Covariate Overlap and CEF Functional Form in the Regression Adjustment Design

Joel Ferguson

9/20/2021

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 $Y_i(1-D_i)$ about how D_i was assigned. 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 that is not accounted for by X_i!

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*:

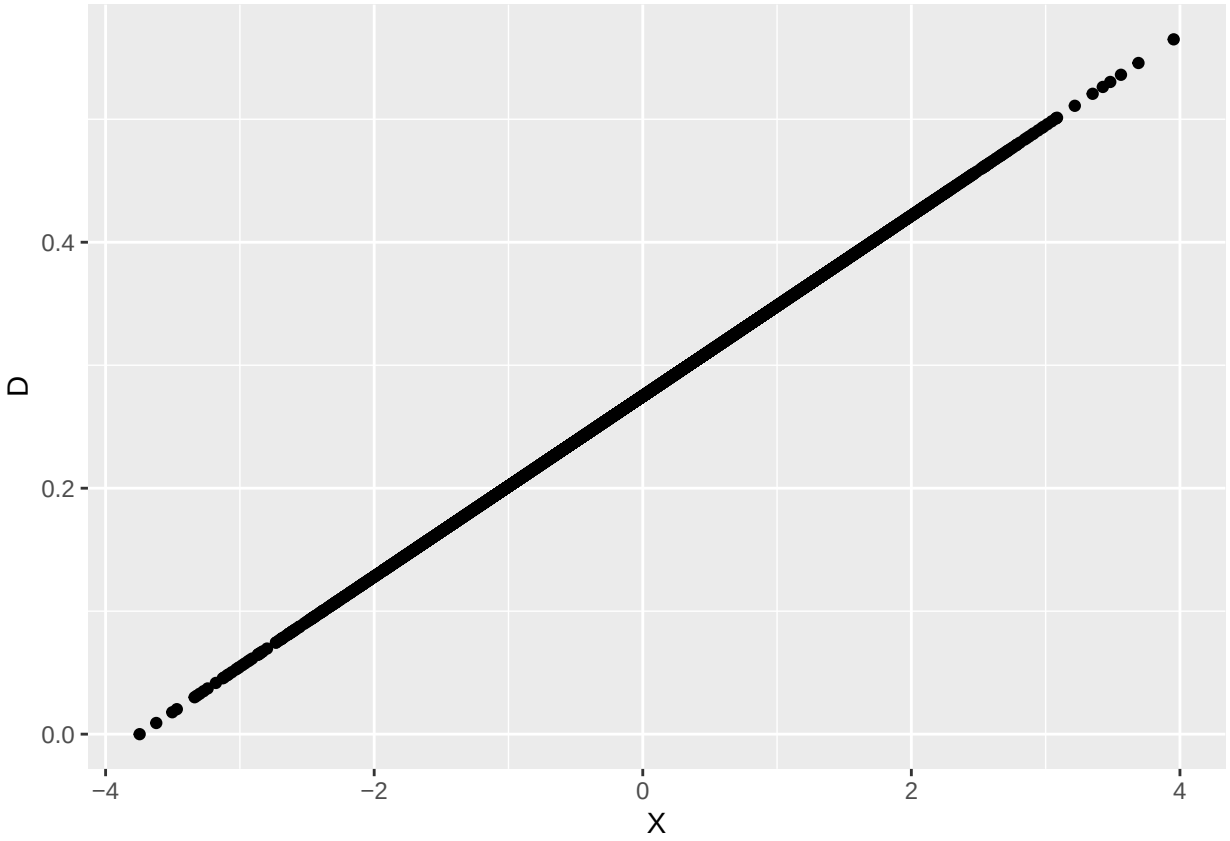
$$0 < P(D_i = 1|X_i) < 1$$

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 when we have a simple additive homogenous treatment effect 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 <- Y_i0+10
  #sapply(Y_i0,function(x) rnorm(1,10+x))

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

```
## [1] "ATE: 10"
```

```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)
```

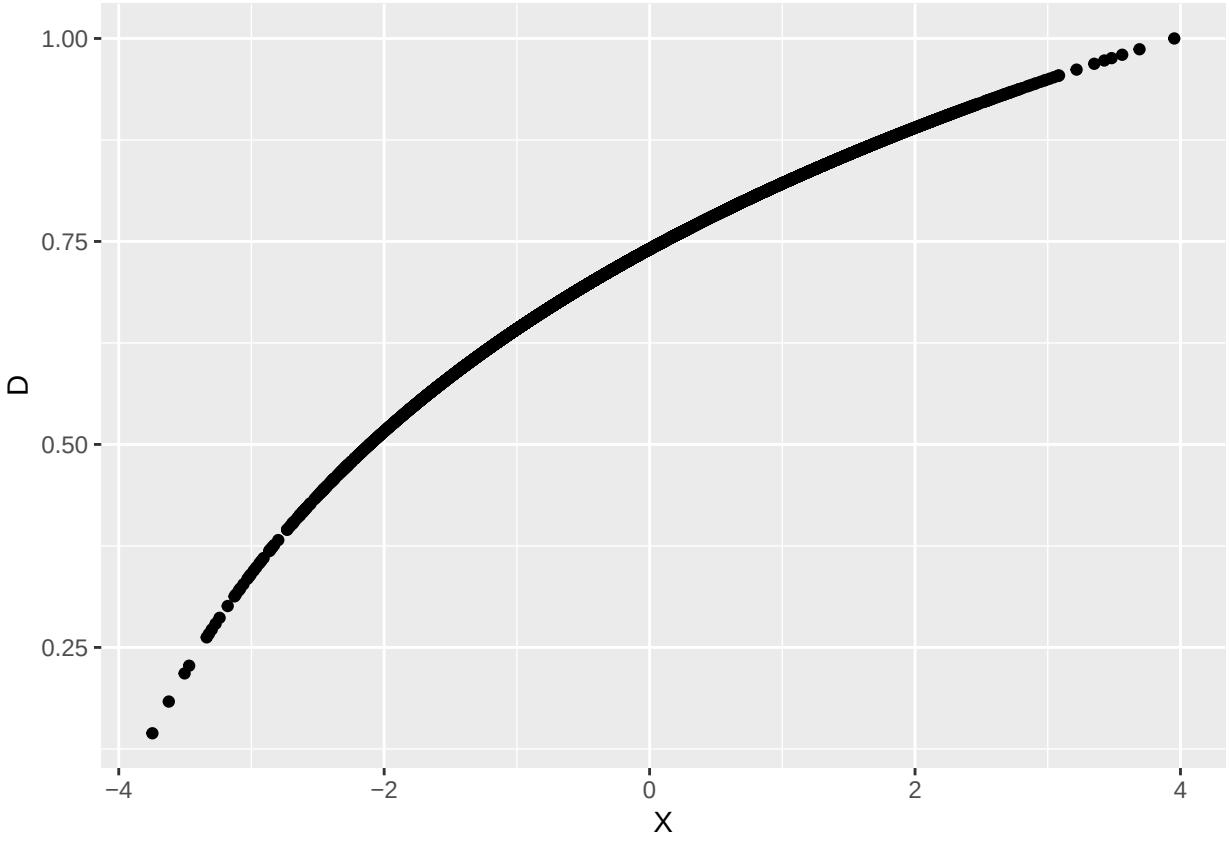



```
getATE(D_lin)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.568  -6.708   0.109   6.746  42.986
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2005     0.1175   1.706    0.088 .
## DTRUE         9.8895     0.2231  44.321   <2e-16 ***
## X             0.1410     0.1002   1.407    0.160
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.906 on 9997 degrees of freedom
## Multiple R-squared:  0.1699, Adjusted R-squared:  0.1697
## F-statistic:  1023 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```
# Now let's try something a bit less linear, like log
Dfn_log <- function(x){
  return(log(x-(minX*1.1)+1)/log(maxX-(minX*1.1)+1))
```

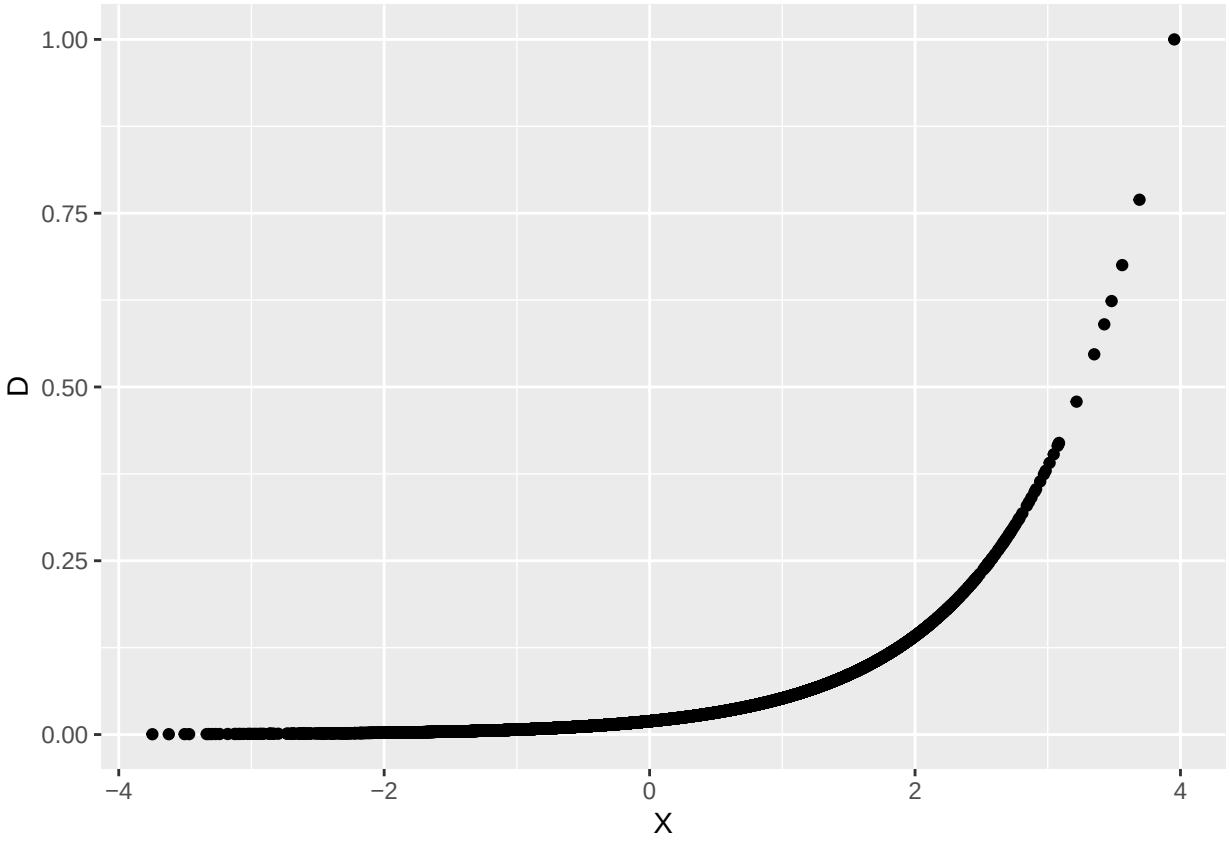
```
}
D_log <- sapply(X_i, function(x) rbernoulli(1,Dfn_log(x)))
plotD(Dfn_log)
```




```
getATE(D_log)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.497  -6.721   0.087   6.740  42.911
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2788     0.1933   1.443    0.149
## DTRUE         9.8497     0.2275  43.295   <2e-16 ***
## X             0.1460     0.1008   1.448    0.148
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.906 on 9997 degrees of freedom
## Multiple R-squared:  0.1654, Adjusted R-squared:  0.1652
## F-statistic: 990.5 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)
```

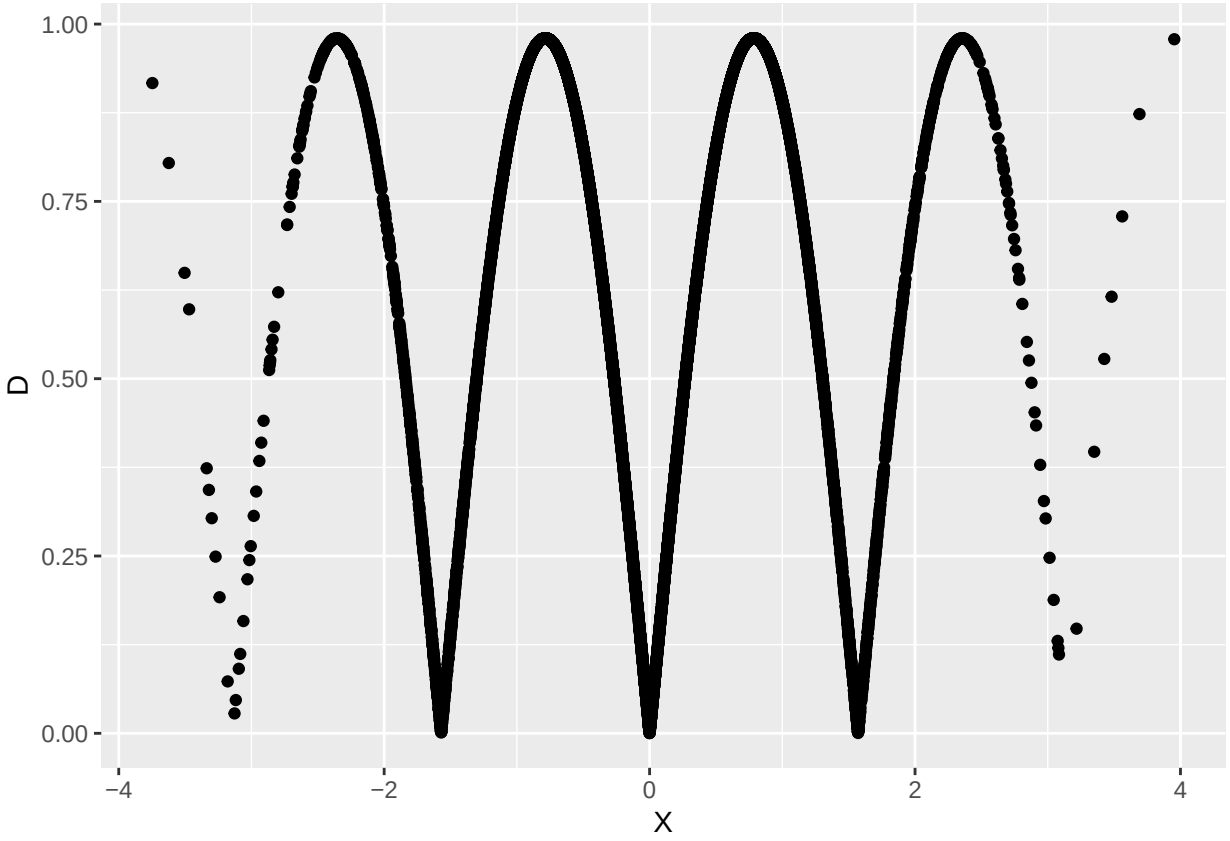



```r
getATE(D_exp)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.561  -6.711   0.113   6.732  43.004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1901     0.1006   1.890   0.0588 .
## DTRUE         9.2853     0.5969  15.556   <2e-16 ***
## X             0.1517     0.1000   1.516   0.1296
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 9.906 on 9997 degrees of freedom
## Multiple R-squared:  0.02519,    Adjusted R-squared:  0.02499
## F-statistic: 129.2 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
## -38.438  -6.713   0.115   6.748  42.952
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.07251    0.16053   0.452    0.651
## DTRUE       10.15616    0.20402  49.781   <2e-16 ***
```

```
## X            0.13358    0.09878   1.352    0.176
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.906 on 9997 degrees of freedom
## Multiple R-squared:  0.1987, Adjusted R-squared:  0.1985
## F-statistic:  1239 on 2 and 9997 DF,  p-value: < 2.2e-16
```

You can see that the non-linear forms of $E[D_i|X_i]$ tend to show larger deviations of estimates from the true ATE of 10 (although sine did pretty well) but in general, all of the estimates are pretty close. This is 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 and treatment effect heterogeneity. In the next example, both potential outcome CEFs are exponential in X_i, and $E[Y_i(0)|X_i] = -E[Y_i(1)|X_i]$, leading to larger treatment effects at higher values of X_i.

```
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: 3.29137271279047"
```

```
# 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
## -36.777  -0.697   0.419   0.655  51.841
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.52761    0.02458  -62.15   <2e-16 ***
## DTRUE        3.71939    0.04636   80.22   <2e-16 ***
```

```
## X           -0.48502    0.02090  -23.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.068 on 9997 degrees of freedom
## Multiple R-squared:  0.3955, Adjusted R-squared:  0.3954
## F-statistic:  3271 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```
# 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
## -17.778  -0.653  -0.397   0.391  46.365
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.80688    0.03695  -21.84   <2e-16 ***
## DTRUE        2.50368    0.04327   57.86   <2e-16 ***
## X            1.02534    0.01900   53.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.866 on 9997 degrees of freedom
## Multiple R-squared:  0.4401, Adjusted R-squared:   0.44
## F-statistic:  3929 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```
# 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
## -33.474  -0.219   0.342   0.594  51.947
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.60961    0.01630  -98.75   <2e-16 ***
## DTRUE        7.14538    0.09242   77.31   <2e-16 ***
## X           -1.35766    0.01624  -83.58   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.603 on 9997 degrees of freedom
```

```
## Multiple R-squared:  0.5238, Adjusted R-squared:  0.5237
## F-statistic:  5498 on 2 and 9997 DF,  p-value: < 2.2e-16
```

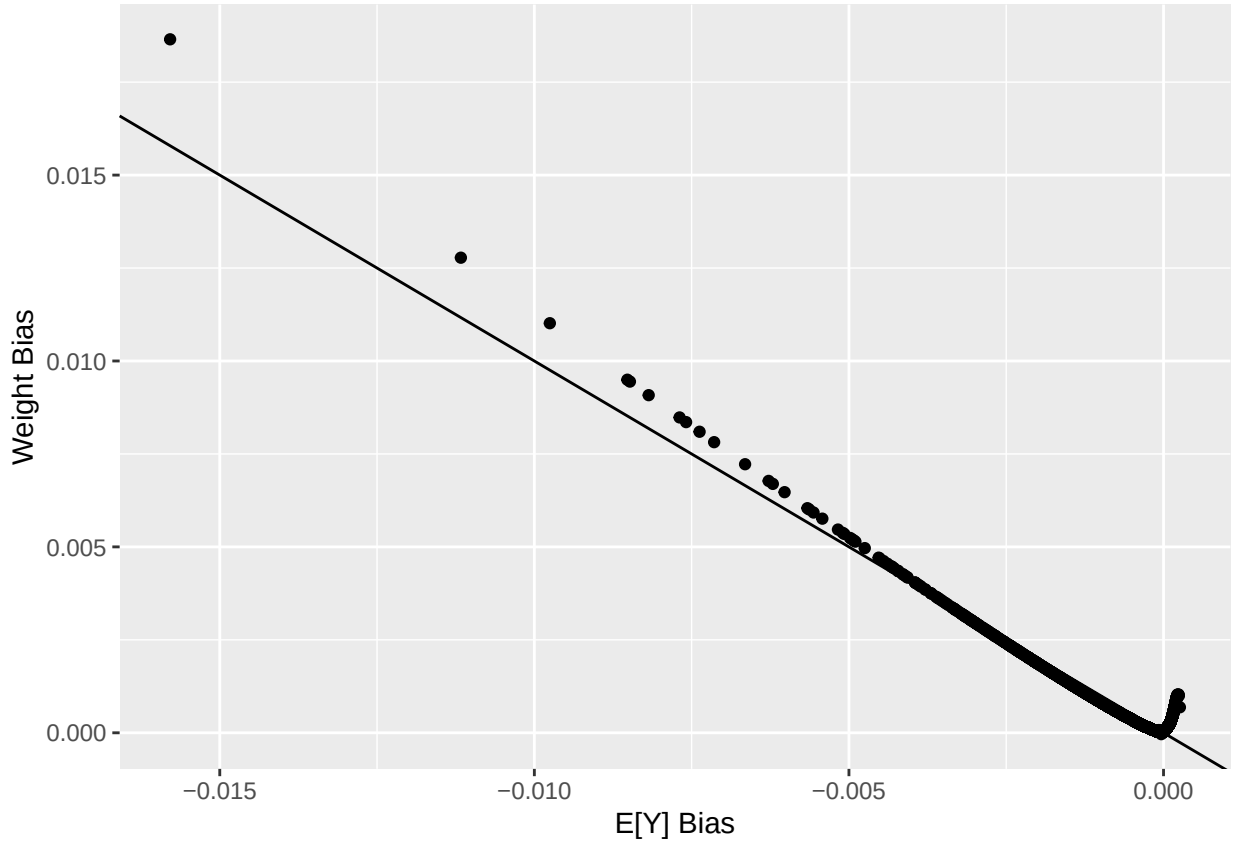
```
# Now super non-linear: abs(sine)
D_sin <- sapply(X_i, function(x) rbernoulli(1,Dfn_sin(x)))
getATE(D_sin)
```

```
##
## Call:
## lm(formula = Y ~ D + X, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.948  -0.855  -0.107   0.723  48.872
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.63194    0.03390  -48.13   <2e-16 ***
## DTRUE        3.28530    0.04320   76.04   <2e-16 ***
## X            0.40210    0.02095   19.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.101 on 9997 degrees of freedom
## Multiple R-squared:  0.3811, Adjusted R-squared:  0.381
## F-statistic:  3078 on 2 and 9997 DF,  p-value: < 2.2e-16
```

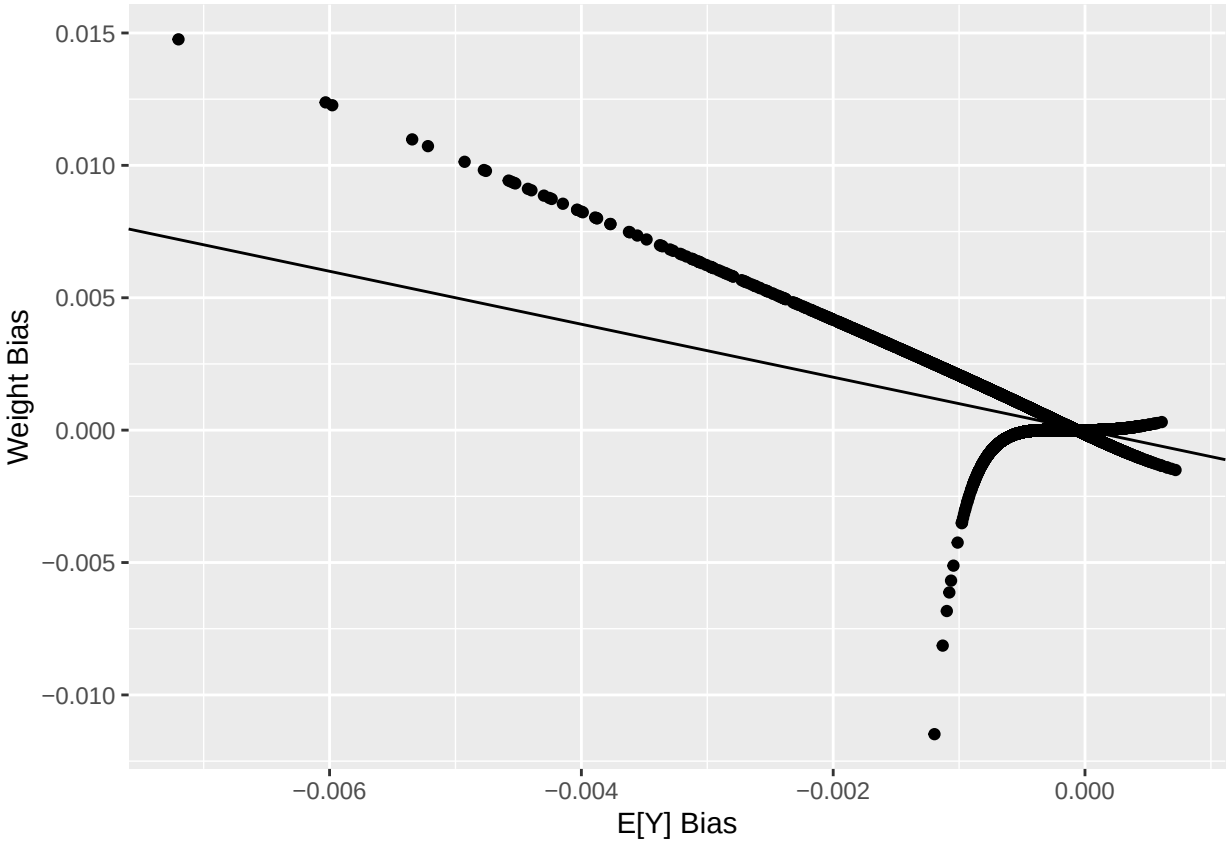
Here we see some real movement in the estimated ATEs. But interestingly, our highly non-linear sine CEF works quite well, even better than the linear CEF. What's going on here? The answer lies in how much error there is to the CEFs $E[Y_i|X_i]$ and $E[D_i|X_i]$ from using a linear approximator and, roughly, how steep the linear approximation to $E[D_i|X_i]$ is.

```
# Function to plot E[Y|D] and OLS approx
plot_EY <- function(D,Dfn){
  ED <- sapply(X_i, Dfn)
  df <-  data.frame("X"=X_i,
                    "Y0"=Y_i0,
                    "Y1"=Y_i1,
                    "D"=D,
                   "ED"=ED,
                    "Y"=Y_i1*D+(1-D)*Y_i0)

  # Run the linear reg Y on X
  regY <- lm(Y~X,df)
  df$Yfwl <- regY$residuals
  df$Y_hat <- predict(regY)


  # Run linear reg D on X
  regD <- lm(D~X,df)
  df$Dfwl <- regD$residuals
  df$bias <- 2*(df$Yfwl-abs(df$Y))/nrow(df)
  df$w <- df$Yfwl*(df$Dfwl*nrow(df)-2*sum(df$Dfwl^2))/(nrow(df)*sum(df$Dfwl^2))
```

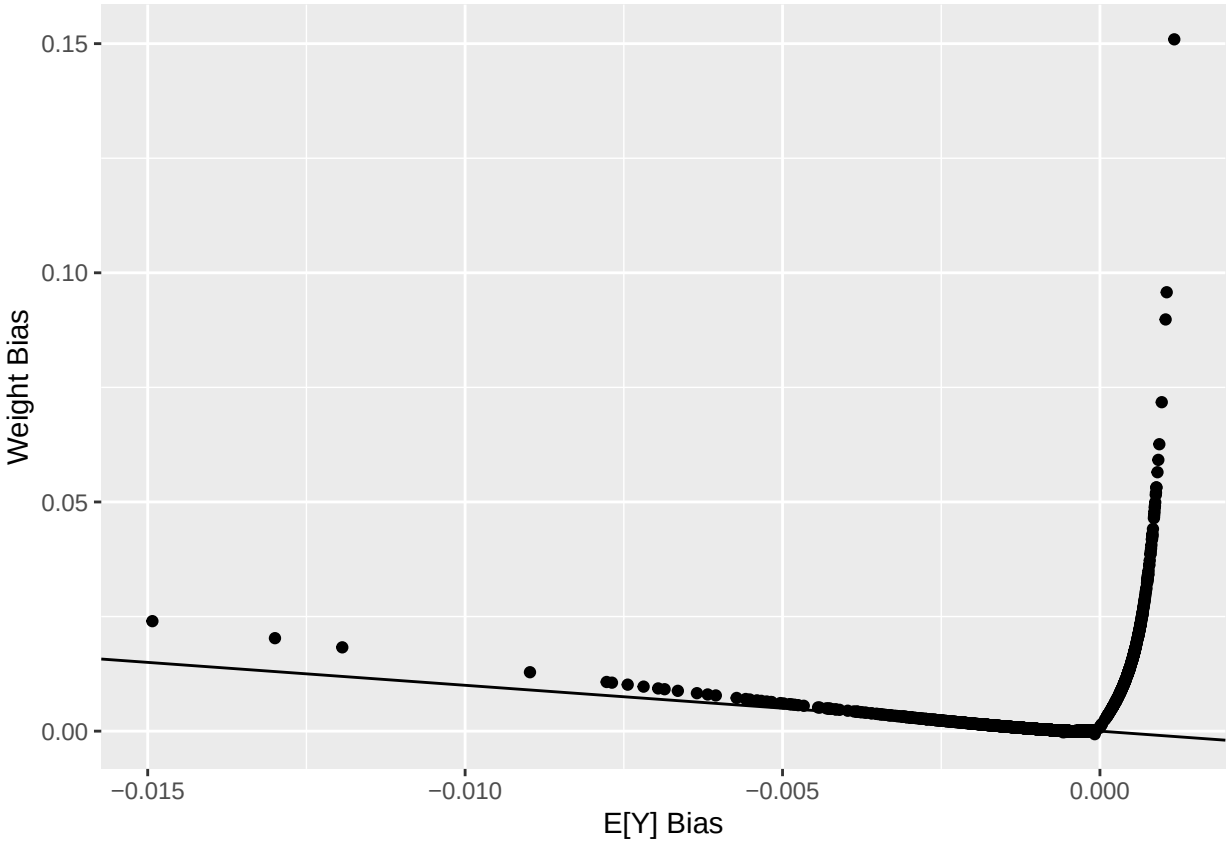
```
  pal <- brewer.pal(3,"Set2")
  out <- ggplot(data=df)+
    geom_point(aes(x=bias,y=w))+
    geom_abline(aes(intercept=0,slope=-1))+
    labs(x="E[Y] Bias",y="Weight Bias")
  return(out)
}

plot_EY(D_lin,Dfn_lin)
```



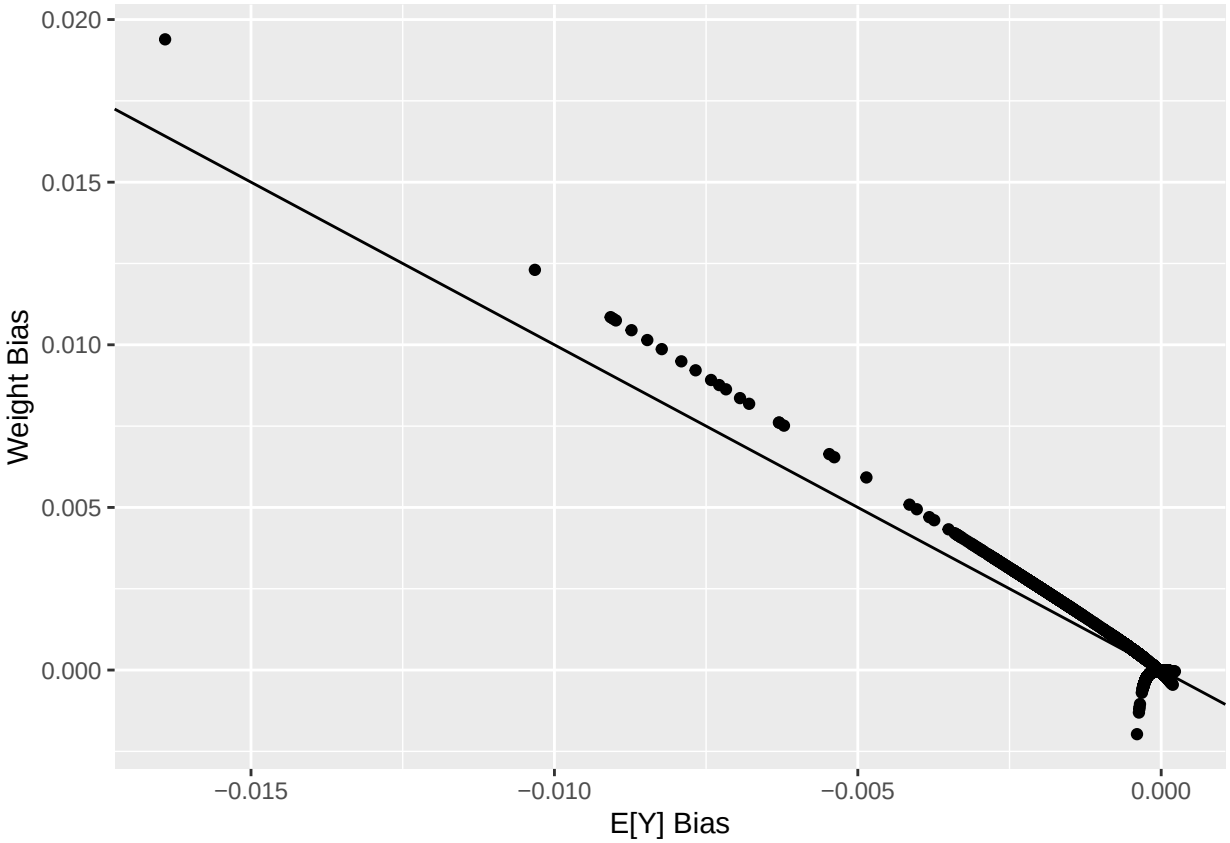

```
plot_EY(D_log,Dfn_log)
```




```
plot_EY(D_exp,Dfn_exp)
```




```
plot_EY(D_sin,Dfn_sin)
```



Roughly, the bias attributable to a single observation can be broken into 1) how far the residualized Y_i is from the actual value of Y and 2) how far the weight placed on the observation is from the proper weight. In particular, we can write

$$\tau^{OLS} - \tau = \sum_{i=1}^{N} [\frac{2}{N}(\tilde{Y}_i - Y_i) + \frac{N\tilde{D}_i - 2\sum_{i=1}^{N}\tilde{D}_i^2}{N\sum_{i=1}^{N}\tilde{D}_i^2}\tilde{Y}_i$$

These two sources of bias are plotted in the figures above. From the plot for our sine CEF, we see that a number of observations with negative weights offset the generally slightly larger weight bias. For the linear and exponential CEFs the bias is generally positive due to observations receiving too much weight. Finally, the logarithmic CEF has a number of observations with large negative weights, biasing the estimate downwards.