometimes called Yule's Paradox.

Why does this happen? In this example, the real driver of the death penalty seems to be the death of a white victim. When the victim is white, the murderer is much more likely to receive the death penalty. And who is most likely to have white victims?… white murderers. As a result, when we just compare black and white murderers it seems that white murderers are more likely to get the death penalty (which they are), but the reason we observe this is because they are the most likely to have a white victim.

Confounder

A **confounder** is any variable that is associated with *both* the outcome of interest and the treatment. We are typically uninterested in the confounder's association with the outcome, but instead just want to account for it in some way.

In the Florida murderers data, the race of the victim is associated with both the race of the murderer and the likelihood of receiving the death penalty. Our research question here focuses on the race of the murderer. We need to account for the race of the victim because we would like to think of two murderers of different races but with similar circumstances and determine who is more likely to receive the death penalty.

Race of the victim is only one confounder. Other mitigating and aggravating factors not recorded in these data could also explain the imposition of the death penalty, such as age of the victim, relationship to the victim, and intentionality of the homicide. Our thoughts about the racial differences in the death penalty could be reversed yet again with another confounder.

1.2 Example: Simulated data (x correlated with t)

Now that we know confounders are important, let's see whether a tool like regression is effective at accounting for potential confounders. Here I simulate some data where

$$y_i = 0 + 0 \times t_i + x_{1i} + x_{2i} + 4x_{1i}x_{2i} + N(0,1)$$

where t_i is a 0/1 treatment assignment indicator and x_1 and x_2 are 0/1 confounders. I have simulated x_1 and x_2 so that they are more likely to be 1 when $t = 1$. Importantly, the outcome does not depend on t since the coefficient in front of t is 0, meaning that in this simulated example there is no treatment effect. Lastly, note that there is a rather large interaction effect. When x_1 and x_2 are both equal to 1, then the expected value of y increases by 4.

```
set.seed(06182001)
beta <- c(0,1,1,4)
dataCor <- data.frame(treat=rep(0:1,each=100)) |>
  mutate(x1 = rbinom(200, 1, 0.3*treat + 0.9*(1-treat)),
         x2 = rbinom(200, 1, 0.3*treat + 0.9*(1-treat)),
         y  = rnorm(200, cbind(1,x1,x2,x1*x2)%*%beta, 1))
```

The traditional social science approach is the regression model with additive terms for the treatment assignment, x_1, and x_2, expecting the $+x_1 + x_2$ part to "control for" the confounders.

```r
summary(lm(y~treat+x1+x2, data=dataCor))
```

```
Call:
lm(formula = y ~ treat + x1 + x2, data = dataCor)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5435 -0.8583  0.1989  0.9039  2.9764

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.08564    0.34940   0.245  0.80663
treat       -0.79105    0.29322  -2.698  0.00759 **
x1           2.65377    0.25975  10.217  < 2e-16 ***
x2           2.95462    0.24213  12.202  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.39 on 196 degrees of freedom
Multiple R-squared:  0.7763,    Adjusted R-squared:  0.7729
F-statistic: 226.8 on 3 and 196 DF,  p-value: < 2.2e-16
```

Yikes! This regression model found a significant treatment effect! But we simulated these data precisely so that there would be no treatment effect. Revisit how we simulated these data. *treat* is not involved in determining the value of y. y only depends on x_1 and x_2, yet the regression model is telling us that $treat

In this case, the failure to include an important x_1x_2 interaction term caused us to make an incorrect inference about the effect of the treatment on the outcome. If we were clever enough to realize that there was an important interaction term, then we could simply add it in and correctly conclude that there was no treatment effect.

```r
summary(lm(y~treat+x1+x2+x1:x2, data=dataCor))
```

```
Call:
lm(formula = y ~ treat + x1 + x2 + x1:x2, data = dataCor)

Residuals:
     Min       1Q   Median       3Q      Max
-2.88473 -0.60375 -0.01706  0.63581  2.33238
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.03804    0.24511   0.155  0.87681
treat        0.16258    0.21628   0.752  0.45313
x1           0.75209    0.22579   3.331  0.00104 **
x2           0.90262    0.22261   4.055 7.25e-05 ***
x1:x2        4.46055    0.31281  14.260  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9751 on 195 degrees of freedom
Multiple R-squared:  0.8905,    Adjusted R-squared:  0.8883
F-statistic: 396.5 on 4 and 195 DF,  p-value: < 2.2e-16
```

This is the problem. In order to be sure to fully account for confounders, linear regression requires that we get the model structure right. This is very hard. There are a lot of methods for regression diagnostics, but they are complicated and become more complicated as the number of features increases. In datasets with dozens or hundreds of features, it becomes challenging to consider fully all of the potential interaction effects and non-linear effects. Neglecting an important one can drastically change your conclusions.

Before we move on to a different simulation, note that this dataset had a strong correlation between $\mathbf{x}$ and t and, at the same time, t was strongly associated with y.

```
cor(dataCor$x1, dataCor$treat)
```

```
[1] -0.6393057
```

```
cor(dataCor$x2, dataCor$treat)
```

```
[1] -0.5673086
```

```
aggregate(y~treat, data=dataCor, mean)
```

```
  treat        y
1     0 5.015015
2     1 0.897501
```

1.3 Example: Simulated data (x uncorrelated with t)

Here we will explore a similar simulation. However, this time $\mathbf{x}$ will be uncorrelated with t. I added some correlation between x_1 and x_2, but both are completely unassociated with t. As before, y has no association with t, but a particularly strong association with the x_1x_2 interaction.

```
# independent
set.seed(06182001)
dataInd <- data.frame(treat=rep(0:1,each=100))
dataInd$x1 <- rbinom(200, 1, 0.6)
dataInd$x2 <- rbinom(200, 1, 0.3*dataInd$x1 + 0.9*(1-dataInd$x1))
beta <- c(0,1,1,4)
dataInd$y  <- rnorm(200, with(dataInd, cbind(1,x1,x2,x1*x2)%*%beta), 1)
```

This simulation is like what happens in a randomized trial. In randomized trials, the treatment assignment is unrelated to any of the x_j by the researcher's design.

If we use the simplest regression approach to check whether there is a treatment effect, we correctly conclude that there is no treatment effect. When t was correlated with $\mathbf{x}$ in our previous simulation we found the opposite.

```
summary(lm(y~treat, data=dataInd))
```

```
Call:
lm(formula = y ~ treat, data = dataInd)

Residuals:
    Min      1Q  Median      3Q     Max
-3.6364 -1.5653 -0.7089  0.5261  6.1881

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.9050     0.2295   8.301 1.59e-14 ***
treat         0.3425     0.3245   1.055    0.293
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.295 on 198 degrees of freedom
Multiple R-squared:  0.005593,  Adjusted R-squared:  0.0005707
F-statistic: 1.114 on 1 and 198 DF,  p-value: 0.2926
```

Let's go ahead and adjust for **x** and see if it matters.

```
summary(lm(y~treat+x1+x2, data=dataInd))
```

```
Call:
lm(formula = y ~ treat + x1 + x2, data = dataInd)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3634 -0.8332 -0.0822  0.6639  4.3394

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.35221    0.23566  -9.981   <2e-16 ***
treat       -0.07211    0.18023  -0.400     0.69
x1           3.98421    0.21135  18.851   <2e-16 ***
x2           3.80216    0.20841  18.243   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.262 on 196 degrees of freedom
Multiple R-squared:  0.7024,    Adjusted R-squared:  0.6978
F-statistic: 154.2 on 3 and 196 DF,  p-value: < 2.2e-16
```

Still a null treatment effect, one that is even closer to 0 and with a smaller standard error.

Let's add in the important missing interaction term.

```
summary(lm(y~treat+x1*x2, data=dataInd))
```

```
Call:
lm(formula = y ~ treat + x1 * x2, data = dataInd)

Residuals:
     Min       1Q   Median       3Q      Max
-2.92707 -0.62959 -0.04675  0.69941  2.27152

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.16154    0.28336   0.570 0.569275
treat        0.05738    0.13958   0.411 0.681445
```

```
x1           1.03411    0.30269   3.416 0.000772 ***
x2           0.75288    0.30876   2.438 0.015648 *
x1:x2        4.15820    0.35936  11.571  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9741 on 195 degrees of freedom
Multiple R-squared:  0.8235,    Adjusted R-squared:  0.8199
F-statistic: 227.5 on 4 and 195 DF,  p-value: < 2.2e-16
```

We get a treatment effect estimate that is even smaller and a standard error that shows even more precision.

> **Note**
>
> When the treatment assignment (t) is uncorrelated with the other features in the dataset ($\mathbf{x}$), then the inclusion or exclusion of $\mathbf{x}$ has little effect on the overall conclusion. Including $+x_1 + x_2 + ...$ and correctly guessing its functional form can increase precision.

What if we could first eliminate the correlation between t and $\mathbf{x}$ before fitting the regression model? In that case we would not have to worry so much if we include or fail to include certain features or transform them in a particular way.

2 Weights to uncorrelate t and $\mathbf{x}$

The distribution of x_1 is completely different among the treated cases compared to the control cases. Among those where $t = 0$ we need a lot more $x_1 = 0$ and a lot fewer $x_1 = 1$.

```
a <- dataCor |> count(x1, treat)
a
```

```
  x1 treat  n
1  0     0 10
2  0     1 73
3  1     0 90
4  1     1 27
```

If each of the 10 control cases with $x_1 = 0$ were weighted by 73/10 and each of the 90 control cases with $x_1 = 1$ were weighted by 27/90, then the treatment and control groups would resemble each other. The 10 control cases with weights 73/10 would have total weight 73. The 90 control cases with weights 27/90 would have total weight 27.

Let's add a new weight column to our data to reflect this idea and see what happens to the table and to the correlation between x_1 and t.

```
dataCor <- dataCor |>
  mutate(w=case_when(
    treat==1 ~ 1,
    treat==0 & x1==0 ~ a$n[2]/a$n[1],
    treat==0 & x1==1 ~ a$n[4]/a$n[3]
))
head(dataCor, 10)
```

```
   treat x1 x2          y   w
1      0  1  1  6.4177368 0.3
2      0  1  1  6.2164648 0.3
3      0  1  1  7.7950744 0.3
4      0  1  1  5.7997508 0.3
5      0  1  1  6.0748075 0.3
6      0  1  1  6.1228532 0.3
7      0  1  1  6.5539440 0.3
8      0  1  1  5.6138476 0.3
9      0  1  0  0.5535691 0.3
10     0  0  1 -0.1484591 7.3
```

Now let's check out the *weighted* total in the treatment and control groups

```
dataCor |>
  group_by(x1, treat) |>
  summarize(sum(w))
```

```
`summarise()` has grouped output by 'x1'. You can override using the `.groups`
argument.
```

```
# A tibble: 4 x 3
# Groups:   x1 [2]
     x1 treat `sum(w)`
  <int> <int>    <dbl>
1     0     0       73
2     0     1       73
3     1     0       27
4     1     1       27
```

In R, the best method for handling weighted data is through the `survey` package.

```
library(survey)
sdesign <- svydesign(ids=~1, weights=~w, data=dataCor)
corXt <-
  svyvar(~treat+x1+x2, design=sdesign) |>
  as.matrix() |>
  cov2cor() |>
  zapsmall()
corXt[,]
```

```
          treat        x1         x2
treat  1.000000 0.0000000 -0.4485170
x1     0.000000 1.0000000  0.0091104
x2    -0.448517 0.0091104  1.0000000
```

Now t is completely uncorrelated with x_1. The correlation matrix still shows very large correlation between t and x_2. Unfortunately, we are only part of the way there since we need to uncorrelate x_1 and x_2 at the same time.

```
a <- dataCor |> count(x1, x2, treat)
a
```

```
  x1 x2 treat  n
1  0  0     0  3
2  0  0     1 48
3  0  1     0  7
4  0  1     1 25
5  1  0     0 11
6  1  0     1 22
7  1  1     0 79
8  1  1     1  5
```

Now it seems that we need 4 different weights for control cases to deal with all the (x_1, x_2) combinations. You can probably see that this is going to get exponentially more complex as we add more features, possibly discrete features with many categorical levels (like state or country) or continuous features (like age, SES, test scores).

Let's rework the weight for control cases where $\mathbf{x}' = [0\ 0]$. From the table above we can see that this weight should be 48/3.

$$\begin{aligned} w(x_1 = 0, x_2 = 0) &= \frac{48}{3} \\ &= \frac{48}{3}\frac{\frac{1}{48+3}}{\frac{1}{48+3}} \\ &= \frac{\frac{48}{51}}{\frac{3}{51}} \\ &= \frac{\frac{48}{51}}{1-\frac{48}{51}} \\ &= \frac{P(t=1|x_1=0, x_2=0)}{1-P(t=1|x_1=0, x_2=0)} \end{aligned}$$

An equivalent way of thinking about the weight is the odds that an observation with features $\mathbf{x}$ is a member of the treatment group. Instead of building a table of all the combinations of $\mathbf{x}$ and t, we can compute the odds of treatment assignment. Logistic regression is a perfect tool for computing the odds.

```
# fit a fully interacted logistic regression model
logit1 <- glm(treat~x1*x2,data=dataCor,family=binomial)
# generate a table of all combinations of x1 and x2
wCheck <- expand.grid(x1=0:1, x2=0:1) |>
  arrange(x1,x2)
# predict by default is on log odds scale, exp() to get odds scale
wCheck$w <- predict(logit1, newdata=wCheck) |>
  exp() |>
  zapsmall()
wCheck
```

```
  x1 x2         w
1  0  0 16.000000
2  0  1  3.571429
3  1  0  2.000000
4  1  1  0.063291
```

With very little work we have produced all the weights we need. Check that the weights are the same as the weights you would have computed directly from the earlier table `a`. Let's insert the appropriate weights into `dataCor` and check the correlations now.

```r
# treatment cases all get weight 1
dataCor$w <- 1
# control cases get weights p/(1-p)
dataCor$w[dataCor$treat==0] <-
  predict(logit1, newdata=subset(dataCor, treat==0)) |>
  exp()

# check that the total weights are equal now
dataCor |>
  group_by(x1, x2, treat) |>
  summarize(sum(w))
```

```
# A tibble: 8 x 4
# Groups:   x1, x2 [4]
     x1    x2 treat `sum(w)`
  <int> <int> <int>    <dbl>
1     0     0     0    48.0
2     0     0     1    48
3     0     1     0    25.0
4     0     1     1    25
5     1     0     0    22.0
6     1     0     1    22
7     1     1     0     5.00
8     1     1     1     5
```

```r
# check correlation
sdesign <- svydesign(ids=~1, weights=~w, data=dataCor)
corXt <-
  svyvar(~treat+x1+x2, design=sdesign) |>
  as.matrix() |>
  cov2cor() |>
  zapsmall()
corXt[,]
```

```
      treat         x1         x2
treat     1  0.0000000  0.0000000
x1        0  1.0000000 -0.1523733
x2        0 -0.1523733  1.0000000
```

Now we have a dataset where the weighted dataset in which the weighted comparison cases collectively resemble the treatment cases. With this weighted dataset, we do not need to be

concerned about x_1 or x_2 being confounders. They cannot be confounders because they are not associated with the treatment assignment.

What happens if we neglect to include x_1 and x_2 in our analysis?

```
summary(svyglm(y~treat, design=sdesign))
```

```
Call:
svyglm(formula = y ~ treat, design = sdesign)

Survey design:
svydesign(ids = ~1, weights = ~w, data = dataCor)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.82561    0.22964   3.595 0.000409 ***
treat        0.07189    0.28742   0.250 0.802741
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 2.576261)

Number of Fisher Scoring iterations: 2
```

We correctly conclude that there is no treatment effect. Even though we simulated data in which x_1 and x_2 had strong effects on y, neglecting their inclusion in our analysis did not really affect our findings. Including x_1 and x_2 may still be worthwhile to gain better precision.

```
summary(svyglm(y~treat+x1+x2, design=sdesign))
```

```
Call:
svyglm(formula = y ~ treat + x1 + x2, design = sdesign)

Survey design:
svydesign(ids = ~1, weights = ~w, data = dataCor)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.14895    0.25873  -0.576    0.565
treat        0.07189    0.24829   0.290    0.772
x1           1.76389    0.28375   6.216 3.00e-09 ***
```

```
x2           1.66102    0.31198   5.324 2.76e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.560209)

Number of Fisher Scoring iterations: 2
```

Note that the treatment effect estimate is *identical*. Including x_1 and x_2 did nothing to change our treatment effect estimate. However, we did decrease the standard error of the treatment effect estimate. And if we consider all main effects and the interaction effect?

```
summary(svyglm(y~treat+x1*x2, design=sdesign))
```

```
Call:
svyglm(formula = y ~ treat + x1 * x2, design = sdesign)

Survey design:
svydesign(ids = ~1, weights = ~w, data = dataCor)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.17465    0.18067   0.967  0.33489
treat        0.07189    0.18848   0.381  0.70329
x1           0.73425    0.22057   3.329  0.00104 **
x2           0.71610    0.26768   2.675  0.00810 **
x1:x2        4.75751    0.40928  11.624  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.8175212)

Number of Fisher Scoring iterations: 2
```

Still, the treatment effect estimate is unchanged, but the standard error is even smaller.

> **Note**
>
> Analytical strategies that uncorrelate the treatment assignment (t) from potential confounders ($\mathbf{x}$) are robust to the inclusion or exclusion of $\mathbf{x}$ or any of their transformations.

A powerful method for uncorrelating the treatment assignment is to build observation weights based on $P(t = 1|\mathbf{x})$, which is known as the **propensity score**.

> **Propensity score**
>
> The propensity score is the probability of assignment to treatment conditional on the observation features, $P(t = 1|\mathbf{x})$

3 Average Treatment Effect (ATE) and the Average Treatment Effect on the Treated (ATT)

In the previous simulated example, we reweighted the control cases so that the distribution of their features aligned with the feature distribution of the treated cases. We accomplished by weighting cases as

$$w_i = t_i + (1 - t_i)\frac{P(t = 1|\mathbf{x}_i)}{1 - P(t = 1|\mathbf{x}_i)}$$

This assigns weight 1 to treatment cases and weights equal to the odds of treatment assignment to the comparison cases.

Because these weights make the control cases "look like" the treatment cases, the overall distribution of x_1 and x_2 match the distribution among the treatment cases. As a result, our treatment effect estimate describes what the treatment effect would be for the kinds of individuals that receive the treatment. If there is a particular value of $\mathbf{x}$ that made it impossible for someone to be in the treatment group, then $P(t = 1|\mathbf{x}) = 0$ and any control case with that feature would receive weight 0 and be eliminated from the analysis.

Since these weights make the control cases resemble the treatment cases, our treatment effect estimate is called the *average treatment effect on the treated* (ATT). The ATE compares what actually happened to the treatment group to what would have happened had they not been treated. In the majority of social science inquiries, this is the primary quantity of interest.

1. When examining racial disparities in the justice system, we ask what would happen to members of a racial minority ("treated") if their cases were handled the same way as the cases of white defendants. We reweight cases involving white defendants to collectively resemble cases involving minority defendants. That's an ATT question.
2. When evaluating a reading program designed for schools in low income neighborhoods, we reweight students without access to the reading program to resemble the students in the schools in the low income neighborhoods with the reading program. …ATT.
3. When testing an urban health initiative, we would reweight residents in neighborhoods without the initiative to collective look like the residents in the neighborhood with the urban health initiative. Again… ATT.

In each of these questions we are not interested in evaluating the treatment on the kinds of individuals or the kinds of places for which the treatment is not relevant. We focus our assessment on the kinds of cases, students, and places that concern us most for our investigation.

In some cases, we may be interested in the *average treatment effect on the population* (ATE). The ATE estimates the difference between what would happen if the entire population of interest was treated versus the entire population was left untreated. Several public health questions fall into this category, like vaccinations. The weights for estimating the ATE are

$$w_i = t_i \frac{1}{P(t = 1|\mathbf{x}_i)} + (1 - t_i) \frac{1}{1 - P(t = 1|\mathbf{x}_i)}$$

Treated cases receive a weight of $\frac{1}{p}$ while control cases receive a weight of $\frac{1}{1-p}$. These weights reweight observations so that the distribution of $\mathbf{x}$ is the same for the treatment and control groups (just like with ATE) but that both of them also match the distribution of $\mathbf{x}$ for the population (rather than just the treatment group).

4 Neyman-Rubin-Holland causal model

Often simply called the "Rubin causal model," the foundational idea in its simplest form is that every unit of analysis has two values

- $Y_i(1)$, outcome of a case if assigned to treatment
- $Y_i(0)$, outcome of a case if assigned to control

We envision that every observation in our dataset has two *potential outcomes*, one that we would observe if they were assigned to the treatment and one that we would observe if they were assigned to the control. Then, ideally, we could compute the difference for each of them and average. This is the perfect way to estimate a causal effect.

Table 1: Structure of the Rubin causal model

Observation	Covariates	Treatment	Control	Unit causal effect
1	$\mathbf{x}_1$	$Y_1(1)$	$Y_1(0)$	$Y_1(1) - Y_1(0)$
2	$\mathbf{x}_2$	$Y_2(1)$	$Y_2(0)$	$Y_2(1) - Y_2(0)$
3	$\mathbf{x}_3$	$Y_3(1)$	$Y_3(0)$	$Y_3(1) - Y_3(0)$
⋮	⋮	⋮	⋮	⋮
n	$\mathbf{x}_n$	$Y_n(1)$	$Y_n(0)$	$Y_n(1) - Y_n(0)$
			$ATE =$	$\frac{1}{n} \sum Y_i(1) - Y_i(0)$

This idealized model, however, is faced with the fundamental problem of causal inference (Rubin 1974).

Fundamental problem of causal inference

For any observation we only get to observe $Y_i(1)$ *or* $Y_i(0)$, but never both.

In spite of this challenge, under certain assumptions we can estimate some key quantities of interest.

$$\begin{aligned} ATT &= \mathbb{E}(Y(1) - Y(0)|t = 1) \\ &= \mathbb{E}(Y(1)|t = 1) - \mathbb{E}(Y(0)|t = 1) \\ ATE &= \mathbb{E}(Y(1) - Y(0)) \end{aligned}$$

Remember that for each observation we only observe one of the potential outcomes. So, we can easily estimate two quantities.

$$\begin{aligned} \mathbb{E}(Y(1)|t = 1) &\approx \frac{1}{n_1} \sum y_i t_i \\ \mathbb{E}(Y(0)|t = 0) &\approx \frac{1}{n_0} \sum y_i (1 - t_i) \end{aligned}$$

In general

$$\mathbb{E}(Y(0)|t = 0) \neq \mathbb{E}(Y(0)|t = 1)$$

But if cases are randomized to treatment then

$$\mathbb{E}(Y(0)|t = 0) = \mathbb{E}(Y(0)|t = 1)$$

The assumption of strong ignorability is essential for causal inference from observational data.

Strong ignorability

Treatment assignment is *strongly ignorable* if the potential outcomes are independent of treatment assignment, conditional on $\mathbf{x}$.

$$(Y(1), Y(0)) \perp\!\!\!\perp t \,|\, \mathbf{x}$$

or

$$P(t = 1|\mathbf{x}, Y(1), Y(0)) = P(t = 1|\mathbf{x})$$

Strong ignorability holds if

- $\mathbf{x}$ contains all of the confounders, or
- treatment assignment depends on $\mathbf{x}$ alone, or
- treatment is randomly assigned (special case of the previous)

Regression models assume

$$\begin{aligned}\mathbb{E}(Y(t)) &= \beta_0 + \alpha t + \beta_1 x_1 + ... + \beta_d x_d \\ &= \alpha t + \beta' \mathbf{x}\end{aligned}$$

In addition to strong ignorability, we have to assume that this is the correct relationship. As we saw previously, regression analysis is not kind to researchers when we neglect to include important terms.

5 Estimation of ATT with propensity scores

With a data set with $(y_i, t_i, \mathbf{x}_i)$ for $i = 1, ..., n$, we want to estimate

$$ATT = \mathbb{E}(Y(1)|t = 1) - \mathbb{E}(Y(0)|t = 1)$$

The first term is actually easy to estimate. It asks us "among all the individuals who received the treatment, what would their average outcome be if they receive the treatment?" That we actually observed.

$$\hat{\mathbb{E}}(Y(1)|t = 1) = \frac{\sum t_i y_i}{\sum t_i}$$

The harder one is the second term. It asks "for all the individuals who received treatment, what would their average outcome have been if they had not received the treatment?" This is a counterfactual expected value. We have no direct observations of this. But let's work through some mathematics to see how we might estimate this.

Three items to recall from our probability discussion at the beginning of the course.

Expected value of a random variable

If Y is a random variable with probability distribution $p(y)$, then expected value of Y is

$$\mathbb{E}(Y) = \int_{-\infty}^{\infty} y\, p(y)\, dy$$

Law of large numbers

If $y_1, \ldots, y_n$ is a random sample from $p(y)$, then as $n \to \infty$

$$\frac{1}{n}\sum_{i=1}^{n} g(y_i) \to \mathbb{E}(g(Y))$$

Bayes Theorem

$$p(y|t) = \frac{p(t|y)p(y)}{p(t)}$$

We will use all of these properties to work through how to estimate $\mathbb{E}(Y(0)|t=1)$. As you go through this line-by-line, ask yourself why we are able to move from one line to the next. Sometimes, it is simply algebra. Other steps involve the definition of expected value, Bayes Theorem, the assumption of strong ignorability, and the law of large numbers.

$$\begin{aligned}
\mathbb{E}(Y(0)|t=1) &= \iint y_0 p(y_0, \mathbf{x}|t=1)\, d\mathbf{x}\, dy_0 \\
&= \iint y_0 p(y_0, \mathbf{x}|t=1)\frac{p(y_0, \mathbf{x}|t=0)}{p(y_0, \mathbf{x}|t=0)}\, d\mathbf{x}\, dy_0 \\
&= \iint y_0 \frac{p(y_0, \mathbf{x}|t=1)}{p(y_0, \mathbf{x}|t=0)} p(y_0, \mathbf{x}|t=0)\, d\mathbf{x}\, dy_0 \\
&= \iint y_0 \frac{p(t=1|y_0, \mathbf{x})p(y_0, \mathbf{x})}{p(t=1)} \frac{p(t=0)}{p(t=0|y_0, \mathbf{x})p(y_0, \mathbf{x})} p(y_0, \mathbf{x}|t=0)\, d\mathbf{x}\, dy_0 \\
&= \frac{p(t=0)}{p(t=1)} \iint y_0 \frac{p(t=1|y_0, \mathbf{x})}{p(t=0|y_0, \mathbf{x})} p(y_0, \mathbf{x}|t=0)\, d\mathbf{x}\, dy_0 \\
&= \frac{p(t=0)}{p(t=1)} \iint y_0 \frac{p(t=1|\mathbf{x})}{p(t=0|\mathbf{x})} p(y_0, \mathbf{x}|t=0)\, d\mathbf{x}\, dy_0 \\
&= \frac{p(t=0)}{p(t=1)} \iint y_0 \frac{p(t=1|\mathbf{x})}{1-p(t=1|\mathbf{x})} p(y_0, \mathbf{x}|t=0)\, d\mathbf{x}\, dy_0 \\
&\approx \frac{p(t=0)}{p(t=1)} \frac{\sum_{i=1}^{n}(1-t_i) y_i \frac{p(t=1|\mathbf{x}_i)}{1-p(t=1|\mathbf{x}_i)}}{\sum_{i=1}^{n}(1-t_i)} \\
&\approx \frac{\sum_{i=1}^{n}(1-t_i) y_i \frac{p(t=1|\mathbf{x}_i)}{1-p(t=1|\mathbf{x}_i)}}{\sum_{i=1}^{n}(1-t_i) \frac{p(t=1|\mathbf{x}_i)}{1-p(t=1|\mathbf{x}_i)}}
\end{aligned}$$

The final step replacing $\frac{p(t=1)}{1-p(t=1)}\sum(1-t_i)$ with $\sum(1-t_i)\frac{p(t=1|\mathbf{x}_i)}{1-p(t=1|\mathbf{x}_i)}$ is not necessarily obvious,

but here are the details.

$$\begin{aligned}
1 &= \iint p(y_0, \mathbf{x}|t=1)\, d\mathbf{x}\, dy_0 \\
&= \iint p(y_0, \mathbf{x}|t=1)\frac{p(y_0, \mathbf{x}|t=0)}{p(y_0, \mathbf{x}|t=0)}\, d\mathbf{x}\, dy_0 \\
&= \iint y_0 \frac{p(t=1|y_0, \mathbf{x})p(y_0, \mathbf{x})}{p(t=0|y_0, \mathbf{x})p(y_0, \mathbf{x})}\frac{p(t=0)}{p(t=1)} p(y_0, \mathbf{x}|t=0)\, d\mathbf{x}\, dy_0 \\
&= \frac{p(t=0)}{p(t=1)} \iint y_0 \frac{p(t=1|\mathbf{x})}{p(t=0|\mathbf{x})} p(y_0, \mathbf{x}|t=0)\, d\mathbf{x}\, dy_0 \\
&\approx \frac{p(t=0)}{p(t=1)} \frac{\sum_{i=1}^n (1-t_i)\frac{p(t=1|\mathbf{x}_i)}{1-p(t=1|\mathbf{x}_i)}}{\sum_{i=1}^n (1-t_i)} \\
\frac{p(t=1)}{p(t=0)} \sum_{i=1}^n (1-t_i) &\approx \sum_{i=1}^n (1-t_i)\frac{p(t=1|\mathbf{x}_i)}{1-p(t=1|\mathbf{x}_i)}
\end{aligned}$$

This formally shows that the approach we took earlier of weighting comparison cases with $\frac{p}{1-p}$ is a general approach for estimating ATT. The propensity score estimator for ATT is

$$\begin{aligned}
\widehat{ATT} &= \hat{\mathbb{E}}(Y(1)|t=1) - \hat{\mathbb{E}}(Y(0)|t=1) \\
&= \frac{\sum t_i y_i}{\sum t_i} - \frac{\sum (1-t_i) w_i y_i}{\sum (1-t_i) w_i}
\end{aligned}$$

where $w_i = \frac{p(t=1|\mathbf{x}_i)}{1-p(t=1|\mathbf{x}_i)}$.

The remaining problem is that we need to estimate the propensity score, $p(t=1|\mathbf{x}_i)$. We will first work through an example with the Florida murders dataset and then we will work on a larger dataset.

6 Example: Florida murderers revisited

Let's start with the easy part.

$$\hat{\mathbb{E}}(Y(1)|t=1) = \frac{\sum t_i y_i}{\sum t_i}$$

We have all of our "treated" cases and we get to observe the outcome for them when they are in the treatment.

```
dFLmurder |>
  filter(murderer=="B") |>
  summarize(deathPenrate = mean(death))
```

```
  deathPenrate
1    0.1024096
```

To compute $\hat{\mathbb{E}}(Y(0)|t=1)$ we need to start with getting the propensity score weight for each murderer.

```
# predict treatment assignment from confounders
ps1 <- glm((murderer=="B")~victim, data=dFLmurder, family=binomial)
# set propensity score weights
dFLmurder$w <- 1
dFLmurder$w[dFLmurder$murderer=="W"] <-
  predict(ps1, newdata=subset(dFLmurder, murderer=="W")) |>
  exp()
```

Let's check that black and white murderers now have the same distribution of victim race after weighting.

```
dFLmurder |>
  group_by(murderer, victim) |>
  summarize(totW=sum(w)) |>
  group_by(murderer) |>
  mutate(pct = totW/sum(totW))
```

```
# A tibble: 4 x 4
# Groups:   murderer [2]
  murderer victim  totW   pct
  <chr>    <chr>  <dbl> <dbl>
1 B        B      103   0.620
2 B        W       63   0.380
3 W        B      103.  0.620
4 W        W       63.0 0.380
```

The distribution of victim race is now identical for black and white murderers. We are now set to compute

$$\hat{\mathbb{E}}(Y(0)|t=1) = \frac{\sum(1-t_i)w_i y_i}{\sum(1-t_i)w_i}$$

```
dFLmurder |>
  filter(murderer=="W") |>
  summarize(deathPenRateW = weighted.mean(death,w))
```

```
  deathPenRateW
1    0.04775393
```

Putting this all into a regression framework allows us to compute ATT and its standard error more easily.

```
sdesign <- svydesign(ids=~1, weights=~w, data=dFLmurder)
glm1 <- svyglm(death~murderer, design=sdesign)
summary(glm1)
```

```
Call:
svyglm(formula = death ~ murderer, design = sdesign)

Survey design:
svydesign(ids = ~1, weights = ~w, data = dFLmurder)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.10241    0.02357   4.345 1.86e-05 ***
murdererW   -0.05466    0.02765  -1.977   0.0489 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.06890908)

Number of Fisher Scoring iterations: 2
```

Our estimate of the ATT is -0.055, suggesting that the black murderer death penalty rate would be -5.5 percentage points smaller had they been treated as white murderers.

We can extract the ATT and a 95% confidence interval from the regression model as

```
a <- coef(summary(glm1))
(a["murdererW","Estimate"] + c(0,-1,1)*1.96*a["murdererW","Std. Error"]) |>
  round(3)
```

```
[1] -0.055 -0.109  0.000
```

Could we get a more precise estimate for the ATT? Recall that in our simulation we tried adding some features to our model even when they were uncorrelated with the treatment assignment. When features are predictive of the outcome, then we can get more precision. Let's give it a try.

```r
glm1 <- svyglm(death~murderer+victim, design=sdesign)
summary(glm1)
```

```
Call:
svyglm(formula = death ~ murderer + victim, design = sdesign)

Survey design:
svydesign(ids = ~1, weights = ~w, data = dFLmurder)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.05645    0.02144   2.633  0.00887 **
murdererW   -0.05466    0.02533  -2.158  0.03168 *
victimW      0.12109    0.02983   4.059 6.19e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.06544564)

Number of Fisher Scoring iterations: 2
```

```r
a <- coef(summary(glm1))
(a["murdererW","Estimate"] + c(0,-1,1)*1.96*a["murdererW","Std. Error"]) |>
  round(3)
```

```
[1] -0.055 -0.104 -0.005
```

Including the victim's race as a covariate has no effect on the ATT, but it does increase the precision of the ATT estimate.

It turns out that for least squares and Poisson regression, you can simply add covariates of interest to the regression model and the treatment effect estimate will simply be the coefficient in front of the treatment indicator. For other regression models, and specifically for logistic regression, extracting the ATT takes a little more work.

7 Boosted propensity score estimation with NSDUH

In this section we are going to use propensity score estimation with the National Survey on Drug Use and Health (NSDUH) to estimate the effect of past heroin use on current year arrest.

NSDUH provides national data on tobacco, alcohol, drug use, mental health, and related health topics. The data collection has been ongoing since 1971 and has been conducted every year since. The most recent data collection had 70,000 people interviewed. The Substance Abuse and Mental Health Services Administration (SAMHSA) with the Department of Health and Human Services manages the data collection. RTI International has been collecting the data since 1988.

Previously we used logistic regression to estimate propensity scores. All the complaints I had about regression being insufficient for estimating causal effects, map over to using logistic regression for propensity scores. That is, failure to include important non-linear and interaction effects in the propensity score model will result in bad propensity score estimates and incorrect treatment effect estimates. For this reason, we will move beyond using logistic regression to estimate the propensity score and use boosting instead. The use of boosting for propensity score estimation was originally developed in McCaffrey, Ridgeway, and Morral (2004).

Let's start by loading up the NSDUH 2010 dataset.

```
load("data/nsduh.Rdata")
nsduh <- da32722.0001
```

We're going to study whether past heroin use is associated with arrest in the last year. We will set `heroin` to 1 if the respondent indicated that they had last used heroin more than 12 months ago (and none in the last year). I exclude respondents reporting heroin use in the current year. We will set `arrest` to 1 if the respondent reports being arrested in the prior 12 months. Responses that are `NA` indicate no heroin use or no arrests.

```
# NOBOOKY measures number of times arrested and booked in previous 12 months
# NA = logical skip, blank, don't know... here assume no arrest
# 1 = 1 arrest
# 2 = 2 arrests
# 3 = 3 or more arrests
nsduh <- nsduh |>
  mutate(arrest = as.numeric(!is.na(NOBOOKY2) & NOBOOKY2>0)) |>
# HERREC measures last time used
# 1 = Within the past 30 days
# 2 = More than 30 days ago but within the past 12 mos
# 3 = More than 12 months ago
# 8 = Used at some point in the past 12 mos
# 9 = Used sometime in lifetime
# NA = never used, refused, blank
  filter(!(HERREC %in% c(1,2,8))) |>
  mutate(heroin = as.numeric(!is.na(HERREC) & HERREC %in% c(3,9)))
```

We will consider a handful of potential confounders, race, sex, marital status, and age. Note that a marital status of `NA` is considered a legitimate value and those respondents are not dropped from the analysis.

```
nsduh <- nsduh |>
  mutate(NEWRACE2 = factor(NEWRACE2,
                           levels=1:7,
                           labels=c("W","B","NA.AN","PI","A","Multi","H")),
         IRSEX = factor(IRSEX,
                        levels=1:2,
                        labels=c("M","F")),
         IRMARIT = ifelse(is.na(IRMARIT), 5, IRMARIT),
         IRMARIT = factor(IRMARIT,
                          levels=1:5,
                          labels=c("M","W","D","N","NA")),
         CATAG7 = factor(CATAG7,
                          levels=1:7,
                          labels=c("12-13","14-15","16-17","18-20",
                                   "21-25","26-34","35+")))
```

NSDUH has sampling weights `ANALWT_C`. These reweight the observations to collectively resemble the United States population. NSDUH oversamples people and oversamples in places where they believe they are more likely to get respondents answering yes to drug use. The weights are essential for adjusting the sample back to reflecting the population. To properly account for sampling weights in propensity score analysis, you should include them as weights in the propensity score model and multiply the resulting propensity score weights by the sampling weight in the outcome model (Ridgeway et al. 2015). We will do this in the example below.

Let's see how the treatment (prior heroin use) and control (no heroin use) groups differ on these features.

```
nsduh |>
  group_by(heroin,NEWRACE2) |>
  summarize(totW=sum(ANALWT_C)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW,
              values_fill = 0)
```

```
# A tibble: 7 x 3
  NEWRACE2     `0`     `1`
```

```
  <fct>      <dbl>   <dbl>
1 W        0.670   0.710
2 B        0.120   0.0957
3 NA.AN    0.00473 0.0160
4 PI       0.00298 0
5 A        0.0458  0.00435
6 Multi    0.0123  0.0151
7 H        0.145   0.159
```

Note that 71% of those reporting heroin use are white, while 67% of those reporting no heroin use are white. There are smaller differences for black and Hispanic respondents, but clear evidence that race is associated with heroin use.

```
nsduh |>
  group_by(heroin,IRSEX) |>
  summarize(totW=sum(ANALWT_C)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW)
```

```
# A tibble: 2 x 3
  IRSEX   `0`   `1`
  <fct> <dbl> <dbl>
1 M     0.483 0.710
2 F     0.517 0.290
```

Also, sex of the respondent appears to be associated with heroin use with men making up 71% of the heroin users compared with 48% of the non-heroin users.

```
nsduh |>
  group_by(heroin,IRMARIT) |>
  summarize(totW=sum(ANALWT_C)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW,
              values_fill = 0)
```

```
# A tibble: 5 x 3
  IRMARIT    `0`      `1`
  <fct>    <dbl>    <dbl>
1 M       0.479  0.302
```

```
2 W       0.0585 0.0220
3 D       0.126  0.309
4 N       0.289  0.367
5 NA      0.0472 0.000816
```

Marital status seems strongly associated with heroin use, with 30% of heroin users reported being married and 48% of non-heroin users reported being married. 68% of heroin users reported being divorced or never married compared to 42% for non-heroin users.

```
nsduh |>
  group_by(heroin,CATAG7) |>
  summarize(totW=sum(ANALWT_C)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW,
              values_fill = 0)
```

```
# A tibble: 7 x 3
  CATAG7    `0`      `1`
  <fct>   <dbl>    <dbl>
1 12-13  0.0317 0.000119
2 14-15  0.0321 0.00257
3 16-17  0.0335 0.00442
4 18-20  0.0533 0.0311
5 21-25  0.0808 0.0899
6 26-34  0.143  0.210
7 35+    0.626  0.662
```

Heroin use particularly afflicts 26-34 year olds disproportionately. All of these respondent features, race, sex, marital status, and age all are associated with prior heroin use. And all of them could plausibly be related to arrest as well. Therefore, it is essential that we properly account for their potential confounding effect.

7.1 Propensity score estimation with logistic regression

We begin by estimating the propensity score. Since NSDUH has sampling weights, we will first create a survey design object so that R knows that `ANALWT_C` is a special column in `nsduh` that needs to be incorporated into all analyses (weighted means, weighted variances, weighted regressions, etc.).

```
sdesign <- svydesign(ids=~1, weights=~ANALWT_C, data=nsduh)
```

What would we conclude without any adjustment for demographic characteristics? Respondents with prior heroin use are 24 percentage points more likely to report being arrested in the last 12 months compared with non-heroin respondents.

```
svyglm(arrest~heroin, design=sdesign) |>
  summary()
```

```
Call:
svyglm(formula = arrest ~ heroin, design = sdesign)

Survey design:
svydesign(ids = ~1, weights = ~ANALWT_C, data = nsduh)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.027004   0.001084  24.906  < 2e-16 ***
heroin      0.089607   0.018818   4.762 1.92e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.02731343)

Number of Fisher Scoring iterations: 2
```

Let's pursue our propensity score strategy by fitting a standard logistic regression model predicting prior heroin use from the four demographic variables of interest. We use `quasibinomial` here just so that R does not complain about non-integer weights.

```
ps1 <- svyglm(heroin ~ NEWRACE2+IRSEX+IRMARIT+CATAG7,
              design=sdesign,
              family=quasibinomial)
```

All heroin users receive weight 1 and all non-heroin users receive weights $\frac{p}{1-p}$, where p is the propensity score.

```
nsduh$w <- 1
nsduh$w[nsduh$heroin==0] <- nsduh |>
  filter(heroin==0) |>
  predict(ps1, newdata=_) |>
  exp()
```

Even though we already used `ANALWT_C` in constructing the propensity scores, we need to multiply the propensity score weights by `ANALWT_C` to get the right answers. Gory details are in Ridgeway et al. (2015).

```
nsduh <- nsduh |>
  mutate(w = w*ANALWT_C)
```

If all of our prior reasoning is correct, then the propensity score weights should uncorrelate `heroin` and each of the demographic variables. Let's check their weighted distributions.

```
nsduh |>
  group_by(heroin,NEWRACE2) |>
  summarize(totW=sum(w)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW,
              values_fill = 0)
```

```
# A tibble: 7 x 3
  NEWRACE2          `0`     `1`
  <fct>           <dbl>   <dbl>
1 W         0.711         0.710
2 B         0.0960        0.0957
3 NA.AN     0.0156        0.0160
4 PI        0.0000000133  0
5 A         0.00435       0.00435
6 Multi     0.0152        0.0151
7 H         0.158         0.159
```

```
nsduh |>
  group_by(heroin,IRSEX) |>
  summarize(totW=sum(w)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW)
```

```
# A tibble: 2 x 3
  IRSEX   `0`   `1`
  <fct> <dbl> <dbl>
1 M     0.710 0.710
2 F     0.290 0.290
```

```
nsduh |>
  group_by(heroin,IRMARIT) |>
  summarize(totW=sum(w)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW)
```

```
# A tibble: 5 x 3
  IRMARIT      `0`      `1`
  <fct>      <dbl>    <dbl>
1 M       0.302    0.302
2 W       0.0221   0.0220
3 D       0.308    0.309
4 N       0.367    0.367
5 NA      0.000816 0.000816
```

```
nsduh |>
  group_by(heroin,CATAG7) |>
  summarize(totW=sum(w)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW)
```

```
# A tibble: 7 x 3
  CATAG7      `0`      `1`
  <fct>     <dbl>    <dbl>
1 12-13  0.000119 0.000119
2 14-15  0.00258  0.00257
3 16-17  0.00443  0.00442
4 18-20  0.0313   0.0311
5 21-25  0.0903   0.0899
6 26-34  0.210    0.210
7 35+    0.661    0.662
```

Everything appears to have worked out! The heroin and non-heroin groups now have the same race distribution, sex distribution, distribution of marital status, and age distribution. This is what we should see if heroin usage were randomly assigned to respondents. Now that `heroin` is uncorrelated with the demographic features, any regression analysis we conduct will be more robust to their inclusion or exclusion or use of interactions or lack of interaction effects.

7.2 Outcome analysis

The column w contains our propensity score weights multiplied by the NSDUH sampling weights. We first set up a new survey design object with these new weights and proceed with a weighted regression model to compute the ATT.

```
sdesign <- svydesign(ids=~1, weights=~w, data=nsduh)
lm1 <- svyglm(arrest ~ heroin, design=sdesign)
summary(lm1)
```

```
Call:
svyglm(formula = arrest ~ heroin, design = sdesign)

Survey design:
svydesign(ids = ~1, weights = ~w, data = nsduh)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.044266   0.002941  15.052  < 2e-16 ***
heroin      0.072345   0.019016   3.804 0.000142 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.0726703)

Number of Fisher Scoring iterations: 2
```

```
# report a 95% confidence interval
a <- coef(summary(lm1))
round(a["heroin","Estimate"] + c(0,-1,1)*1.96*a["heroin","Std. Error"], 3)
```

```
[1] 0.072 0.035 0.110
```

This is only a modest change from our original estimate that involved no adjustment for potential confounders. As before we can also include the demographic features in an effort to increase precision of our ATT.

```
lm1 <- svyglm(arrest ~ heroin+NEWRACE2+IRSEX+IRMARIT+CATAG7,
              design=sdesign)
a <- summary(lm1) |>
  coef()
round(a["heroin","Estimate"] + c(0,-1,1)*1.96*a["heroin","Std. Error"], 3)
```

```
[1] 0.072 0.036 0.108
```

We get about a minimal reduction on the width of the confidence interval (a modest reduction) and no change in the actual estimate of ATT as expected.

All of this analysis is based on the assumption that we really uncorrelated those demographic features from prior heroin use. But did we? The balance tables shown previously just looked at the marginal distribution of the demographic features and not their interactions. Perhaps we are still doing the analysis wrong and neglecting to account for important interactions.

If we compare the treatment and comparison groups in terms of a race/sex interaction, we see some differences.

```
nsduh |>
  group_by(heroin,NEWRACE2,IRSEX) |>
  summarize(totW=sum(w)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW,
              values_fill = 0) |>
  mutate(`0`=zapsmall(`0`)) |>
  filter(`1`>0.01)
```

```
# A tibble: 8 x 4
  NEWRACE2 IRSEX     `0`    `1`
  <fct>    <fct>   <dbl>  <dbl>
1 W        M     0.510   0.462
2 W        F     0.202   0.248
3 B        M     0.0635  0.0764
4 B        F     0.0325  0.0193
5 NA.AN    M     0.00940 0.0136
6 Multi    M     0.0102  0.0127
7 H        M     0.114   0.142
8 H        F     0.0440  0.0173
```

It is clear that we have not really matched well on this interaction. Whether the differences observed here matter all depends on how much the interaction affects the arrest outcome. Even if it does not matter in this example, we need to be concerned about those research questions with a large number of potential confounders. Also, all of the demographic features included here are categorical features. Non-linear effects may arise in the propensity score stage and failure to include them can result in incorrect estimates. We need a more robust method for estimating propensity scores.

7.3 Propensity score estimation with boosting

We will first walk through an example of obtaining boosted propensity scores directly using `gbmt()`. Then we will use the `fastDR` package that integrates a lot of these ideas.

```
library(gbm3)
```

We start by estimating a propensity score model, predicting `heroin` from the demographic features. Some aspects of fitting the generalized boosted model change when estimating propensity scores. The primary change is our motivation. We are no longer interested in out-of-sample predictive performance. There is no future dataset that we are trying to predict. We are trying to understand for this specific dataset, how to explain which observations are in the treatment group and which ones are in the comparison group. The implication is that we do not do any cross-validation and we set `bag.fraction=1.0`. Both of these changes result in better *in-sample* predictive performance.

```
set.seed(20240316)
gbm1 <- gbmt(heroin~NEWRACE2+IRSEX+IRMARIT+CATAG7,
             data=nsduh,
             weights=nsduh$ANALWT_C,
             distribution=gbm_dist("Bernoulli"),
             train_params = training_params(
               num_trees = 3000,          # number of trees
               shrinkage = 0.003,         # lambda
               bag_fraction = 1.0,
               num_train = nrow(nsduh),
               min_num_obs_in_node = 10,
               interaction_depth = 3,     # number of splits
               num_features = 4),         # number of features
             par_details=gbmParallel(num_threads=8),
             is_verbose = FALSE)
```

Now that we have a model fit, we need to decide the optimal number of iterations. It is somewhat of an open question about what it means to optimize the propensity score model. There is general consensus that the optimal propensity score model should produce propensity scores that makes the balance tables "look good." There is also no consensus on what "looks good" means. I characterize the balance table based on the single largest difference in the table. The propensity score model is only as good as how well it balances the worst balanced feature. If there is one feature that does not align across the treatment and control groups, then there is a risk that we have not adequately accounted for its potential confounding effect.

For now we will use all 3,000 trees of our GBM model. We will consider optimizing the number of trees when we use `fastDR()`.

```
nsduh$bw <- 1
nsduh$bw[nsduh$heroin==0] <- nsduh |>
  filter(heroin==0) |>
  predict(gbm1, newdata=_, n.trees=3000) |>
  exp()

nsduh <- nsduh |>
  mutate(bw = bw*ANALWT_C)
```

Let's check the balance that we get from `gbmt()`. All of the marginal distributions look good. The largest difference in the treatment and control columns is about 0.006, less than a percentage point.

```
nsduh |>
  group_by(heroin,NEWRACE2) |>
  summarize(totW=sum(bw)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW,
              values_fill = 0)
```

```
# A tibble: 7 x 3
  NEWRACE2      `0`      `1`
  <fct>       <dbl>    <dbl>
1 W         0.716    0.710
2 B         0.0936   0.0957
3 NA.AN     0.0144   0.0160
4 PI        0.000398 0
5 A         0.00714  0.00435
6 Multi     0.0144   0.0151
7 H         0.154    0.159
```

```
nsduh |>
  group_by(heroin,IRSEX) |>
  summarize(totW=sum(bw)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW)
```

```
# A tibble: 2 x 3
```

```
  IRSEX   `0`   `1`
  <fct> <dbl> <dbl>
1 M     0.704 0.710
2 F     0.296 0.290
```

```
nsduh |>
  group_by(heroin,IRMARIT) |>
  summarize(totW=sum(bw)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW)
```

```
# A tibble: 5 x 3
  IRMARIT     `0`      `1`
  <fct>     <dbl>    <dbl>
1 M       0.305   0.302
2 W       0.0231  0.0220
3 D       0.307   0.309
4 N       0.361   0.367
5 NA      0.00396 0.000816
```

```
nsduh |>
  group_by(heroin,CATAG7) |>
  summarize(totW=sum(bw)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW)
```

```
# A tibble: 7 x 3
  CATAG7     `0`      `1`
  <fct>    <dbl>    <dbl>
1 12-13  0.00266 0.000119
2 14-15  0.00367 0.00257
3 16-17  0.00521 0.00442
4 18-20  0.0321  0.0311
5 21-25  0.0912  0.0899
6 26-34  0.205   0.210
7 35+    0.660   0.662
```

And let's check that race/sex interaction. There are some modest differences with the largest about 0.025, much smaller than the 0.048 difference we got when we used logistic regression to estimate the propensity scores.

```r
nsduh |>
  group_by(heroin,NEWRACE2,IRSEX) |>
  summarize(totW=sum(bw)) |>
  group_by(heroin) |>
  mutate(totW=totW/sum(totW)) |>
  pivot_wider(names_from = heroin, values_from = totW,
              values_fill = 0) |>
  mutate(`0`=zapsmall(`0`)) |>
  filter(`1`>0.01) |>
  print(n=Inf)
```

```
# A tibble: 8 x 4
  NEWRACE2 IRSEX     `0`    `1`
  <fct>    <fct>   <dbl>  <dbl>
1 W        M     0.490   0.462
2 W        F     0.227   0.248
3 B        M     0.0656  0.0764
4 B        F     0.0280  0.0193
5 NA.AN    M     0.0122  0.0136
6 Multi    M     0.00995 0.0127
7 H        M     0.121   0.142
8 H        F     0.0328  0.0173
```

Finally, let's complete our outcome analysis.

```r
sdesign <- svydesign(ids=~1, weights=~bw, data=nsduh)
lm1 <- svyglm(arrest ~ heroin, design=sdesign)
summary(lm1)
```

```
Call:
svyglm(formula = arrest ~ heroin, design = sdesign)

Survey design:
svydesign(ids = ~1, weights = ~bw, data = nsduh)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.044079   0.004972   8.865  < 2e-16 ***
heroin      0.072532   0.019434   3.732  0.00019 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for gaussian family taken to be 0.07262952)

Number of Fisher Scoring iterations: 2
```

```
a <- coef(summary(lm1))
round(a["heroin","Estimate"] + c(0,-1,1)*1.96*a["heroin","Std. Error"], 3)
```

```
[1] 0.073 0.034 0.111
```

```
lm1 <- svyglm(arrest ~ heroin+NEWRACE2+IRSEX+IRMARIT+CATAG7,
              design=sdesign)
a <- coef(summary(lm1))
round(a["heroin","Estimate"] + c(0,-1,1)*1.96*a["heroin","Std. Error"], 3)
```

```
[1] 0.072 0.035 0.109
```

7.4 `fastDR` to estimate propensity score estimates (and doubly robust estimates)

`fastDR` was originally developed as the Toolkit for Weighting and Analysis of Non-equivalent Groups (twang), a package I wrote some time ago. RAND continues to develop and maintain the `twang` package (https://www.rand.org/statistics/twang.html). `twang` has many options and those options come with some computational burdens. I stripped out the features that I use all the time, upgraded the boosting algorithm to `gbm3`, and added the ability to compute doubly robust estimates.

Start by installing `fastDR` from GitHub.

```
# remotes::install_github("gregridgeway/fastDR")
library(fastDR)
library(kableExtra)
```

To estimate the propensity score estimates and the ATT, run the `fastDR()` function. You just need to separate out the outcome, treatment indicator, confounders, sampling weights, and an observation identifier.

```
nsduh$w <- NULL
ps2 <- fastDR(list(y.form=~arrest,
                   t.form=~heroin,
                   x.form=~NEWRACE2+IRSEX+IRMARIT+CATAG7,
```

```
                    weights.form=~ANALWT_C,
                    key.form=~QUESTID2),
                data=nsduh,
                shrinkage=0.006,
                n.trees=3000,
                y.dist="quasibinomial",
                verbose=FALSE,
                par_details=gbmParallel(12,1024))
```

Start by taking a peek at the resulting object and see what is stored in the `ps2` object.

```
names(ps2)
```

```
 [1] "ks"             "best.iter"      "balance.tab"    "w"
 [5] "p"              "ks.un"          "balance.tab.un" "shrinkage"
 [9] "n1"             "ESS"            "glm.un"         "glm.ps"
[13] "glm.dr"         "z"              "effects"        "y.dist"
```

The `balance.tab.un` displays the unadjusted balance table. Here we can examine how the treatment and control groups differ before propensity score weighting. The column on the far right shows the Kolmogorov-Smirnov statistic, which for these variables is just the difference in their rates. It is a little more complicated for continuous variables.

```
ps2$balance.tab.un |>
  kbl(digits = 3) |>
  kable_material_opt(lightable_options="striped")
```

The largest difference in the Table 2 is for `IRSEX` with a difference of 0.227. We can also extract that largest difference from the `ps2` object.

```
ps2$ks.un
```

```
[1] 0.2267797
```

What we are really interested in is the `balance.tab` object that contains the balance table after applying the propensity score weights.

```
ps2$balance.tab |>
  kbl(digits = 3) |>
  kable_material_opt(lightable_options="striped")
```

Table 2: Balance table for unweighted observations

	control	treatment	KS
NEWRACE2:W	0.672	0.710	0.038
NEWRACE2:B	0.120	0.096	-0.024
NEWRACE2:NA.AN	0.005	0.016	0.011
NEWRACE2:A	0.046	0.004	-0.042
NEWRACE2:Multi	0.012	0.015	0.003
NEWRACE2:H	0.145	0.159	0.014
IRSEX:M	0.483	0.710	0.227
IRSEX:F	0.517	0.290	-0.227
IRMARIT:M	0.479	0.302	-0.178
IRMARIT:W	0.059	0.022	-0.037
IRMARIT:D	0.126	0.309	0.183
IRMARIT:N	0.288	0.367	0.078
IRMARIT:NA	0.047	0.001	-0.046
CATAG7:12-13	0.032	0.000	-0.032
CATAG7:14-15	0.032	0.003	-0.030
CATAG7:16-17	0.034	0.004	-0.029
CATAG7:18-20	0.053	0.031	-0.022
CATAG7:21-25	0.081	0.090	0.009
CATAG7:26-34	0.143	0.210	0.067
CATAG7:35+	0.626	0.662	0.036

Table 3: Balance table for propensity score weighted observations

	control	treatment	KS
NEWRACE2:W	0.711	0.710	-0.002
NEWRACE2:B	0.094	0.096	0.002
NEWRACE2:NA.AN	0.016	0.016	0.000
NEWRACE2:A	0.006	0.004	-0.001
NEWRACE2:Multi	0.015	0.015	0.000
NEWRACE2:H	0.158	0.159	0.001
IRSEX:M	0.709	0.710	0.001
IRSEX:F	0.291	0.290	-0.001
IRMARIT:M	0.303	0.302	-0.001
IRMARIT:W	0.022	0.022	0.000
IRMARIT:D	0.308	0.309	0.000
IRMARIT:N	0.364	0.367	0.002
IRMARIT:NA	0.002	0.001	-0.001
CATAG7:12-13	0.001	0.000	-0.001
CATAG7:14-15	0.003	0.003	0.000
CATAG7:16-17	0.005	0.004	0.000
CATAG7:18-20	0.031	0.031	0.000
CATAG7:21-25	0.090	0.090	0.000
CATAG7:26-34	0.208	0.210	0.002
CATAG7:35+	0.662	0.662	0.000

Note that after weighting, the treatment and control groups look very similar. The largest difference in the table above is 0.002. You can access this largest KS statistic from the `ps2` object.

```
ps2$ks
```

```
[1] 0.002178828
```

We can also extract the number of treatment cases and the effective sample size of the weighted control cases.

```
ps2$n1
```

```
[1] 599
```

```
ps2$ESS
```

```
[1] 4223.858
```

The effective sample size (ESS) is the equivalent number of independent, unweighted observations that would give you the same precision as the observed, weighted data. If $\text{Var}(y) = \sigma^2$, then we are looking for ESS that solves

$$\begin{aligned}
\frac{\sigma^2}{ESS} &= \text{Var}\left(\frac{\sum w_i y_i}{\sum w_i}\right) \\
&= \frac{\text{Var}(\sum w_i y_i)}{(\sum w_i)^2} \\
&= \frac{\sum w_i^2 \text{Var}(y_i)}{(\sum w_i)^2} \\
&= \sigma^2 \frac{\sum w_i^2}{(\sum w_i)^2} \\
&= \frac{\sigma^2}{\frac{(\sum w_i)^2}{\sum w_i^2}} \\
ESS &= \frac{(\sum w_i)^2}{\sum w_i^2}
\end{aligned}$$

Since the cases all have weights, many of which could be near 0, the ESS gives a useful way of gauging the amount of information in the data for the estimate of $\hat{\mathbb{E}}(Y(0)|t = 1)$.

Lastly, we can extract the estimates of ATT.

Table 4: Results table showing unweighted, propensity score weighted, and DR estimates

	E.y1	E.y0	se.y1	se.y0	TE	se.TE	p
un	0.117	0.027	0.019	0.001	0.090	0.019	0
ps	0.117	0.045	0.019	0.006	0.072	0.020	0
dr	0.117	0.056	0.019	0.006	0.061	0.023	0

```
ps2$effects$arrest |>
  kbl(digits = 3) |>
  kable_material_opt(lightable_options="striped")
```

The first row of Table 4 shows the unadjusted analysis. The second row shows a propensity score analysis. The third row shows a doubly robust analysis, one that includes the features in the outcome regression model. The first column is the arrest rate for the heroin group. This remains unchanged regardless of the estimator. The second column is the arrest rate for the comparison group that is either unadjusted, propensity score weighted, or a doubly robust estimate. The next two columns show their standard errors. `TE` and `se.TE` show the treatment effect (ATT) and the standard error of the treatment effect. The final column shows the p-value.

Heroin users had an arrest rate of 11.7%. Based on the doubly robust analysis, respondents who were similar to the heroin users (based on age, race, sex, and marital status) had an arrest rate of 5.6%. The ATT is the difference in these two percentages, 6.1 percentage points. We can get a 95% confidence interval by taking the `TE` and adding/subtracting 1.96 times the `se.TE`, which gives us

```
att <- ps2$effects$arrest["dr","TE"]
se  <- ps2$effects$arrest["dr","se.TE"]
# estimate and 95% confidence interval
round(att + 1.96*c(0,-1,1)*se, 1)
```

```
[1] 0.1 0.0 0.1
```

These are powerful tools, but can only give results that are as good as the data. If the data have limited features, then you will be limited in how much confounding you can address. The most common failure of propensity score methods is the inability to measure important confounders in the first place. Propensity score methods only produce causal effects if strong ignorability holds. To get there often means very comprehensive data collection on the units of analysis and accounting for all of them in the propensity score model.

8 Summary

We covered a few ways of estimating treatment effects in observational studies, noting the fundamental problem of causal inference, inferring what would have happened in the absence of treatment. We covered propensity score estimation, weighting techniques to adjust for confounders, and the use of machine learning for propensity score estimation.

1. **Causal Estimation**:
    - Simpson's Paradox: after incorporating a third variable in an analysis, the conclusions can reverse
    - Observational studies aim to infer causal relationships where randomization is impractical
    - Key challenges include addressing confounding variables
    - Neyman-Rubin Causal Model: Framework underpinning causal inference methods, formalizing the challenge of unobserved counterfactuals
2. **Propensity Score Methods**:
    - Probability of treatment assignment given observed covariates
    - Balances covariate distributions between treatment and control groups to estimate causal effects
    - Use boosting to improve estimates compared to logistic regression
3. **Practical Applications**:
    - Florida murderers analysis showed the importance of addressing race of victims, an important confounding variable
    - NSDUH to study the effect of prior heroin use on future arrest
        - demonstrating propensity score with sampling weights

McCaffrey, D. M., G. Ridgeway, and A. R. Morral. 2004. "Propensity Score Estimation with Boosted Regression for Evaluating Causal Effects in Observational Studies." *Psychological Methods* 9 (4): 403–25.

Radelet, Michael L. 1981. "Racial Characteristics and the Imposition of the Death Penalty." *American Sociological Review* 46 (6): 918–27. http://www.jstor.org/stable/2095088.

Ridgeway, Greg, Stephanie A Kovalchik, Beth Ann Griffin, and Mohammed U. Kabeto. 2015. "Propensity Score Analysis with Survey Weighted Data." *Journal of Causal Inference* 3: 237–49. https://api.semanticscholar.org/CorpusID:3490612.

Rubin, Donald B. 1974. "Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies." *Journal of Educational Psychology* 66 (5): 688.