ARE 213 Section 9/20/2021

Joel Ferguson

9/18/2021

Covariate Overlap and CEF Functional Form in the Regression Adjustment Design

We've already seen how we need two particular restrictions on the relationship between potential outcomes and the treatment assignment process for causal identification: (conditional) independence of potential oucomes from treatment status and SUTVA (or a similar restriction). We'll now introduce an additional assumption: *covariate overlap*, which will allow us to use a particularly simple approach to causal estimation: running a regression.

First, let's review conditional independence of potential oucomes from treatment status, otherwise known as *unconfoundedness.* This assumption states

$$(Y_i(0), Y_i(1)) \perp D_i | X_i$$

.

People will often state this assumption as "treatment is as-good-as-randomly assigned, conditional on X_i. However, another way to think of it that I find useful (and I think will be particularly useful when we get to propensity score methods in a week or two) is that if you know X_i, you cannot learn anything from Y_i about D_i. This is a powerful result because it says we can compare $E[Y_i|D_i = 0, X_i = x]$ and $E[Y_i|D_i = 1, X_i = x]$ and learn the average treatment effect at x, since there cannot be any selection!

In practice, it may be difficult to find a meaningful number of observations with either treatment status at every observed value of X_i, and in fact the assumptions we've made so far don't rule out that it's **completely impossible**. In particular, this is the case if

$$\{X_i : D_i = 1\} \cap \{X_i : D_i = 0\} = \emptyset$$

To make headway, we can bump up the unconfoundedness assumption to *strong unconfoundedness* by additionally assuming *covariate overlap*:

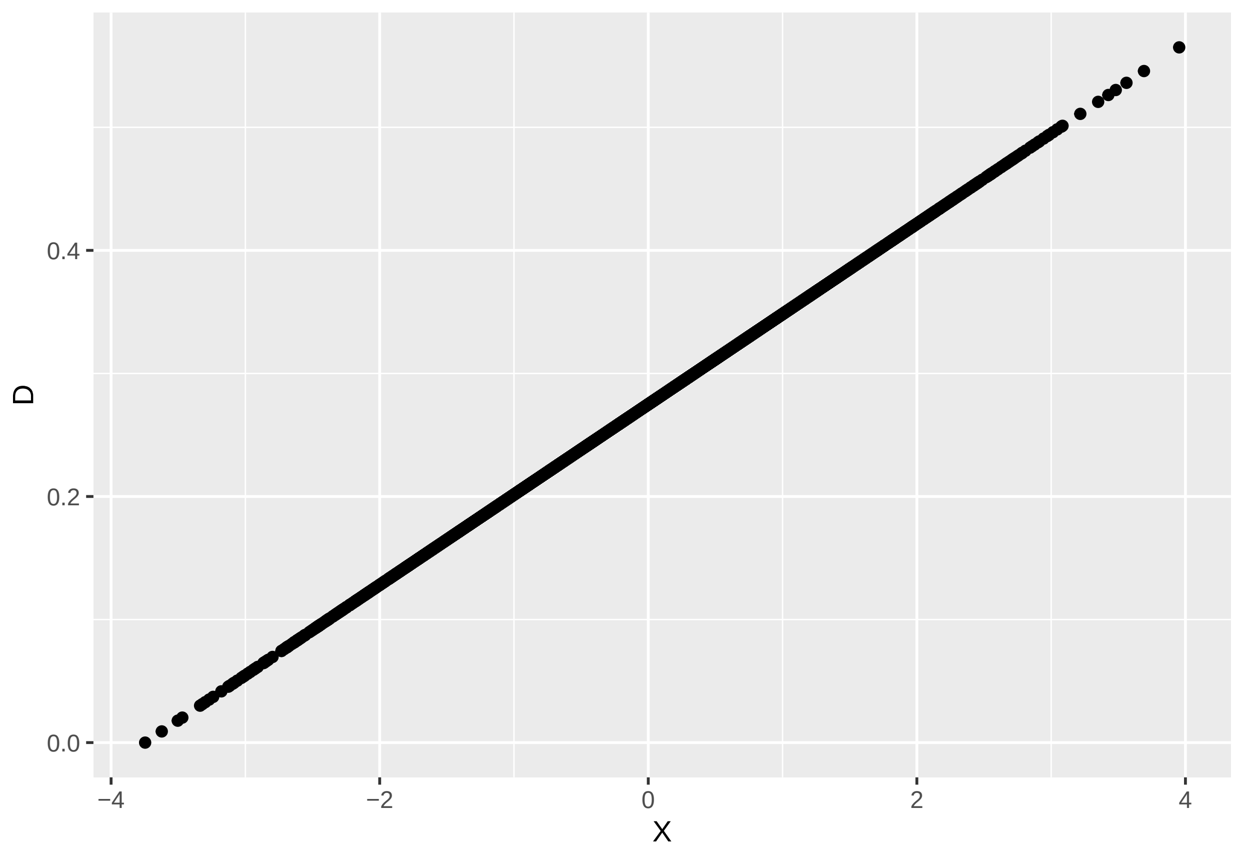
$$0 \leq P(D_i = 1|X_i) \leq 1$$

Covariate overlap enforces that we have treated and untreated observations at every point in the covariate distribution, at least in a sufficiently large sample. Equipped with strong unconfoundedness we know that we should always be able to calculate $E[Y_i(1) - Y_i(0)|X_i]$ by finding $E[Y_i|X_i, D_i]$. One tool for such a task is OLS regression. As shown in the class notes, OLS is the lowest MSE linear approximator of the CEF, and as such, we might expect it to do a good job in this case. As also shown in the class notes, how well it does depends on how close to linear $E[D_i|X_i]$ is. This is what we'll explore via simulation

```r
set.seed(92021)
N <- 10000
X_i <- rnorm(N) # One continuous covariate drawn from ~N(0,1)
# Find min and max of X
minX <- min(X_i)
maxX <- max(X_i)

# Draw untreated POs from ~N(0,10), treated POs from ~N(Y(0)+10,1)
Y_i0 <- rnorm(N,0,10)
Y_i1 <- sapply(Y_i0,function(x) rnorm(1,10+x))

print(paste("ATE:",mean(Y_i1-Y_i0)))
```

```
## [1] "ATE: 10.0179292428428"
```

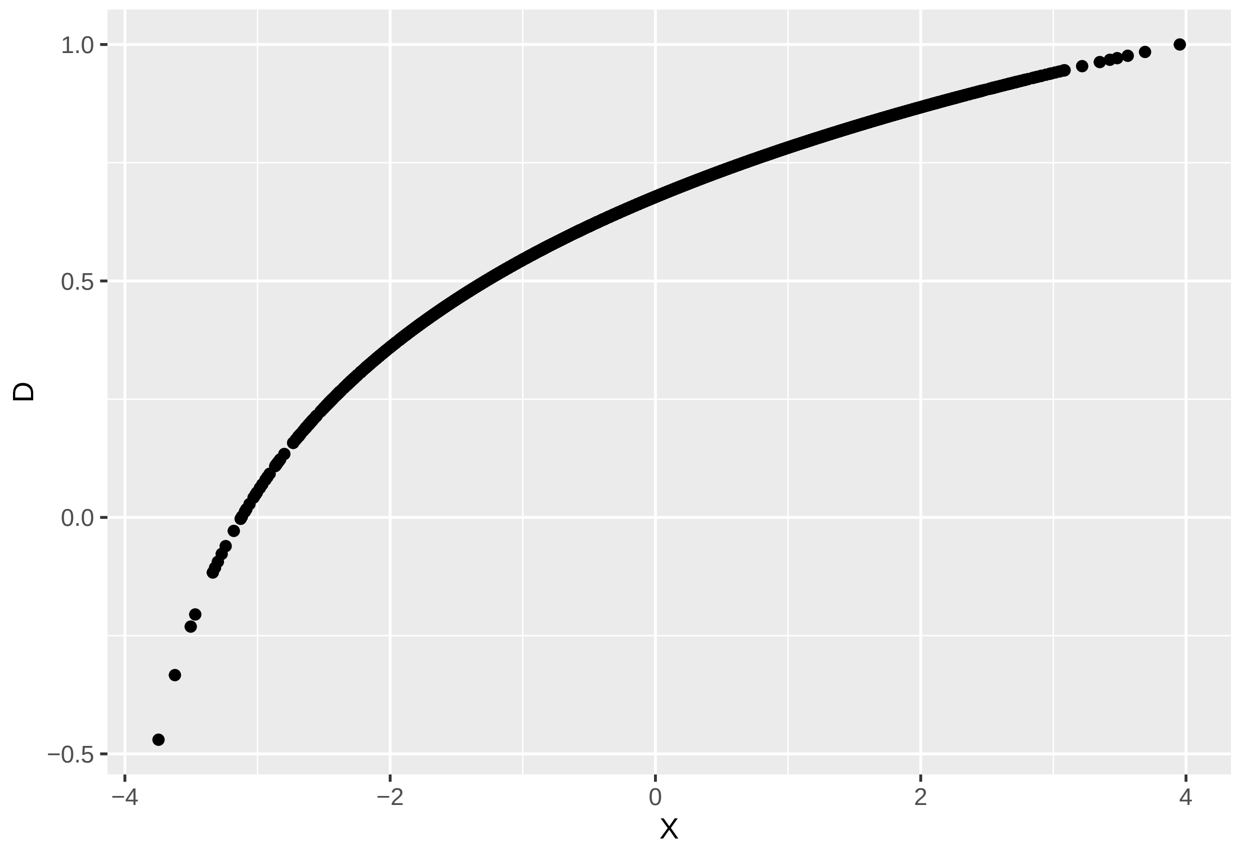
```r
# F'n to plot the prob of treatment by X
plotD <- function(Dfn){ # Dfn is a function which assigns treatment prob to X
  D <- sapply(X_i,Dfn) # Find treatment Probs
  df <-  data.frame("X"=X_i,
                    "Y0"=Y_i0,
                    "Y1"=Y_i1,
                    "D"=as.numeric(D),
                    "Y"=Y_i1*D+(1-D)*Y_i0) # Make a DF
  out <- ggplot(data=df)+
    geom_point(aes(x=X,y=D))
  return(out)
}


# F'n to compute ATE given a treatment vector
getATE <- function(D){
  df <- data.frame("X"=X_i,
                    "Y0"=Y_i0,
                    "Y1"=Y_i1,
                    "D"=D,
                    "Y"=Y_i1*D+(1-D)*Y_i0)
  ATE_lin <- lm(Y~D+X,df) # Reg Y on X and D
  return(summary(ATE_lin))
}

# First, lets try a nice linear form for E[D|X]
Dfn_lin <- function(x){
  return((x-minX)/((1+(maxX-minX)/10)*maxX-(1+(maxX-minX)/10)*minX))
}
D_lin <- sapply(X_i, function(x) rbernoulli(1,Dfn_lin(x)))
plotD(Dfn_lin)
```



```r
getATE(D_lin)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.627  -6.705   0.099   6.710  42.212
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2574     0.1163   2.214   0.0269 *
## DTRUE         9.6842     0.2266  42.735   <2e-16 ***
## X             0.1500     0.1000   1.500   0.1336
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.917 on 9997 degrees of freedom
## Multiple R-squared:  0.159,  Adjusted R-squared:  0.1589
## F-statistic: 945.2 on 2 and 9997 DF,  p-value: < 2.2e-16
```

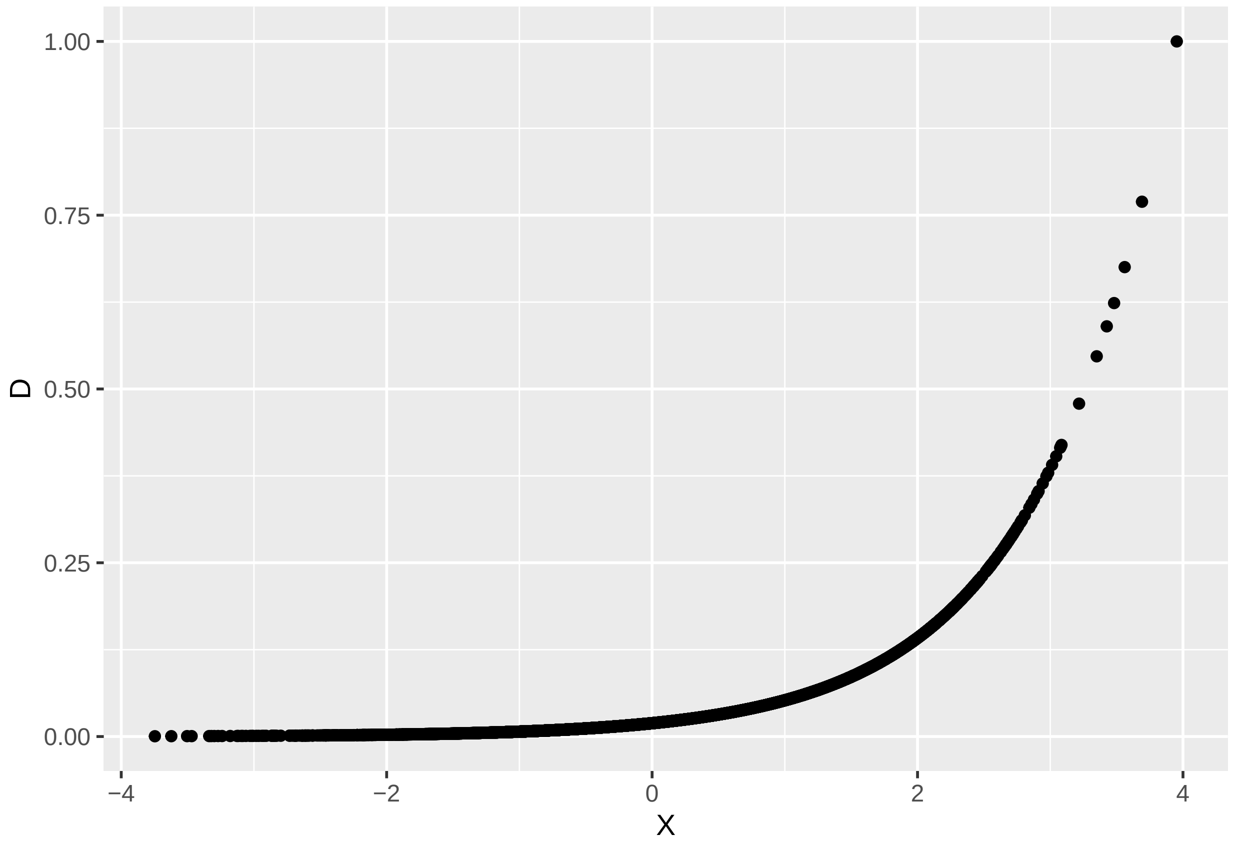
```r
# Now let's try something a bit less linear, like log
Dfn_log <- function(x){
  return(log(x-(minX*1.1))/log(maxX-(minX*1.1)))
}
D_log <- sapply(X_i, function(x) rbernoulli(1,Dfn_log(x)))
plotD(Dfn_log)
```



```
getATE(D_log)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.397  -6.749   0.088   6.713  42.774
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.4478     0.1743   2.569   0.0102 *
## DTRUE         9.6009     0.2168  44.276   <2e-16 ***
## X             0.1893     0.1024   1.848   0.0646 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.935 on 9997 degrees of freedom
## Multiple R-squared:  0.1765, Adjusted R-squared:  0.1763
## F-statistic:  1071 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```r
# Even more non-linear, like an exponential
Dfn_exp <- function(x){
  return(exp(x)/exp(maxX))
}
D_exp <- sapply(X_i, function(x) rbernoulli(1,Dfn_exp(x)))
plotD(Dfn_exp)
```

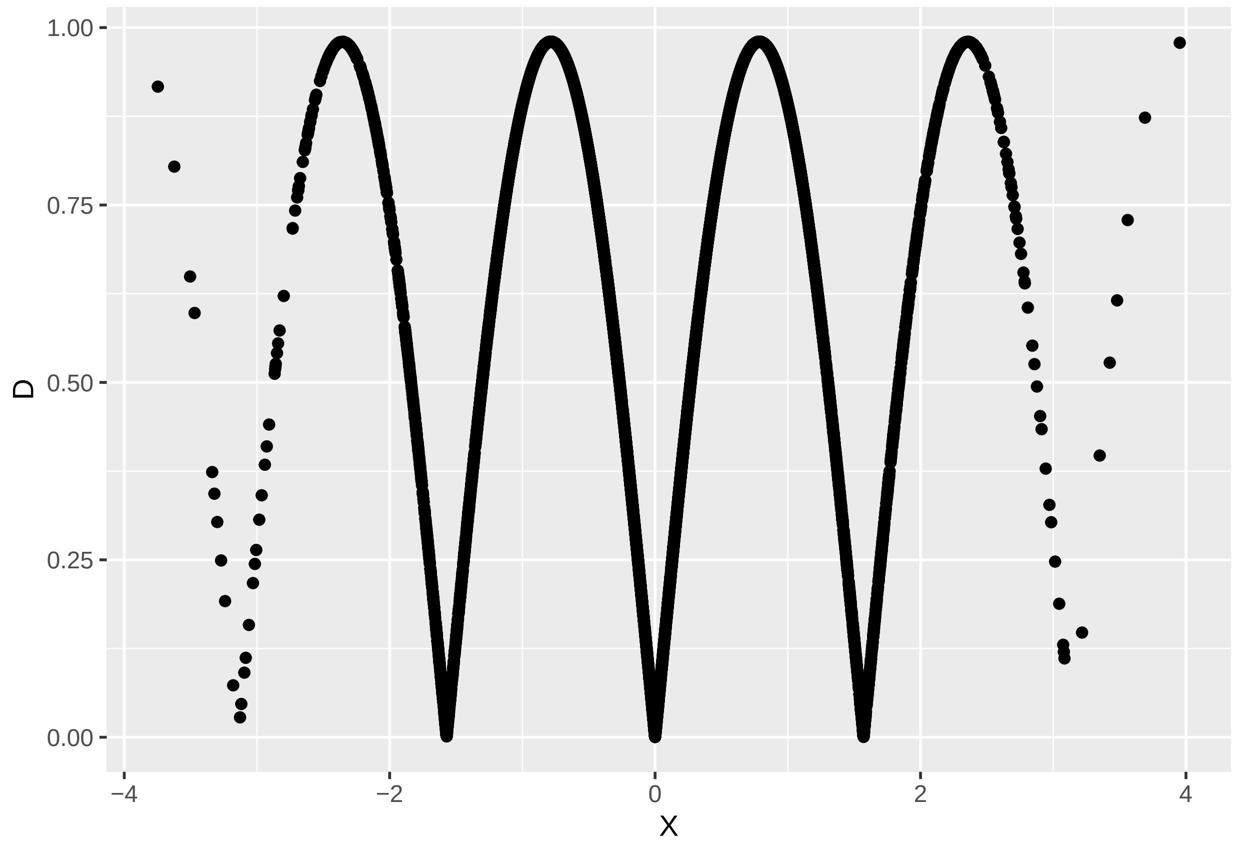


```r
getATE(D_exp)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.523  -6.727   0.104   6.754  43.013
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1611     0.1008   1.599    0.110
## DTRUE        10.3255     0.5757  17.934   <2e-16 ***
## X             0.1242     0.1004   1.237    0.216
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 9.911 on 9997 degrees of freedom
## Multiple R-squared:  0.03307,    Adjusted R-squared:  0.03288
## F-statistic:   171 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```
# Now super non-linear: abs(sine)
Dfn_sin <- function(x){
  return(abs(.98*sin(x*2)))
}
D_sin <- sapply(X_i, function(x) rbernoulli(1,Dfn_sin(x)))
plotD(Dfn_sin)
```



```
getATE(D_sin)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.527  -6.719   0.088   6.693  41.960
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.15532    0.16212   0.958    0.338
## DTRUE        10.03573    0.20516  48.916   <2e-16 ***
## X             0.14590    0.09907   1.473    0.141
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.935 on 9997 degrees of freedom
## Multiple R-squared:  0.1933, Adjusted R-squared:  0.1932
## F-statistic:  1198 on 2 and 9997 DF,  p-value: < 2.2e-16
```

You can see that the more non-linear forms of $E[D_i|X_i]$ show larger deviations of estimates from the true ATE of 10 but in general, all of the estimates are pretty close. This was in part due to the linearity of $E[Y_i(0)|X_i]$ and $E[Y_i(1)|X_i]$. Even if we have few obervations with a particular treatment status in parts of the covariate distribution, we can use information from other parts along with the imposed linearity to form nice counterfactuals. This starts to break down if we have non-linear CEFs for the potential outcomes

```r
set.seed(92021)
N <- 10000
X_i <- rnorm(N) # One continuous covariate drawn from ~N(0,1)
# Find min and max of X
minX <- min(X_i)
maxX <- max(X_i)

# Draw untreated PO errors from ~N(0,1)
e_i0 <- rnorm(N)
e_i1 <- rnorm(N)

# E[Y(1)|X] is exponential in X, E[Y(0)|X] is negative of that
Y_i0 <- -1*exp(X_i)+e_i0
Y_i1 <- exp(X_i)+e_i1

print(paste("ATE:",mean(Y_i1-Y_i0)))
```

```
## [1] "ATE: 3.29241998651511"
```

```r
# First, lets try a nice linear form for E[D|X]
D_lin <- sapply(X_i, function(x) rbernoulli(1,Dfn_lin(x)))
getATE(D_lin)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.434  -0.945   0.112   0.996  50.970
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.55238    0.02618  -59.29   <2e-16 ***
## DTRUE        3.73300    0.05104   73.14   <2e-16 ***
## X           -0.66108    0.02253  -29.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.234 on 9997 degrees of freedom
## Multiple R-squared:  0.3629, Adjusted R-squared:  0.3628
## F-statistic:  2848 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```r
# Now let's try something a bit less linear, like log
D_log <- sapply(X_i, function(x) rbernoulli(1,Dfn_log(x)))
getATE(D_log)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.038  -1.083  -0.149   0.898  45.134
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.74934    0.03779  -19.83   <2e-16 ***
## DTRUE        2.52662    0.04701   53.75   <2e-16 ***
## X            0.91703    0.02220   41.31   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.154 on 9997 degrees of freedom
## Multiple R-squared:  0.3794, Adjusted R-squared:  0.3792
## F-statistic:  3055 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```r
# Even more non-linear, like an exponential
D_exp <- sapply(X_i, function(x) rbernoulli(1,Dfn_exp(x)))
getATE(D_exp)
```

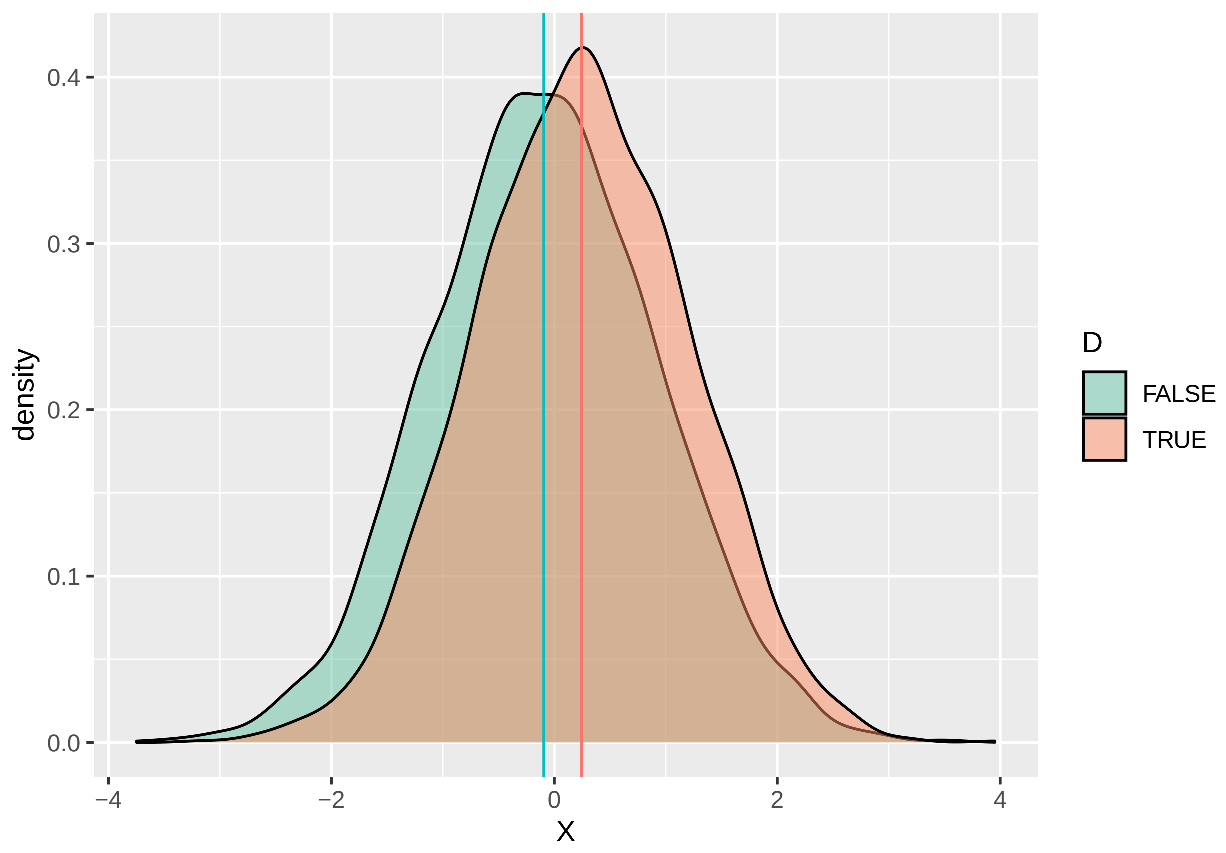
```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.759  -0.739   0.128   0.904  49.690
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.57988    0.01943  -81.30   <2e-16 ***
## DTRUE        7.46139    0.11102   67.21   <2e-16 ***
## X           -1.27344    0.01937  -65.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.911 on 9997 degrees of freedom
## Multiple R-squared:  0.4288, Adjusted R-squared:  0.4287
## F-statistic:  3752 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```r
# Now super non-linear: abs(sine)
D_sin <- sapply(X_i, function(x) rbernoulli(1,Dfn_sin(x)))
getATE(D_sin)
```

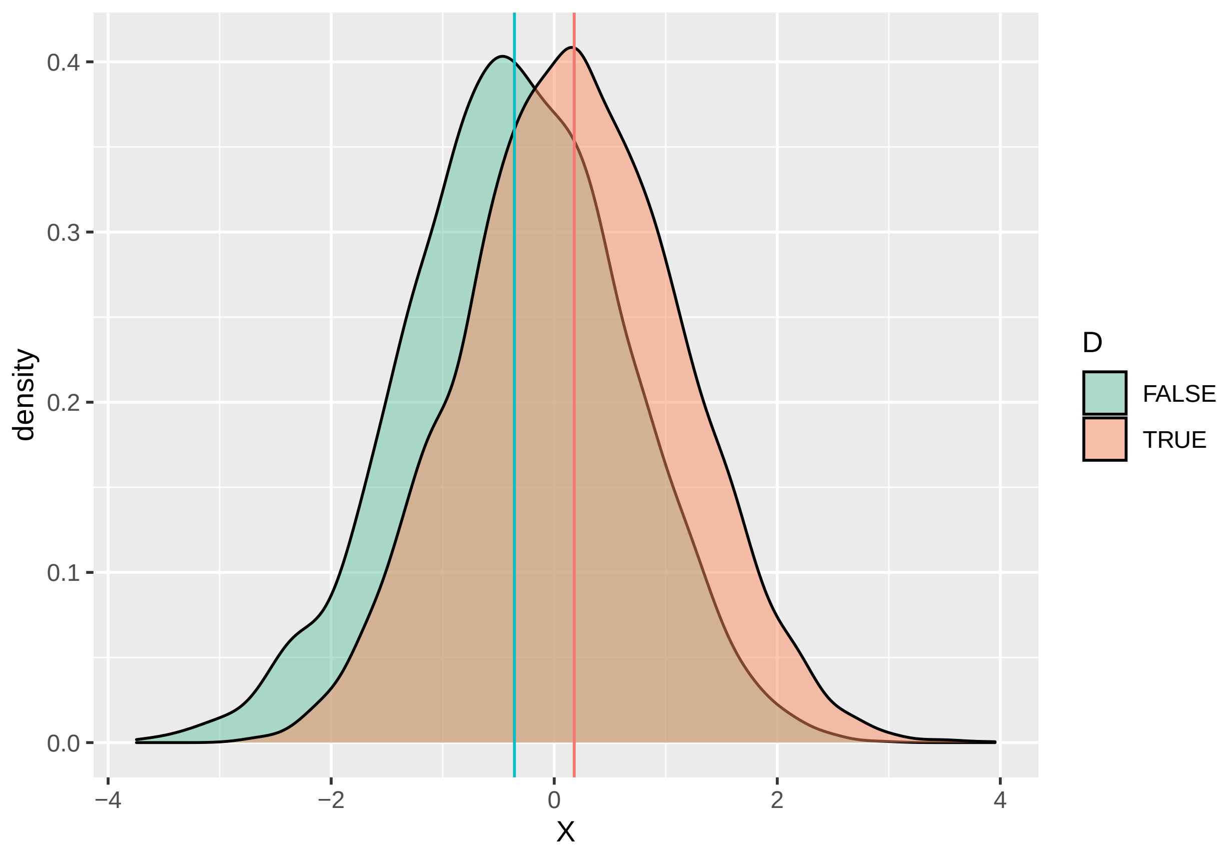
```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.550  -1.073  -0.047   1.006  47.138
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.61803    0.03784  -42.76   <2e-16 ***
## DTRUE        3.28172    0.04789   68.53   <2e-16 ***
## X            0.43885    0.02312   18.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.319 on 9997 degrees of freedom
## Multiple R-squared:  0.3365, Adjusted R-squared:  0.3363
## F-statistic:  2535 on 2 and 9997 DF,  p-value: < 2.2e-16
```

Here we see some real movement in the estimated ATEs. But interestingly, our highly non-linear sine CEF works quite well, even better than the linear CEF. What's going on here? The answer lies in distributions of X_i by treatment status.

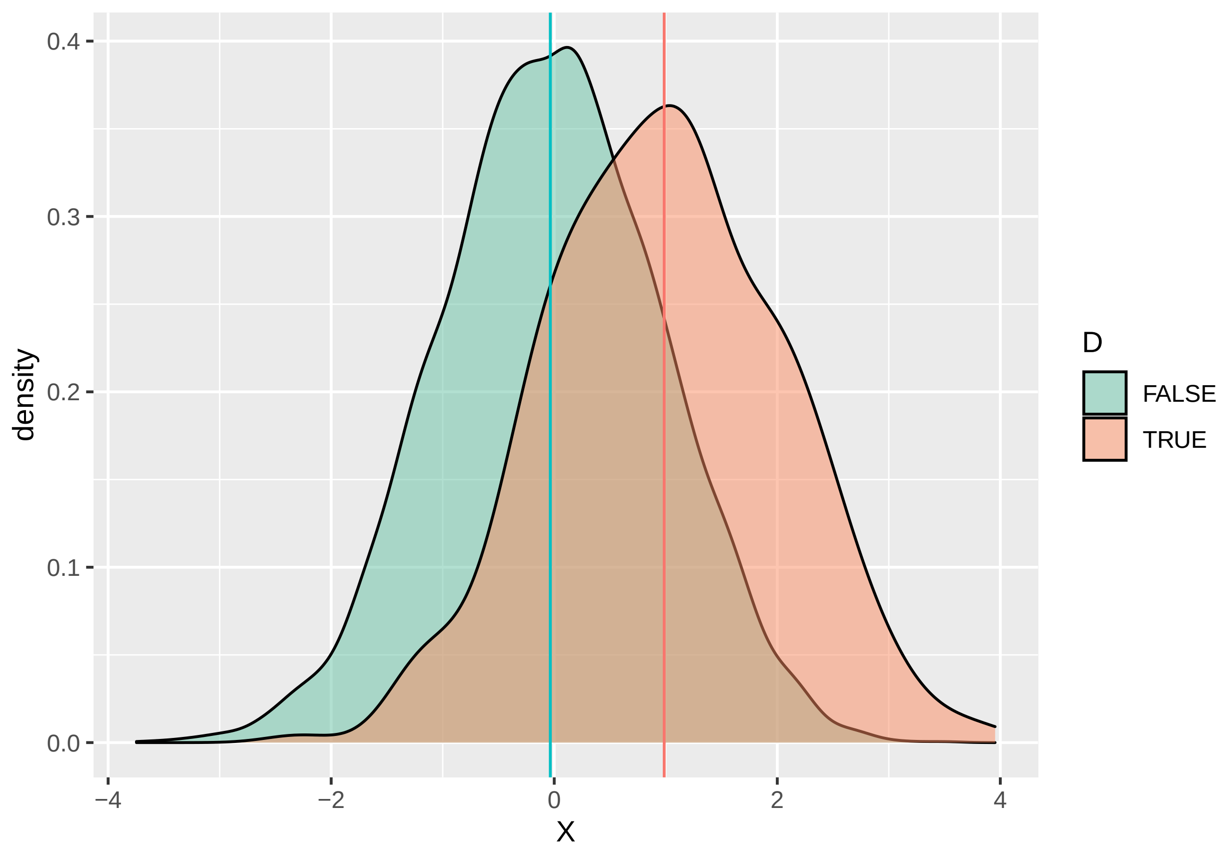
```r
# Function to plot covariate distributions by treatment status
plot_covar_dists <- function(D){
  df <-  data.frame("X"=X_i,
                    "Y0"=Y_i0,
                    "Y1"=Y_i1,
                    "D"=D,
                    "Y"=Y_i1*D+(1-D)*Y_i0)
  pal <- brewer.pal(3,"Set2")
  out <- ggplot(data=df)+
    geom_density(aes(x=X,group=D,fill=D),alpha=0.5)+
    scale_fill_brewer(palette = "Set2")+
    geom_vline(aes(xintercept=mean(X_i[D==1]),color=pal[1]),
               show.legend = F)+
    geom_vline(aes(xintercept=mean(X_i[D==0]),color=pal[2]),
               show.legend = F)
  return(out)
}

plot_covar_dists(D_lin)
```



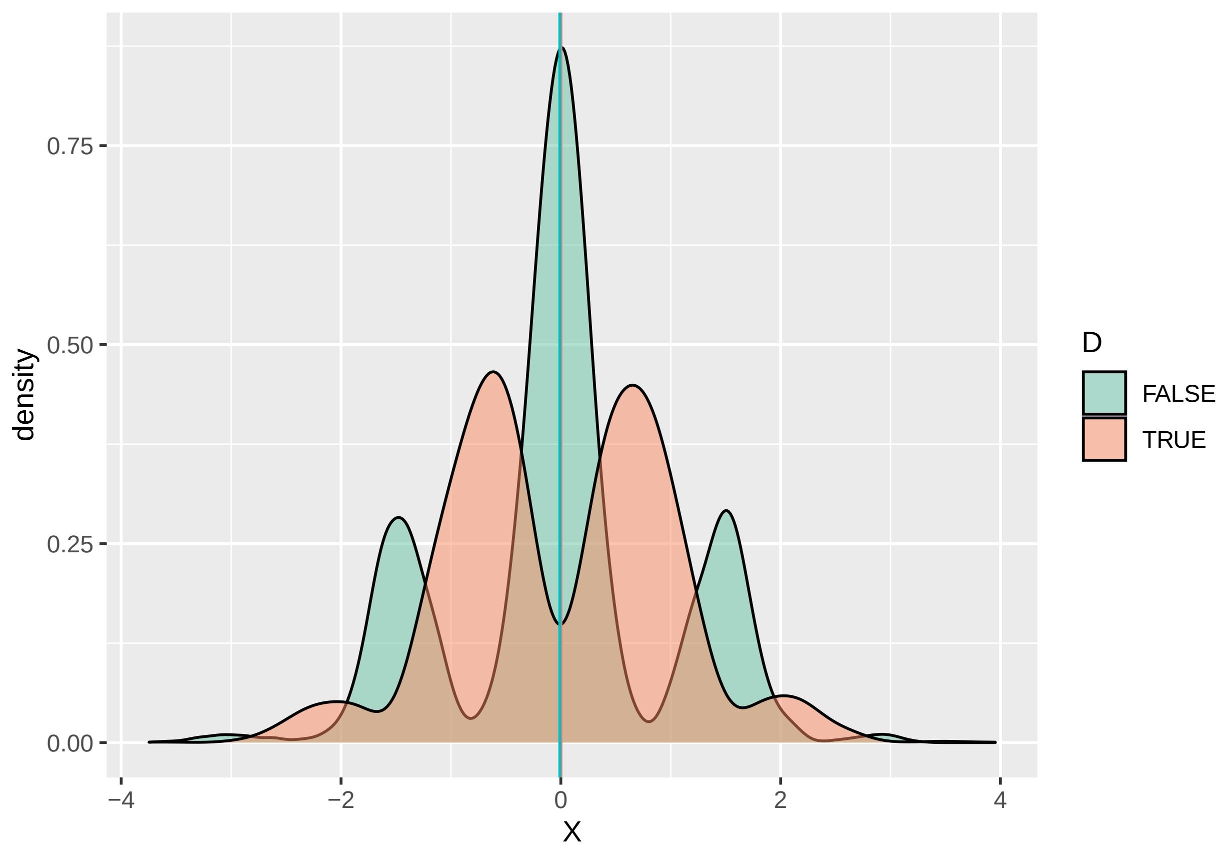
```
plot_covar_dists(D_log)
```



```
plot_covar_dists(D_exp)
```



```r
plot_covar_dists(D_sin)
```



Even though the sine CEF is highly non-linear, its distributions of $X_i|D_i$ have very close means. This is why linear regression performs well in this case. Unlike the other CEFs, for which the probability of receiving treatment is particularly higher or lower on the end of the X_i distribution with high treatment effects, the parts of the X_i distribution for which the probability of receiving treatment is high under the sine CEF are not particularly on either side. Note that where in the distribution of X_i treatment effects are high matters, just because the means are close does not guarantee the estimate of the ATE will be good. If treatment effects were high at the peaks of the sine, we'd over estimate the ATE despite the close means.