SOC 382 Causal Inference In-Class Exercise

Austin van Loon

February 28, 2019

Contents

Introduction

Here we're going to load up a couple of packages. We're all familiar with *tidyverse* by now. *MatchIt* is a useful tool for doing propensity score matching in R (we'll see how it works later, find the documentation here). *margins* is a package that will let us go from logistic regression coefficients to effect sizes (you'll talk more about this in SOC 383). Then I'm just clearing my work space, setting my working directory (if you're doing this on your machine, you'll need to change the working directory of course), and setting the random seed (so that you can reproduce my results). If you don't have these libraries already installed, you just need to run the code *install.packages("name of package")*.

```r
library(tidyverse)
library(MatchIt)
library(margins)

remove(list=ls())
setwd("C:/Users/Austin/OneDrive - Leland Stanford Junior University/Desktop")
set.seed(382)
```

Here I'm loading the data (which is publicly available and can be downloaded here) and doing some data cleaning. Here are the variables we'll be working with:

- **y**: whether the participant voted in the 2006 primary election
- **w**: this is actually a collapsed variable of several treatments, but the pertinent information is that individuals who were "treated" (w=1) were sent a letter telling them that after the election their voting record would be shared with people they know
- **male**: is equal to 1 if the participant is male, 0 if they are female (apologies for the binary)
- **age**: integer representing the age of the participant
- **p2004**: equal to 1 if the participant voted in the 2004 primary election
- **income**: self-reported mean income of the household of the participant

```r
df <- read.csv('neighbors.csv') %>%
  select(sex, yob, p2004, mean_income, outcome_voted, treatment_dum) %>%
  rename(y = outcome_voted, w=treatment_dum, income=mean_income, male=sex) %>%
  mutate(age = 2008 - yob) %>%
  select(y, w, male, p2004, age, income) %>%
  na.omit()
```

Now I'm going to define a function that will allow us to estimate the ATE ($\tau = \mathbb{E}(Y_{i(w=1)} - Y_{i(w=0)})$) assuming unconfoundedness of the treatment ($W_i \perp\!\!\!\perp Y_{i(w=1)}, Y_{i(w=0)}$), $\hat{\tau} = \mathbb{E}(Y_{i \in w=1}) - \mathbb{E}(Y_{i \in w=0})$.

```
difference_in_means <- function(dataset) {

  y1 <- dataset %>%
    filter(w == 1) %>%
    pull(y)
  y0 <- dataset %>%
    filter(w == 0) %>%
    pull(y)

  n1 <- sum(df[,"w"])
  n0 <- sum(1 - df[,"w"])

  tauhat <- mean(y1) - mean(y0)

  se_hat <- sqrt( var(y0)/(n0-1) + var(y1)/(n1-1) )
  lower_ci <- tauhat - 1.96 * se_hat
  upper_ci <- tauhat + 1.96 * se_hat

  return(c(ATE = tauhat, lower_ci = lower_ci, upper_ci = upper_ci))
}
```

Given that this is a huge, well-performed experiment, let's assume that this ATE is correct and use it as a so-called "gold standard".

```
difference_in_means(df)
```

```
##        ATE   lower_ci   upper_ci
## 0.08639696 0.07905938 0.09373454
```

This is just to show you that this formula for ATE is equivalent to doing a t-test.

```
t.test(y~w, data=df)
```

```
##
##  Welch Two Sample t-test
##
## data:  y by w
## t = -23.079, df = 27568, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.09373457 -0.07905936
## sample estimates:
## mean in group 0 mean in group 1
##        0.303903        0.390300
```

Now we're going to purposefully bias our data set! Just as background, women are more likely to vote, as are people who voted in previous primary elections. We're going to create two biased data sets:

- The first (**df.biased**) will have approximately 50% of individuals who did not vote in the last primary dropped from the treatment condition, and 50% of individuals who did vote in the last primary dropped from the control condition
- The second (**df.biased.hi**) will have 95% of men and 95% of individuals who did not vote in the last primary dropped from the treatment condition and 95% of women and 95% of individuals who did vote

in the last primary dropped from the control group

Note that both of these data sets should be biased such that we **over-estimate** the ATE, since $W_i \not\perp Y_{i(w=0)}, Y_{i(w=1)}$, i.e. treatment is correlated with one or more covariates which are correlated with Y.

```r
bias.coef <- 0.5

df.biased <- df %>%
  mutate(rand = runif(n=nrow(df))) %>%

  mutate(drop1 = ((1-p2004)*w*rand)) %>%
  filter(drop1 < (1-bias.coef)) %>%

  mutate(drop2 = (p2004*(1-w)*rand)) %>%
  filter(drop2 < (1-bias.coef)) %>%

  select(y, w, male, p2004, age, income)

bias.coef <- 0.95

df.biased.hi <- df %>%
  mutate(rand = runif(n=nrow(df))) %>%

  mutate(drop1 = ((1-p2004)*w*rand)) %>%
  filter(drop1 < (1-bias.coef)) %>%

  mutate(drop2 = (p2004*(1-w)*rand)) %>%
  filter(drop2 < (1-bias.coef)) %>%

  mutate(drop3 = (male*w*rand)) %>%
  filter(drop3 < (1-bias.coef)) %>%

  mutate(drop4 = ((1-male)*(1-w)*rand)) %>%
  filter(drop4 < (1-bias.coef)) %>%

  select(y, w, male, p2004, age, income)
```

As expected, we now over-estimate the ATE. You should note that this same result would come about if we were looking at an observed "treatment" that happened more often to women and/or individuals who voted in the last primary.

```r
difference_in_means(df)
```

```
##        ATE   lower_ci   upper_ci
## 0.08639696 0.07905938 0.09373454
```

```r
difference_in_means(df.biased)
```

```
##       ATE  lower_ci  upper_ci
## 0.1363934 0.1290029 0.1437839
```

```r
difference_in_means(df.biased.hi)
```

```
##       ATE  lower_ci  upper_ci
## 0.2178114 0.2103842 0.2252387
```

Estimating propensity scores

Now, let's relax our assumptions about the relationships between X, Y, and W. Specifically, let's assume that $W_i \perp\!\!\!\perp Y_{i(w=0)}, Y_{i(w=1)} | X_i$. That is to say, the treatment is only confounded with our outcome through observed variables. In other words, the only things biasing our treatment are variables that we are able to measure and account for. This assumption is often called **selection on observables**. From this assumption, if we had exact matches between all covariate values in the treated and control conditions we could easily calculate the ATE. Of course, the whole problem we're running into is that the two conditions **aren't** balanced on covariates! In standard econometrics, we take care of this by estimating the relationship between our covariates and our outcome, and by assuming a relationship between the effect of the treatment and the effect of these covariates (typically that they are **linearly combined**). Here, we're going to make a (in my opinion) really cool analytical move and choose a slightly more sophisticated assumption, $W_i \perp\!\!\!\perp Y_{i(w=0)}, Y_{i(w=1)} | \hat{e}_i$, where $\hat{e}_i$ is the individual's **propensity score**, or $\mathbb{P}(W_i = 1|X_i)$. This assumption is far more appropriate for observational studies than the assumption we made earlier. Now, we're going to estimate individuals' propensity scores (their probability of being treated based on the values of their covariates) with a simple logistic regression. We already know that the original data is from a randomized experiment, and so we should find that $W_i \perp\!\!\!\perp X_i$, and therefore $\hat{e}_i \perp\!\!\!\perp W_i$ but just to be sure let's regress our treatment on our covariates.

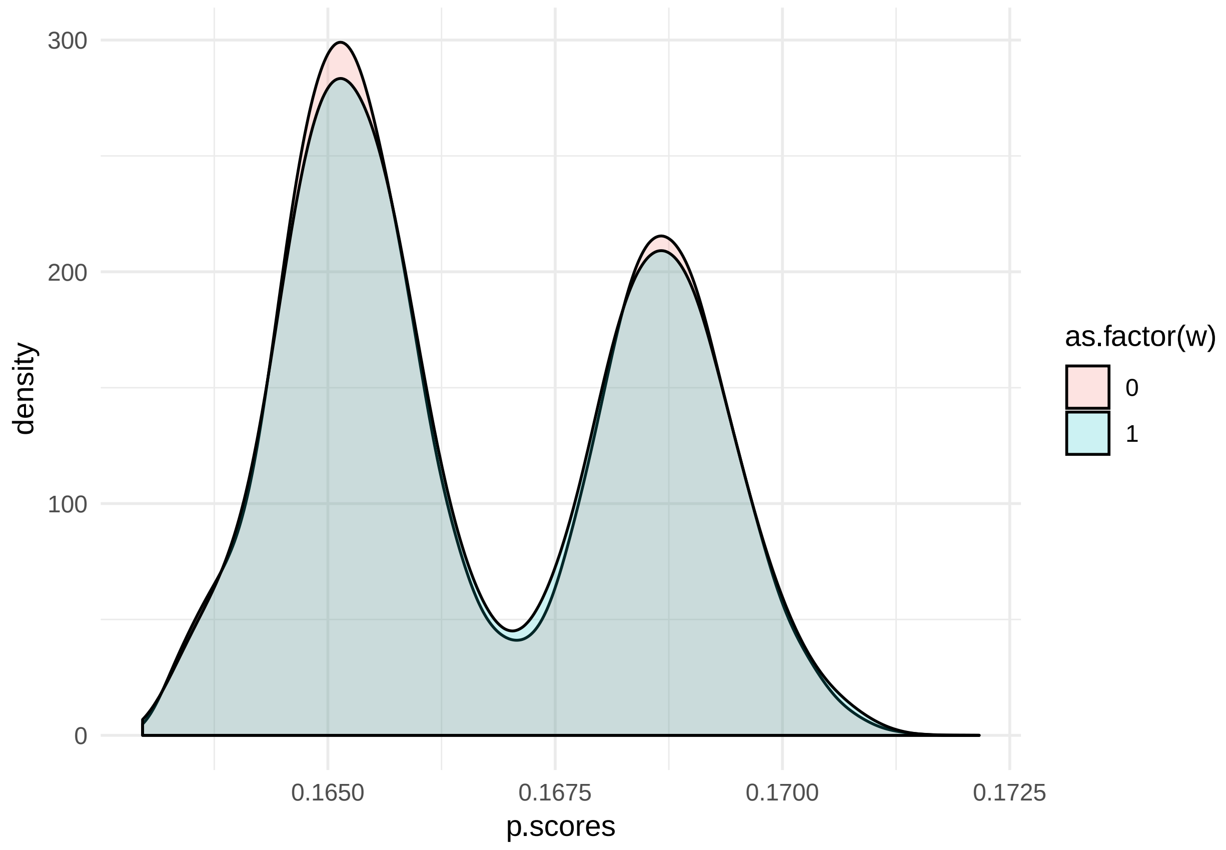
```
logit <- glm(w ~ male + p2004 + age + income, family=binomial("logit"), data=df)
summary(logit)
```

```
##
## Call:
## glm(formula = w ~ male + p2004 + age + income, family = binomial("logit"),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6147  -0.6071  -0.6016  -0.5994   1.9047
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.648e+00  4.302e-02 -38.298   <2e-16 ***
## male         3.166e-03  1.550e-02   0.204    0.838
## p2004        2.590e-02  1.576e-02   1.644    0.100
## age          3.738e-04  5.467e-04   0.684    0.494
## income       7.634e-08  3.357e-07   0.227    0.820
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 108134  on 119998  degrees of freedom
## Residual deviance: 108131  on 119994  degrees of freedom
## AIC: 108141
##
## Number of Fisher Scoring iterations: 3
```

Now we'll predict the probabilities for each participant to receive the treatment. Again, we shouldn't expect there to be any systematic differences in the propensity scores between the treated and control conditions, since this is a random experiment.

```
df$p.scores <- predict(logit, df, type='response')

ggplot(data=df, aes(p.scores, fill = as.factor(w))) + geom_density(alpha=0.2) + theme_minimal()
```



That looks great! Just as a note about the code, the last bit *+ theme_minimal* is an easy thing to experiment with! ggplot2 comes with a bunch of great themes; this one just tends to be my favorite. If you download a package called *ggthemes*, there's a whole bunch of other ones you can explore, including *theme_tufte* (yes, THAT Tufte)!

Now let's regress the treatment on the only covariate that should be related to the treatment in the slightly biased data set:

```
logit.biased <- glm(w ~ p2004, family=binomial("logit"), data=df.biased)
summary(logit.biased)
```

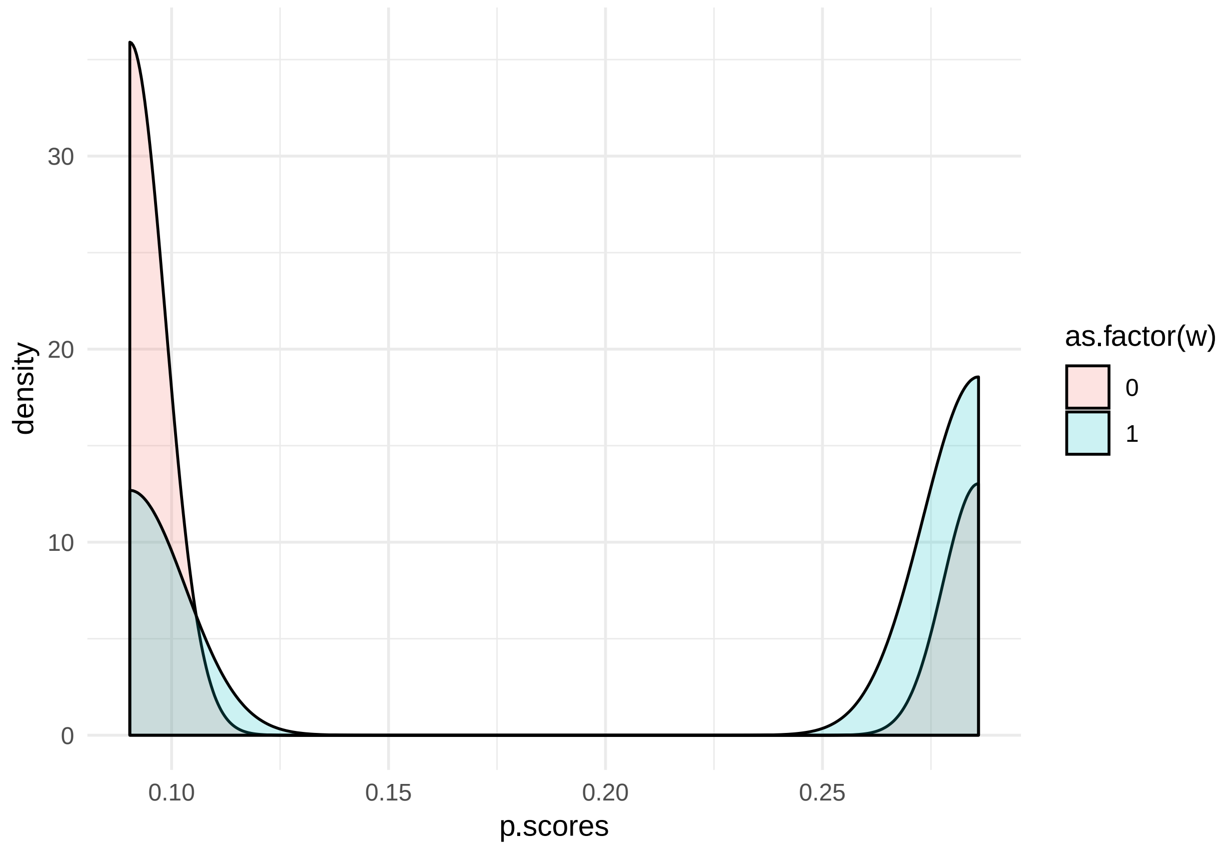
```
##
## Call:
## glm(formula = w ~ p2004, family = binomial("logit"), data = df.biased)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8208  -0.4353  -0.4353  -0.4353   2.1926
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.30906    0.01378 -167.56   <2e-16 ***
## p2004        1.39400    0.01885   73.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 79921  on 93680  degrees of freedom
## Residual deviance: 74332  on 93679  degrees of freedom
## AIC: 74336
##
## Number of Fisher Scoring iterations: 5
```

Hooray, statistics works! Now let's estimate the propensity scores and plot them:

```
df.biased$p.scores <- predict(logit.biased, df.biased, type='response')

ggplot(data=df.biased, aes(p.scores, fill = as.factor(w))) + geom_density(alpha=0.2) + theme_minimal()
```



Now we see that there is some systematic differences between the treated condition (the blue) and the control condition (the red). As someone who does experiments this might make you sad, but as someone who does causal inference on observational studies this should actually make you quite happy: the important thing you should be looking for is how much "overlap" there is between the two regions. Here, this is a pretty good amount of overlap, so even relatively simple methods should be able to recover the ATE. Now let's make our propensity score estimator for the extremely biased data set.

```
logit.biased.hi <- glm(w ~ p2004 + male, family=binomial("logit"), data=df.biased.hi)
summary(logit.biased.hi)
```

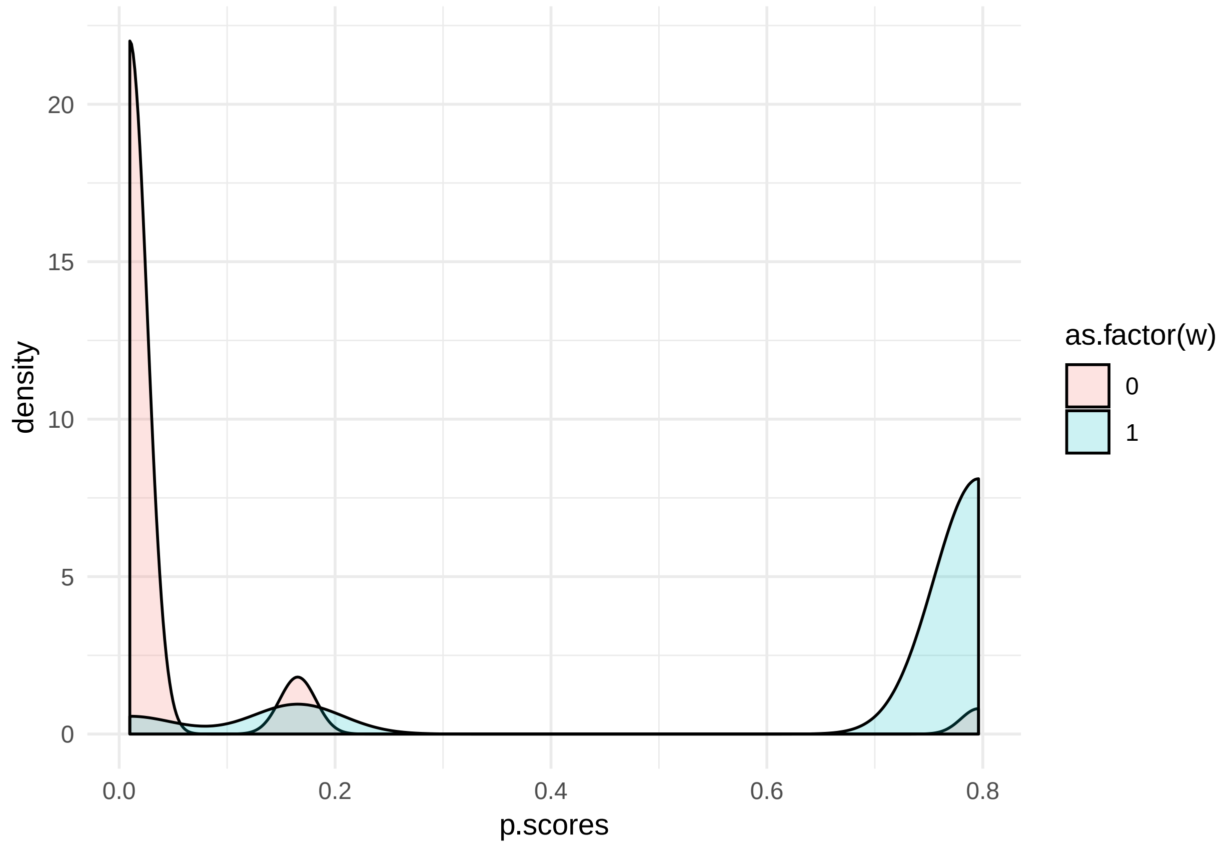
```
##
## Call:
## glm(formula = w ~ p2004 + male, family = binomial("logit"), data = df.biased.hi)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -1.7832  -0.1408  -0.1408  -0.1408   3.0393
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.59450    0.05449  -29.26   <2e-16 ***
## p2004        2.95639    0.05818   50.81   <2e-16 ***
## male        -3.01439    0.06051  -49.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 29481  on 37858  degrees of freedom
## Residual deviance: 11284  on 37856  degrees of freedom
## AIC: 11290
##
## Number of Fisher Scoring iterations: 7
```

And then we'll estimate the propensity scores for those in the treatment and control conditions and plot them.

```r
df.biased.hi$p.scores <- predict(logit.biased.hi, df.biased.hi, type='response')

ggplot(data=df.biased.hi, aes(p.scores, fill=as.factor(w))) +
  geom_density(alpha=0.2) + theme_minimal()
```



Notice here that there is much less "overlap" in the propensity scores of those who were treated and those who were not. Basically, this means that we can predict who would be in the treated group and who would be in the control group, which makes sense given how we biased the data. As a consequence, this also means that it will be harder to parse out the effect of our treatment since our treatment is so heavily biased (though even with this amount of overlap, I would still feel pretty good about my chances of recovering the ATE). Now we'll examine three ways that we can use these estimated propensity scores to regain a correct inference of the effect of the treatment.

Propensity score matching

Remember we're working off the assumption that $W_i \perp\!\!\!\perp Y_{i(w=0)}, Y_{i(w=1)} | \hat{e}_i$. Perhaps the most straight-forward way, then, to regain our ATE would be to compare individuals in the control group and the experimental group who have the same propensity scores. That is, we'll **match** individuals with the same propensity scores in the two conditions, and then the average difference between them will be our ATE. Of course, there's a major problem here: the individuals in the control and in the treatment group don't have the same distributions of propensity scores, so there's no way that we could match perfectly! So, here's what we're going to do: we're going to give up on estimating the ATE and instead estimate the ATT, or the **average treatment effect on the treated**, $ATT = \mathbb{E}(Y_{i(w=1)} - Y_{i(w=0)}) | W_i = 1$. To estimate this, we'll find some individuals *in the control condition* who have similar propensity scores to *those who were treated* and then compare their values of y. Note that this inherently privileging the information given to us by those in the experimental condition. There are reasons to prefer propensity score matching (mainly researchers like it because it is intuitive), but the two major drawbacks are that (1) some would call it **inefficient** since we're throwing away at least some of the information given to us by those individuals who were in the control group and (2) we can only estimate the ATT and can't estimate the ATE.

```
matched.biased <- matchit(logit.biased, data = df.biased,
                          method="nearest", ratio=1)

matched.biased <- match.data(matched.biased)[1:ncol(df.biased)]
nrow(matched.biased)
```

```
## [1] 28522
```

The above command is creating a new data frame (matched.bias) with only observations who were matched on their propensity score. Notice that we lose quite a few observations (the original data set had almost 120k observations). Now, you'll notice that we specify the "method" parameter as "nearest". There are many other ways to specify this, including "exact" and "optimal" (which would take a long time to run on this fairly large data set).

```
difference_in_means(df)
```

```
##        ATE   lower_ci   upper_ci
## 0.08639696 0.07905938 0.09373454
```

```
difference_in_means(df.biased)
```

```
##       ATE  lower_ci  upper_ci
## 0.1363934 0.1290029 0.1437839
```

```
difference_in_means(matched.biased)
```

```
##        ATE   lower_ci   upper_ci
## 0.09199916 0.08456403 0.09943429
```

And our estimated ATE falls within the 95% CI of our "gold standard" estimation! Let's try out the same method for the highly biased data set.

```
matched.biased.hi <- matchit(logit.biased.hi, data = df.biased.hi,
                             method="nearest", ratio=1)

matched.biased.hi <- match.data(matched.biased.hi)[1:ncol(df.biased.hi)]
nrow(matched.biased.hi)
```

```
## [1] 9962
```

```
difference_in_means(df)
```

```
##        ATE   lower_ci   upper_ci
## 0.08639696 0.07905938 0.09373454
```

```
difference_in_means(df.biased.hi)
```

```
##       ATE  lower_ci  upper_ci
## 0.2178114 0.2103842 0.2252387
```

```
difference_in_means(matched.biased.hi)
```

```
##       ATE  lower_ci  upper_ci
## 0.1666332 0.1591478 0.1741186
```

Here we don't do nearly as well. I will say that doing exact matching would probably get us a lot closer to the actual ATE, but would also probably eliminate most of our sample in the process. We could also include multiple matches per treated unit, but then we'd probably bias our estimate. Every choice you make has trade-offs.

Inverse propensity-score weighting

Under our new assumption $W_i \perp\!\!\!\perp Y_{i(w=0)}, Y_{i(w=1)} | \hat{e}_i$, we can derive that $\hat{\tau} = \mathbb{E}(\frac{W_i Y_i}{\hat{e}_i} - \frac{(1-W_i)Y_i}{1-\hat{e}_i})$. This is nice because we get to use all of our data insofar as it is informative and only insofar as it is informative. Below I create a function that calculates this for us (so easy!).

```
ipw_ate <- function(dataset) {
  y <- dataset %>%
    pull(y)

  w <- dataset %>%
    pull(w)

  p <- dataset %>%
    pull(p.scores)

  tau.hat <- mean(((w*y)/p)-(((1 - w)*y)/(1-p)))

  return(c(ATE = tau.hat))
}
```

It might be worth proving to yourself that if $W_i \perp\!\!\!\perp X_i, Y_{i(w=1)}, Y_{i(w=0)}$ (i.e. if we have perfectly random assignment), then this equation can be simplified to our original equation, $\hat{\tau} = \mathbb{E}(Y_{i \in w=1}) - \mathbb{E}(Y_{i \in w=0})$.

```
df.biased$weight <- ifelse(df.biased$w==1,
                           1/df.biased$p.scores,
                           1/(1 - df.biased$p.scores))
```

```
difference_in_means(df)
```

```
##        ATE   lower_ci   upper_ci
## 0.08639696 0.07905938 0.09373454
```

```
difference_in_means(df.biased)
```

```
##       ATE  lower_ci  upper_ci
## 0.1363934 0.1290029 0.1437839
```

```
difference_in_means(matched.biased)
```

```
##        ATE   lower_ci   upper_ci
## 0.09199916 0.08456403 0.09943429
```

```
ipw_ate(df.biased)
```

```
##        ATE
## 0.08796612
```

We're able to almost exactly recover the ATE! Awesome! But that was the "easy" test… How does it do on the heavily biased data set?

```
df.biased.hi$weight <- ifelse(df.biased.hi$w==1,
                              1/df.biased.hi$p.scores,
                              1/(1 - df.biased.hi$p.scores))

difference_in_means(df)
```

```
##        ATE   lower_ci   upper_ci
## 0.08639696 0.07905938 0.09373454
```

```
difference_in_means(df.biased.hi)
```

```
##       ATE  lower_ci  upper_ci
## 0.2178114 0.2103842 0.2252387
```

```
difference_in_means(matched.biased.hi)
```

```
##       ATE  lower_ci  upper_ci
## 0.1666332 0.1591478 0.1741186
```

```
ipw_ate(df.biased.hi)
```

```
##        ATE
## 0.07578658
```

Not nearly as well, unfortunately!

Logistic Regression

So, the standard way to approach this problem for a sociologist would be to simply "control for the effect of" the confounding variables. It's important that you think through what a linear model is doing when it "controls for something" to understand why that might not be a good idea. If you go through that reasoning, it should become pretty apparent that under some conditions this technique should do reasonably well, and in others it will go horribly wrong. So, let's see how a standard logistic regression does in this context!

```r
margins(glm(y ~ w + p2004, data=df.biased, family=binomial("logit")))
```

```
## Average marginal effects

## glm(formula = y ~ w + p2004, family = binomial("logit"), data = df.biased)

##        w  p2004
##  0.08519 0.1237
```

```r
margins(glm(y ~ w + male + p2004, data=df.biased.hi))
```

```
## Average marginal effects

## glm(formula = y ~ w + male + p2004, data = df.biased.hi)

##      w     male  p2004
##  0.101 -0.01801 0.1247
```

So overall it does pretty well! One thing I will note as that this is a relatively "easy" situation for linear models (since the confounding variables are binary and the treatment probably doesn't interact with the confounders to a significant degree). Some people who do causal inference argue that to get the ATE out of a regression you need to run all interaction terms with the treatment. Basically, if there are interactions then not including them in the model will not give you an accurate estimate of the ATE, though this still makes assumptions that many causal inference folks feel uncomfortable about. That would be implemented like this:

```r
margins(glm(y ~ w * p2004, data=df.biased, family=binomial("logit")))
```

```
## Average marginal effects

## glm(formula = y ~ w * p2004, family = binomial("logit"), data = df.biased)

##        w  p2004
##  0.08347 0.1235
```

```r
margins(glm(y ~ w * (male + p2004), data=df.biased.hi))
```

```
## Average marginal effects

## glm(formula = y ~ w * (male + p2004), data = df.biased.hi)

##       w     male  p2004
##  0.0675 -0.01776 0.1208
```

And then some folks suggest going further and using propensity weights in logistic regression with full interactions with the treatment, as shown below.

```r
margins(glm(y ~ w * p2004, data=df.biased, family=binomial("logit"), weights=p.scores))
```

```
## Average marginal effects

## glm(formula = y ~ w * p2004, family = binomial("logit"), data = df.biased,     weights = p.scores)

##        w  p2004
##  0.08347 0.1235
```

```r
margins(glm(y ~ w * (male + p2004), data=df.biased.hi, weights=p.scores))
```

```
## Average marginal effects

## glm(formula = y ~ w * (male + p2004), data = df.biased.hi, weights = p.scores)

##        w     male  p2004
##  0.04812 -0.02289 0.1257
```

So, as this exercise has hopefully shown you, NONE of these methods are perfect! However, I will say that many people in social science right now are concerned with “causation” as it is defined by the potential outcome framework, and to be taken seriously in many interdisciplinary journals you either need to show the robustness of your results to the kind of methods we lay out here or say many times over that your research is “only descriptive”. Overall, I would suggest that you always see under what analyses your result is and is not robust, as this is often as informative as the finding itself.