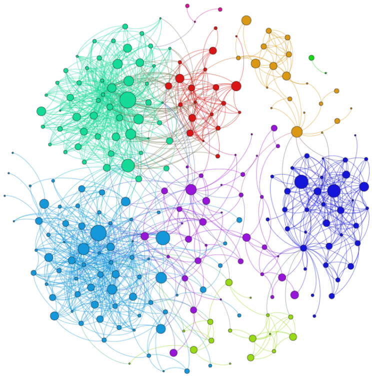
Regression and Matching for Causal Inference

with no time for matching

Jiaming Mao

Xiamen University



Controlling for Confounding

Suppose our goal is to learn $E[y|\text{do}(x)]$. If there are no open back-door paths from x to y[1], then $E[y|\text{do}(x)] = E[y|x]$.

When there are open back-door paths from x to y, according to the back-door criterion, if we observe a set of *pre-treatment*[2] variables z such that conditioning on z blocks these paths, then $E[y|\text{do}(x)]$ is nonparametrically identified:

$$E[y|\text{do}(x)] = \int E[y|x,z]\, p(z)\, dz \tag{1}$$

In this case, we say conditional on z, the treatment assignment mechanism is ignorable, and x is exogenous to y.

[1] according to our causal model $\mathcal{M}$.

[2] i.e., no variable in z is a descendant of x.

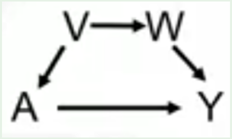
Controlling for Confounding

The back-door criterion shows that we do not need to observe and condition on all confounders, but only a *(minimally) sufficient* set of variables that renders all back-door paths blocked[3].

[3]When all common causes are fully observed, we say there is no unmeasured confounding, or there is selection on observables. As the back-door criterion makes clear, this is sufficient but not necessary for causal effect identification.

Controlling for Confounding

Example 1

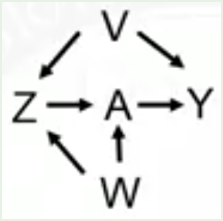


$$\begin{tikzcd} & V \arrow[r] \arrow[dl] & W \arrow[dr] & \\ A \arrow[rrr] & & & Y \end{tikzcd}$$

- Treatment: A; Outcome: Y
- Sets of variables sufficient to control for confounding: $\{V\}$, $\{W\}$, $\{V, W\}$

Controlling for Confounding

Example 2


$$\begin{tikzcd} & V \arrow[dl] \arrow[dr] & \\ Z \arrow[r] & A \arrow[r] & Y \\ & W \arrow[ul] \arrow[u] & \end{tikzcd}$$

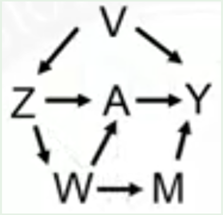
- Treatment: A; Outcome: Y
- There is an open back-door path: $A \leftarrow Z \leftarrow V \rightarrow Y$[a].
- Sets of variables sufficient to control for confounding: $\{V\}$, $\{Z, W\}$, $\{Z, V\}, \{Z, V, W\}$[b].

[a]The other back-door path $A \leftarrow W \rightarrow Z \leftarrow V \rightarrow Y$ is blocked by the collider Z.

[b]Conditioning only on Z alone would *open* the back-door path $A \leftarrow W \rightarrow Z \leftarrow V \rightarrow Y$.

Controlling for Confounding

Example 3


$$\begin{tikzcd}
 & V \arrow[dl] \arrow[dr] & \\
Z \arrow[r] \arrow[d] & A \arrow[r] & Y \\
W \arrow[ur] \arrow[r] & M \arrow[ur] &
\end{tikzcd}$$

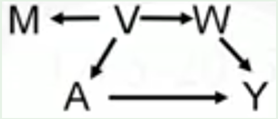
- Treatment: A; Outcome: Y
- Open back-door paths: $A \leftarrow Z \leftarrow V \rightarrow Y$, $A \leftarrow Z \rightarrow W \rightarrow M \rightarrow Y$, $A \leftarrow W \leftarrow Z \leftarrow V \rightarrow Y$, $A \leftarrow W \rightarrow M \rightarrow Y$.
- Sets of variables sufficient to control for confounding: $\{Z, W\}$, $\{Z, M\}$, $\{V, W\}$, $\{V, M\}$, $\{Z, W, M\}$, $\{Z, V, W\}$, $\{Z, V, M\}$, $\{V, M, W\}$, $\{Z, V, W, M\}$.

Disjunctive Cause Criterion

- One method to select what variables to control for confounding among the observed variables is the **disjunctive cause criterion** – control for all (observed) causes of treatment x and outcome y.
- Let V be the set of variables selected based on the disjunctive cause criterion. If there exists a set of observed variables z that satisfies the back-door criterion, then $z \subset V$.
- The disjunctive cause criterion places less knowledge requirement of the underlying causal structure on the investigator.

Disjunctive Cause Criterion

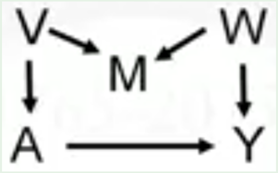
Example 1


$$\begin{tikzcd} M & V \arrow[l] \arrow[r] \arrow[d] & W \arrow[d] \\ & A \arrow[r] & Y \end{tikzcd}$$

- Treatment: A; Outcome: Y
- Observed: $\{A, Y, M, V, W\}$
- Disjunctive cause criterion selects: $\{V, W\}$
- Satisfies back-door criterion? Yes

Disjunctive Cause Criterion

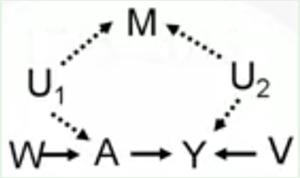
Example 2


$$\begin{tikzcd} V \arrow[r] \arrow[d] & M & W \arrow[l] \arrow[d] \\ A \arrow[rr] & & Y \end{tikzcd}$$

- Treatment: A; Outcome: Y
- Observed: $\{A, Y, M, V, W\}$
- Disjunctive cause criterion selects: $\{V, W\}$
- Satisfies back-door criterion? Yes

Disjunctive Cause Criterion

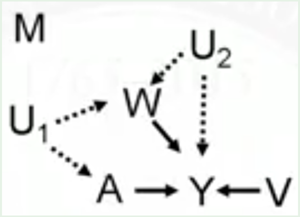
Example 3



- Treatment: A; Outcome: Y
- Observed: $\{A, Y, M, V, W\}$; Unobserved: $\{U_1, U_2\}$
- Disjunctive cause criterion selects: $\{V, W\}$
- Satisfies back-door criterion? Yes

Disjunctive Cause Criterion

Example 4



- Treatment: A; Outcome: Y
- Observed: $\{A, Y, M, V, W\}$; Unobserved: $\{U_1, U_2\}$
- Disjunctive cause criterion selects: $\{V, W\}$
- Satisfies back-door criterion? No. However, here no set of observed variables satisfies the back-door criterion.

Causal Effect Estimation under Sufficient Control for Confounding

Assume we observe a set of variables z that satisfies the back-door criterion. Then according to (1), we have:

$$
\begin{aligned}
E\left[y|\mathrm{do}\left(x=a\right)\right] &= \int E\left[y|x=a,z\right]p\left(z\right)dz \\
&\approx \frac{1}{N}\sum_i E\left[\left.y_i\right|x_i=a,z_i\right] \qquad (2)
\end{aligned}
$$

Causal Effect Estimation under Sufficient Control for Confounding

Therefore, if we can estimate $E[y|x,z]$, then we have an estimate of $E[y|\mathrm{do}(x)]$[4].

[4] If z is discrete with values $\{1,\ldots,K\}$, then

$$E[y|\mathrm{do}(x)] = \sum_{k=1}^{K} E[y|x,z=k]\, p(z=k) \tag{3}$$

, which we can estimate by estimating $E[y|x,z=k]$ separately for each value of k and then combining them according to (3) using the empirical distribution of z as $p(z)$. This strategy is called **standardization** and is possible if (1) z is discrete or continuous but can be discretized without loss of much information; (2) z is low-dimensional ($N \gg \dim(z)$).

Causal Effect Estimation under Sufficient Control for Confounding

Once we have an estimate of $E\left[y|\text{do}\left(x\right)\right]$, we can use it to compute the average treatment effect:

$$\text{ATE}\left(x\right) = \frac{dE\left[y|\text{do}\left(x\right)\right]}{dx}$$

If $x \in \{0, 1\}$ is a binary treatment, then

$$\text{ATE} = E\left[y|\text{do}\left(x=1\right)\right] - E\left[y|\text{do}\left(x=0\right)\right]$$

Regression for Causal Inference

- As the discussion on page 13 makes clear, with a sufficient set of observed variables z that controls for confounding, the causal effect learning problem boils down to the statistical learning problem of estimating the target function $E[y|x,z]$.
- This can be achieved using *any* appropriate parametric, semi-parametric, or non-parametric regression models: linear and polynomial regression, splines, tree-based methods, support vector machines, neural networks, etc.

Homogeneous and Heterogeneous Treatment Effects

Consider a binary treatment $x \in \{0, 1\}$. Let $\tau_i = \mathcal{Y}_i^1 - \mathcal{Y}_i^0$ be the individual treatment effect on individual i.

- In the special case that $\tau_i = \tau \ \forall i$, we say the treatment effect (in this population) is homogeneous.
- If z is a variable that describes individual characteristics, then homogenous treatment effect implies that $E[y|x=1,z] - E[y|x=0,z] = \tau$.
- In general, however, we should expect individual treatment effects to be heterogeneous in any given population.
- Under heterogeneous treatment effects, individual characteristics *could* be effect modifiers: it is possible for $E[y|x=1,z] - E[y|x=0,z] = \tau(z)$ to vary by z.

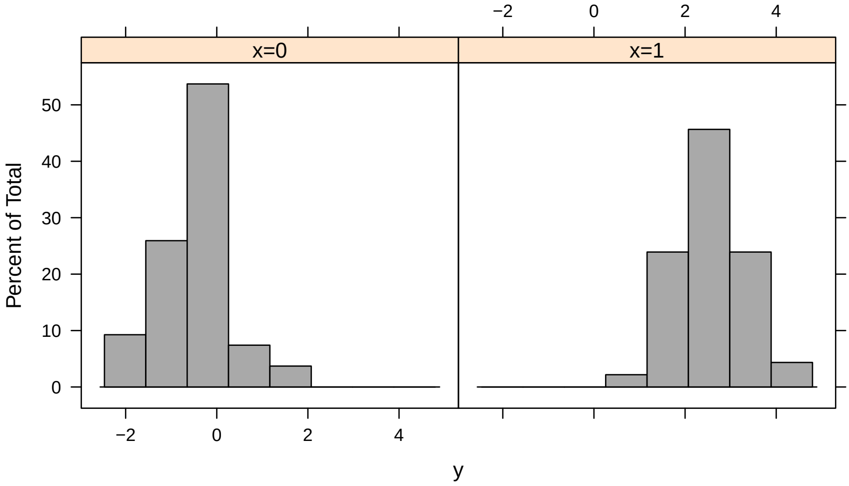
Simulation 1

Let's generate some data with binary treatment x and outcome y:

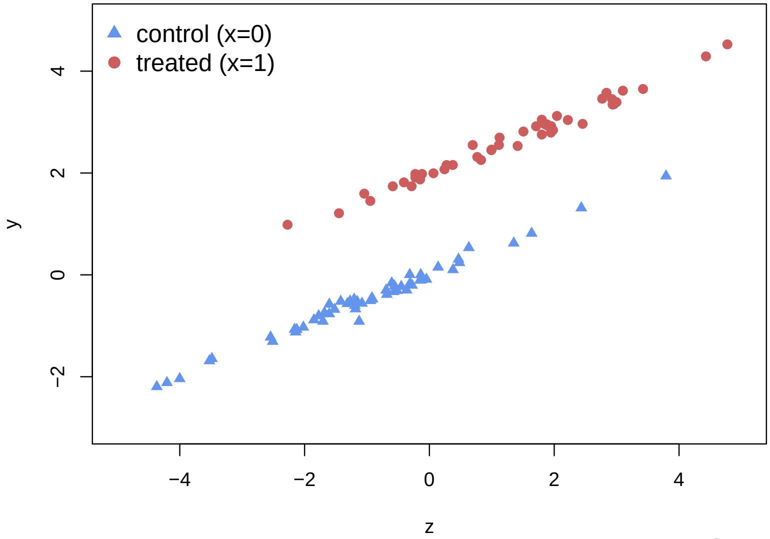
$$\begin{aligned} z &\sim \mathcal{N}(0,4) \\ x &\leftarrow \text{Bernoulli}\left((1+\exp(-z))^{-1}\right) \\ y &\leftarrow 0.5z + 2x + e, \quad e \sim \mathcal{N}(0, 0.01) \end{aligned}$$

```r
# Simulation
require(sigmoid)
n = 100
z = 2*rnorm(n)
x = rbinom(n,1,sigmoid(z))
y = 0.5*z + 2*x + 0.1*rnorm(n)
```

Simulation 1



Simulation 1



Simulation 1

Here the the treatment effect is homogeneous: $\tau = 2$.

Naive comparison of $E\left[y|x=1\right]$ and $E\left[y|x=0\right]$ gives biased estimate:

```
mean(y[x==1]) - mean(y[x==0])

## [1] 3.107458
```

Need to control for confounder z ...

Simulation 1

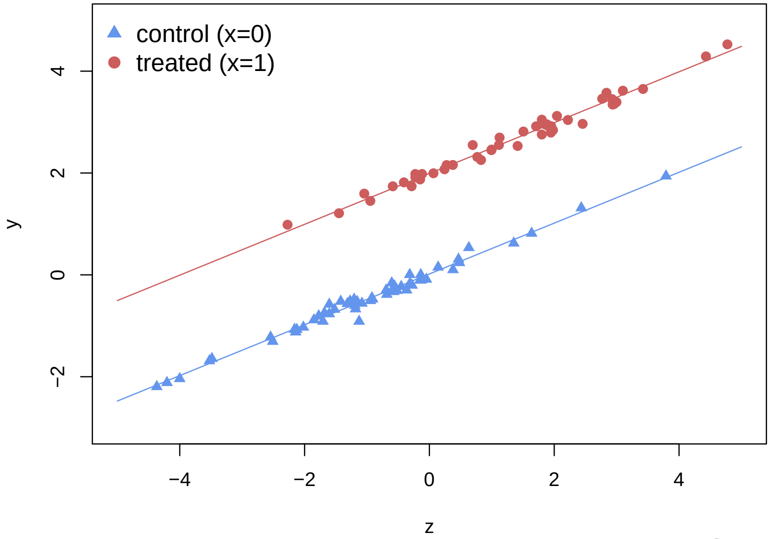
Linear regression:

$$y = \beta_0 + \beta_1 x + \beta_2 z + e \tag{4}$$

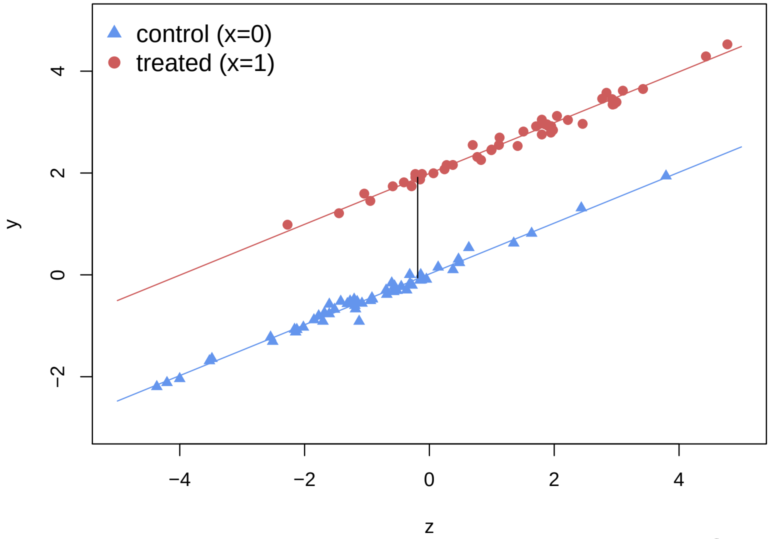
```
require(AER)
data = data.frame(y=y,x=x,z=z)
fit = lm(y~.,data)
coeftest(fit)
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0177336  0.0149535  1.1859   0.2386
## x           1.9727686  0.0248930 79.2499   <2e-16 ***
## z           0.4990824  0.0065954 75.6710   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulation 1



Simulation 1



Simulation 1

In model (4), $\widehat{\beta_1}$ *is* our estimate of the ATE:

$$\begin{aligned}
\widehat{\text{ATE}} &= \widehat{E}\left[y|\text{do}\left(x=1\right)\right] - \widehat{E}\left[y|\text{do}\left(x=0\right)\right] \\
&= \int \left(\widehat{E}\left[y|x=1,z\right] - \widehat{E}\left[y|x=0,z\right]\right) p\left(z\right) dz \\
&= \int \left[\left(\widehat{\beta_0} + \widehat{\beta_1} + \widehat{\beta_2} z\right) - \left(\widehat{\beta_0} + \widehat{\beta_2} z\right)\right] p\left(z\right) dz \\
&= \widehat{\beta_1}
\end{aligned}$$

Simulation 1

In general, if the regression model for $E\left[y|x,z\right]$ is additive in x:

$$E\left[y|x,z\right]=\phi_1\left(x\right)+\phi_2\left(z\right) \tag{5}$$

, i.e., if we assume *no interaction* between x and z, then

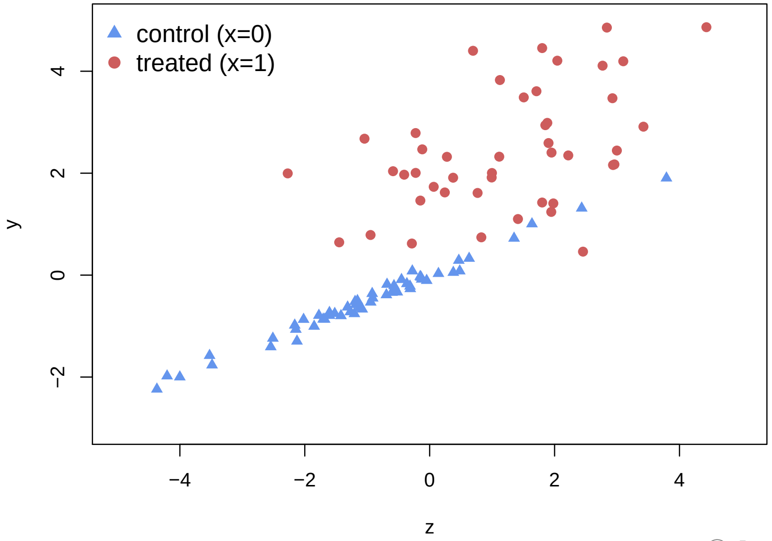
$$\begin{aligned}\text{ATE}\left(x\right)&=\frac{dE\left[y|\text{do}\left(x\right)\right]}{dx}\\&=\int\frac{\partial E\left[y|x,z\right]}{\partial x}p\left(z\right)dz\\&=\int\phi_1'\left(x\right)p\left(z\right)dz\\&=\phi_1'\left(x\right)=\frac{\partial E\left[y|x,z\right]}{\partial x}\end{aligned}$$

Simulation 2

$$z \sim \mathcal{N}(0,4)$$
$$x \leftarrow \text{Bernoulli}\left((1+\exp(-z))^{-1}\right)$$
$$y \leftarrow 0.5z + \alpha x + e, \quad \alpha \sim \mathcal{N}(2,1), \quad e \sim \mathcal{N}(0, 0.01)$$

```
# Simulation
n = 100
z = 2*rnorm(n)
x = rbinom(n,1,sigmoid(z))
y = 0.5*z + (2+1*rnorm(n))*x + 0.1*rnorm(n)
```

Simulation 2



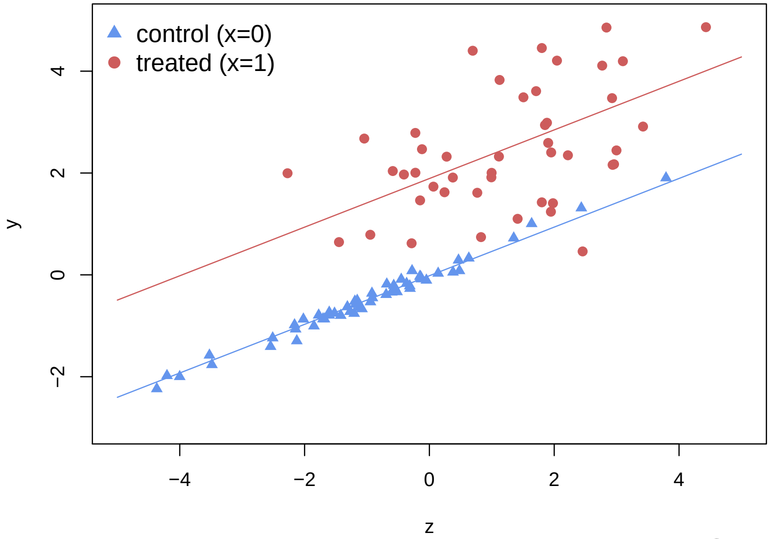
Simulation 2

Linear regression:

$$y = \beta_0 + \beta_1 x + \beta_2 z + e \tag{6}$$

```
data = data.frame(y=y,x=x,z=z)
fit = lm(y~.,data)
coeftest(fit)

##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01704    0.10899 -0.1564   0.8761
## x            1.90869    0.18143 10.5203   <2e-16 ***
## z            0.47746    0.04807  9.9326   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulation 2



Simulation 2

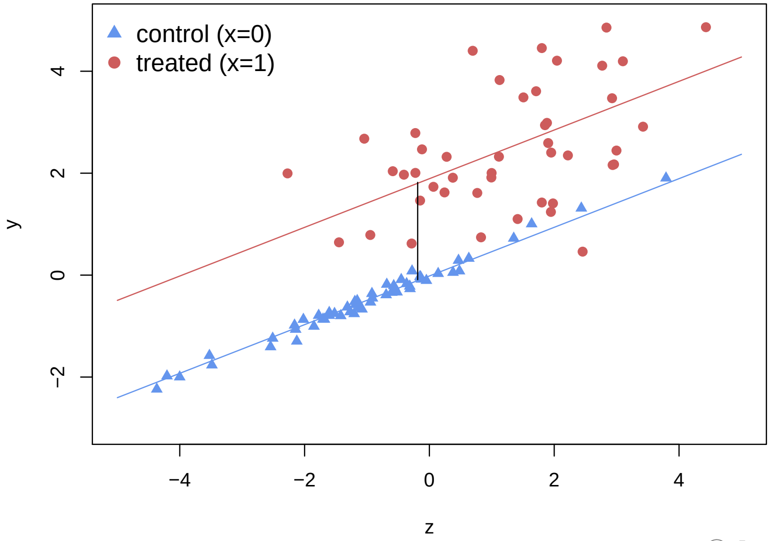
Here the the treatment effect is heterogeneous: $\tau_i = \alpha_i$, but does not vary with z.

The average treatment effect

$$\begin{aligned} \text{ATE} &= E(\tau) = E(\alpha) \\ &= E[y|x=1,z] - E[y|x=0,z] \end{aligned}$$

Hence in (6), $\widehat{\beta_1}$ is again our estimate of the ATE.

Simulation 2

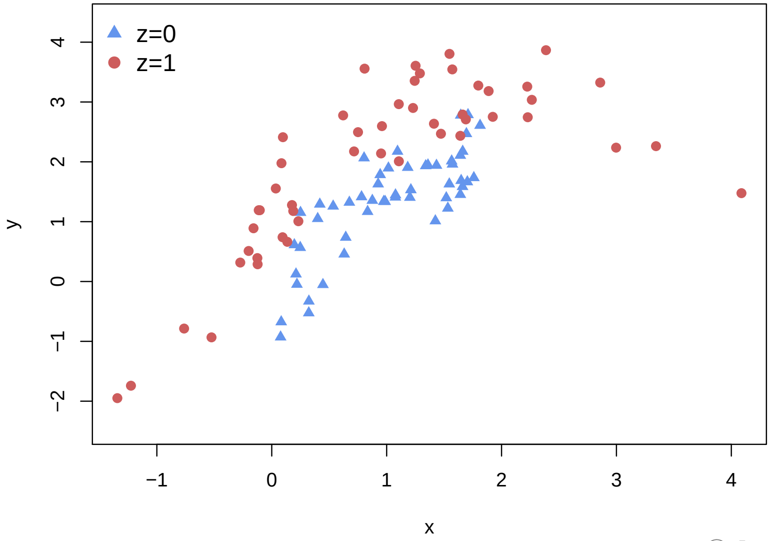


Simulation 3

$$z \sim \text{Bernoulli}(0.5)$$
$$x \leftarrow z \times \mathcal{N}(0,1) + U(0,2)$$
$$y \leftarrow z + 2x - 0.5x^2 + e, \quad e \sim \mathcal{N}(0, 0.25)$$

```r
# Simulation
n = 100
z = rbinom(n,1,0.5)
x = 2*runif(n) + z*rnorm(n)
y = z + 2*x - 0.5*x^2 + 0.5*rnorm(n)
```

Simulation 3



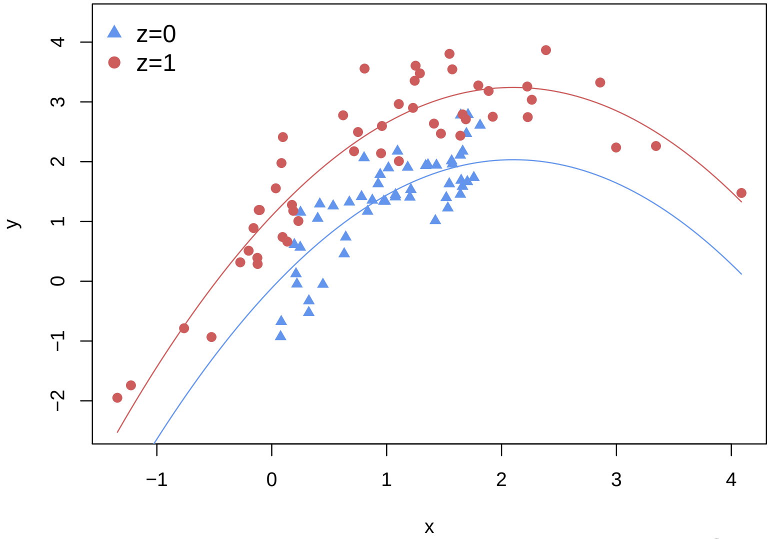
Simulation 3

Polynomial regression in x:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 z + e \tag{7}$$

```
fit = lm(y ~ poly(x,2,raw=T) + z)
coeftest(fit)

##
## t test of coefficients:
##
##                        Estimate Std. Error  t value Pr(>|t|)
## (Intercept)           -0.114061   0.098290  -1.1605   0.2487
## poly(x, 2, raw = T)1  2.042029   0.105225  19.4062   <2e-16 ***
## poly(x, 2, raw = T)2 -0.485534   0.039532 -12.2820   <2e-16 ***
## z                      1.208719   0.108421  11.1484   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulation 3



Simulation 3

Here the treatment effect is homogeneous but non-constant (in x).

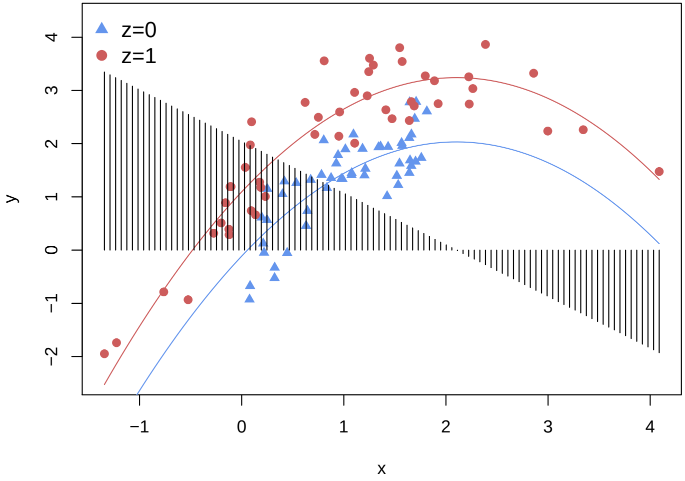
Given the additive structure of (5),

$$\text{ATE}(x) = \phi_1'(x) = 2 - x$$

Given (7),

$$\widehat{\text{ATE}}(x) = \widehat{\beta_1} + 2\widehat{\beta_2}x$$

Simulation 3



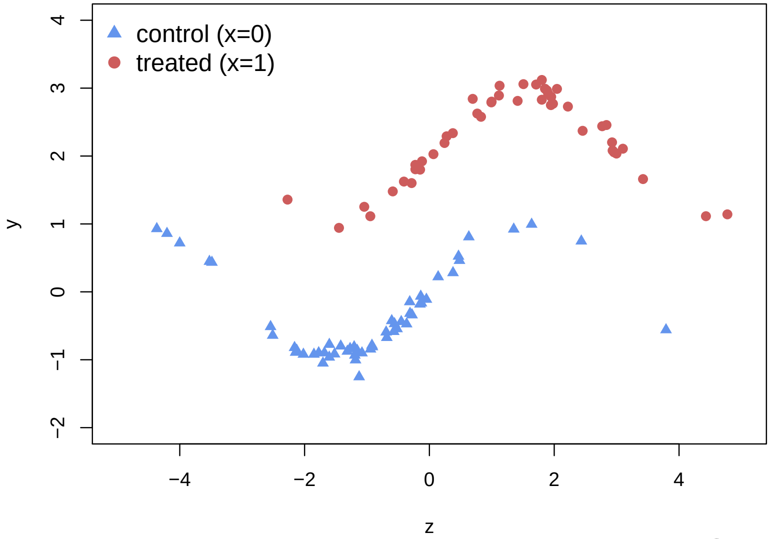
Vertical lines represent $\widehat{\beta_1} + 2\widehat{\beta_2}x$: the estimated ATE.

Simulation 4

$$z \sim \mathcal{N}(0,4)$$
$$x \leftarrow \text{Bernoulli}\left((1+\exp(-z))^{-1}\right)$$
$$y \leftarrow \sin(z) + 2x + e, \quad e \sim \mathcal{N}(0, 0.01)$$

```r
# Simulation
n = 100
z = 2*rnorm(n)
x = rbinom(n,1,sigmoid(z))
y = sin(z) + 2*x + 0.1*rnorm(n)
```

Simulation 4



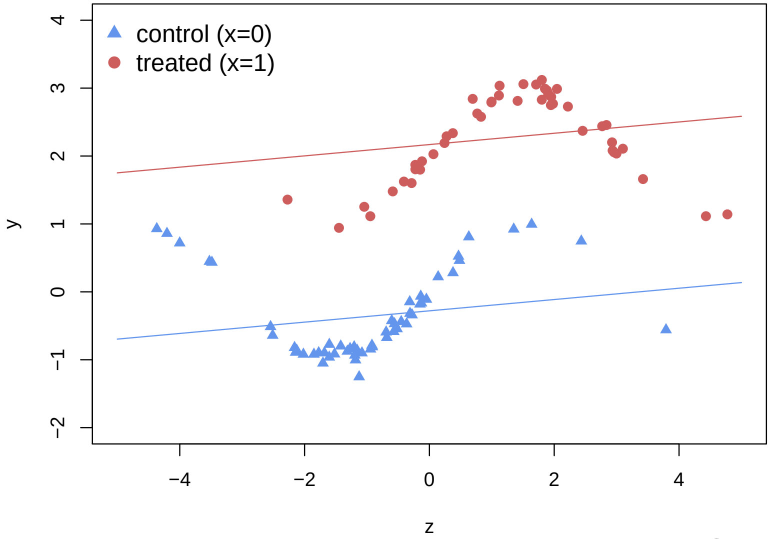
Simulation 4

Linear regression:

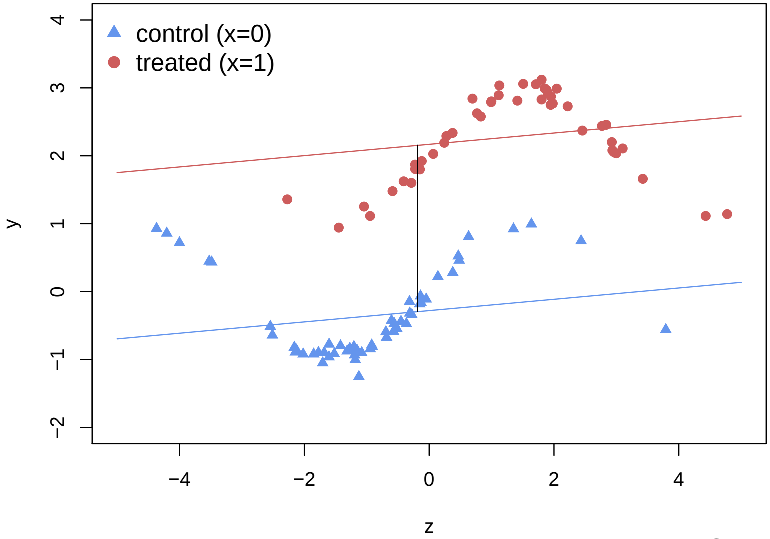
$$y = \beta_0 + \beta_1 x + \beta_2 z + e$$

```
data = data.frame(y=y,x=x,z=z)
fit = lm(y~.,data)
coeftest(fit)

##
## t test of coefficients:
##
##                Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.279858   0.094458 -2.9628  0.003835 **
## x            2.448736   0.157244 15.5729 < 2.2e-16 ***
## z            0.083252   0.041662  1.9983  0.048487 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulation 4



Simulation 4



Simulation 4

Semi-parametric generalized additive model:

$$y = \beta_0 + \beta_1 x + g(z) + e$$

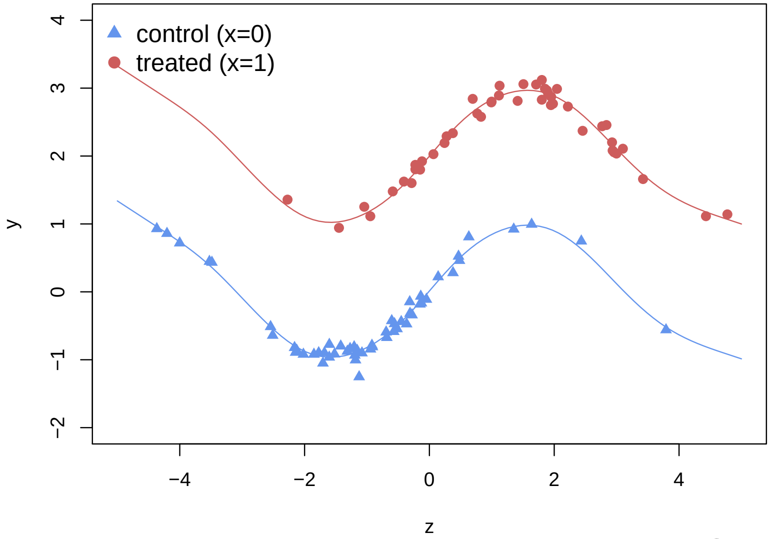
, where $g(z)$ is a smoothing spline.

```r
library(mgcv)
fit = gam(y ~ x + s(z),data,family=gaussian)
```

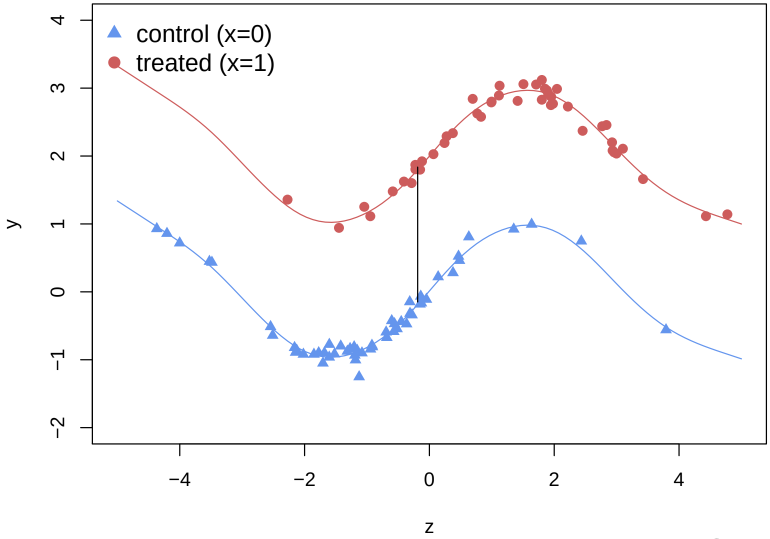
Simulation 4

```
summary(fit)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ x + s(z)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.06169    0.01589  -3.883 0.000197 ***
## x            1.98592    0.02665  74.509  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df     F p-value
## s(z) 8.455  8.914 422.5  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Simulation 4



Simulation 4

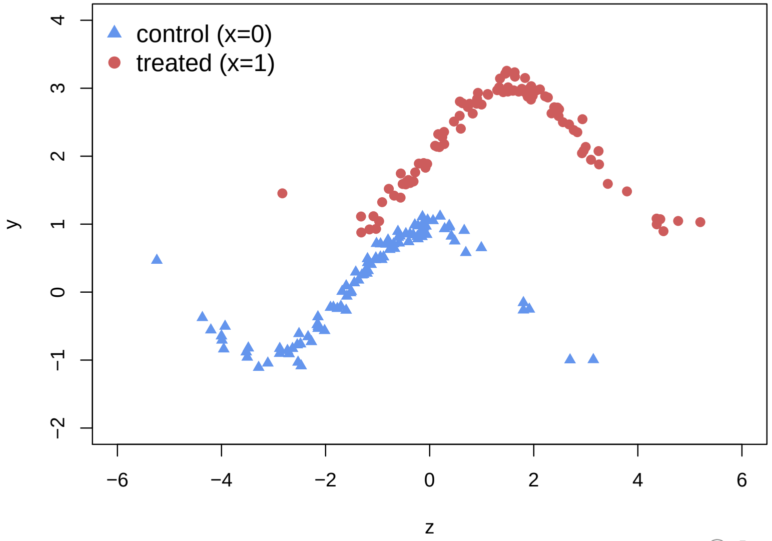


Simulation 5

$$\begin{aligned} z &\sim \mathcal{N}(0,4) \\ x &\leftarrow \text{Bernoulli}\left((1+\exp(-z))^{-1}\right) \\ \mathcal{Y}^0 &\leftarrow \cos(z)+\epsilon, \;\; \epsilon \sim \mathcal{N}(0,0.01) \\ \mathcal{Y}^1 &\leftarrow 2+\sin(z)+\varepsilon, \;\; \varepsilon \sim \mathcal{N}(0,0.01) \\ y &= x\mathcal{Y}^1 + (1-x)\mathcal{Y}^0 \end{aligned}$$

```
# Simulation
n = 200
z = 2*rnorm(n)
x = rbinom(n,1,sigmoid(z))
y0 = cos(z) + 0.1*rnorm(n)
y1 = 2 + sin(z) + 0.1*rnorm(n)
y = y0*(x==0) + y1*(x==1)
```

Simulation 5



Simulation 5

Here the treatment effect is heterogeneous and varies with z:

$$\begin{aligned}\text{ATE} &= E\left[\mathcal{Y}^1 - \mathcal{Y}^0\right] \\ &= \int \left(E\left[y|x=1,z\right] - E\left[y|x=0,z\right]\right) p\left(z\right) dz \\ &\approx \frac{1}{N}\sum_i \left(E\left[y_i|\, x_i = 1, z_i\right] - E\left[y_i|\, x_i = 0, z_i\right]\right)\end{aligned}$$

```r
# ATE
ate = mean(y1 - y0)
ate

## [1] 1.869096
```

Simulation 5

To model the interaction between x and z, we can run the following linear regression:

$$y = \beta_0 + \beta_1 x + \beta_2 z + \beta_3 x \cdot z + e \tag{8}$$

Or more flexibly, we can run the following set of regressions[5]:

$$\begin{cases} y = \beta_0 + \beta_1 z + \epsilon & \text{if } x = 0 \\ y = \beta_2 + \beta_3 z + \varepsilon & \text{if } x = 1 \end{cases} \tag{9}$$

Estimating (9) $\Rightarrow \widehat{E}\left[y|x=0,z\right]$ and $\widehat{E}\left[y|x=1,z\right]$, from which we obtain:

$$\widehat{\text{ATE}} \approx \frac{1}{N}\sum_i \left(\widehat{E}\left[y_i|\, x_i = 1, z_i\right] - \widehat{E}\left[y_i|\, x_i = 0, z_i\right]\right)$$

[5](8) and (9) are equivalent when x is binary.

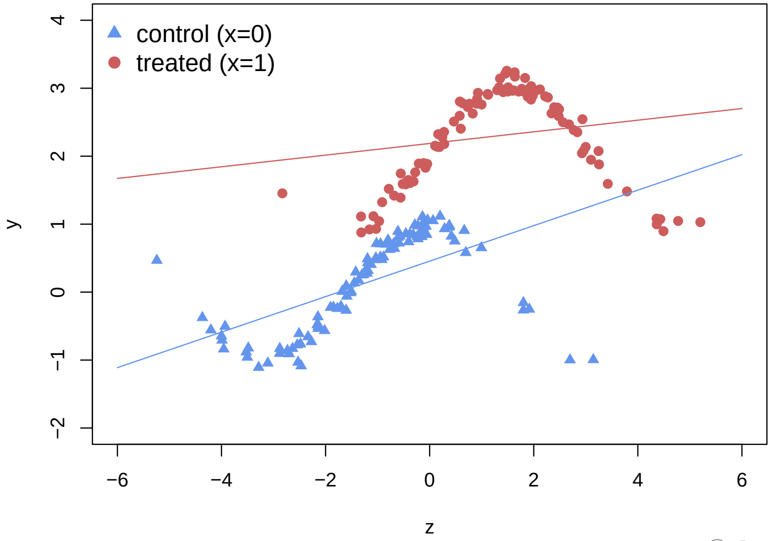
Simulation 5

```r
# Linear Regression
data = data.frame(y=y,x=x,z=z)
fit0 = lm(y ~ z,data,subset=(x==0))
fit1 = lm(y ~ z,data,subset=(x==1))

# Estimated ATE
y0hat = predict(fit0,data)
y1hat = predict(fit1,data)
atehat = mean(y1hat - y0hat)
atehat

## [1] 1.722792
```

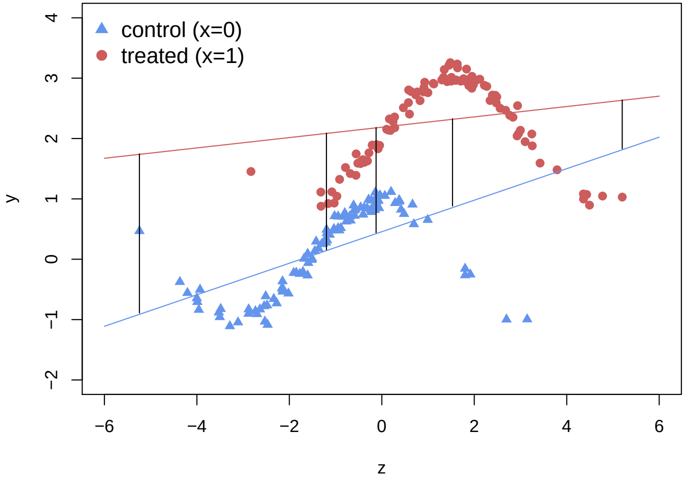
Simulation 5

```
coeftest(fit0)

##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 0.455760   0.074658  6.1047 2.123e-08 ***
## z           0.261113   0.038104  6.8527 6.698e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

coeftest(fit1)

##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.186996   0.090796  24.087  < 2e-16 ***
## z           0.085725   0.045190   1.897  0.06074 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulation 5



Simulation 5



Vertical lines represent estimated ATEs conditional on z at 1%, 25%, 50%, 75%, and 99% percentiles.

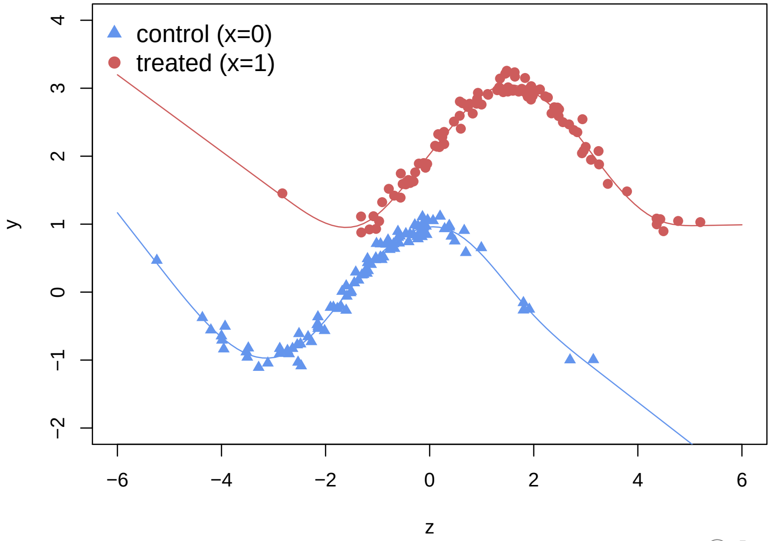
Simulation 5

Smoothing splines:

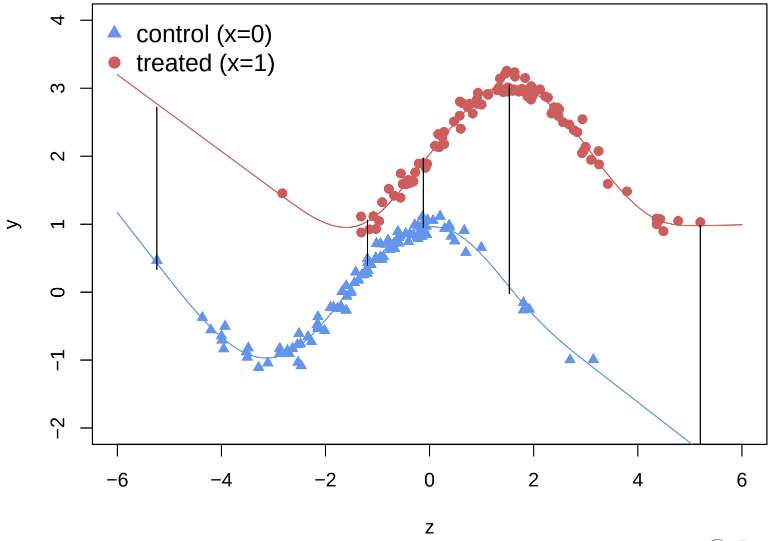
$$\begin{cases} y = g_1(z) + \epsilon & \text{if } x = 0 \\ y = g_0(z) + \varepsilon & \text{if } x = 1 \end{cases}$$

```
# Smoothing Splines
fit0 = gam(y ~ s(z),data,subset=(x==0),family=gaussian)
fit1 = gam(y ~ s(z),data,subset=(x==1),family=gaussian)

# Estimated ATE
y0hat = predict(fit0,data)
y1hat = predict(fit1,data)
atehat = mean(y1hat - y0hat)
atehat

## [1] 1.877149
```

Simulation 5



Simulation 5

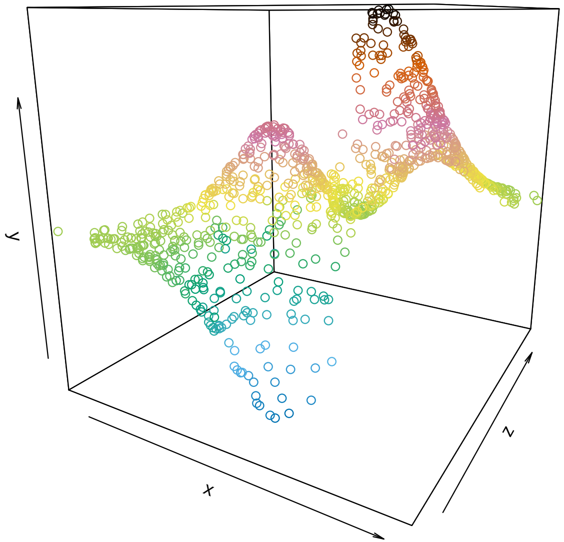


Simulation 6

$$
\begin{aligned}
z &\sim U\left(-\frac{5}{2}, \frac{5}{2}\right) \\
x &\leftarrow \frac{1}{2} z + \frac{1}{2}\mathcal{N}(0,1) \\
y &\leftarrow 3(1-x)^2 \exp\left(-x^2 - (1+z)^2\right) - 10\left(\frac{1}{5}x - x^3 - z^5\right) \\
&\qquad \times \exp\left(-x^2 - z^2\right) - \frac{1}{3}\exp\left(-(1+x)^2 - z^2\right)
\end{aligned}
$$

```r
# Simulation
n = 1000
z = 5*runif(n) - 2.5
x = 0.5*z + 0.5*rnorm(n)
g = expression((3*(1-x)^2)*exp(-(x^2)-(z+1)^2)-
                 10*(x/5-x^3-z^5)*exp(-x^2-z^2)-1/3*exp(-(x+1)^2-z^2))
y = eval(g)
```

Simulation 6



Simulation 6

Here the treatment effect is both heterogeneous and varies with z, and non-constant in x:

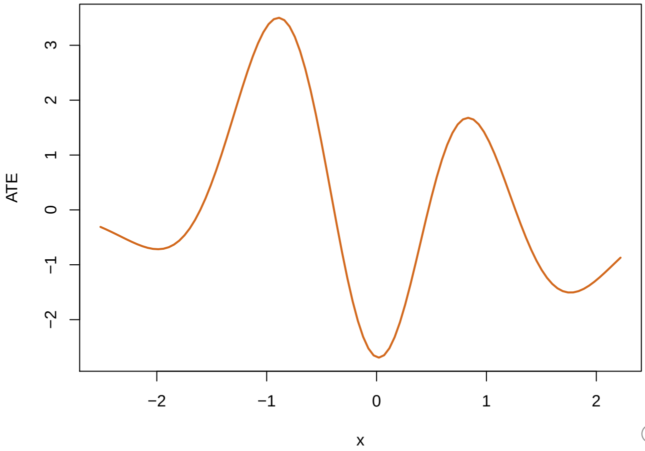
$$\begin{aligned} \text{ATE}(x) &= \int \frac{\partial E[y|x,z]}{\partial x} p(z)\, dz \\ &\approx \frac{1}{N} \sum_i \frac{\partial E[y_i|\, x, z_i]}{\partial x} \end{aligned}$$

Alternatively,

$$\begin{aligned} \text{ATE}(x) &= \frac{d}{dx} \int E[y|x,z]\, p(z)\, dz \\ &\approx \frac{d}{dx} \left(\frac{1}{N} \sum_i E[y_i|\, x, z_i] \right) \end{aligned}$$

Simulation 6

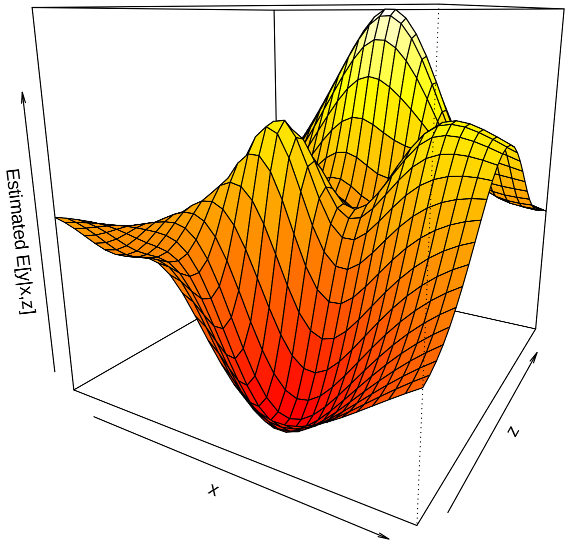
```r
# ATE
dx = D(g,"x") # partial derivative w.r.t. x
xgrid = seq(min(x),max(x),length.out=100) # calculate ate on xgrid
ate = sapply(xgrid, function(a) mean(eval(dx,envir=list(x=a,z=z))))
```



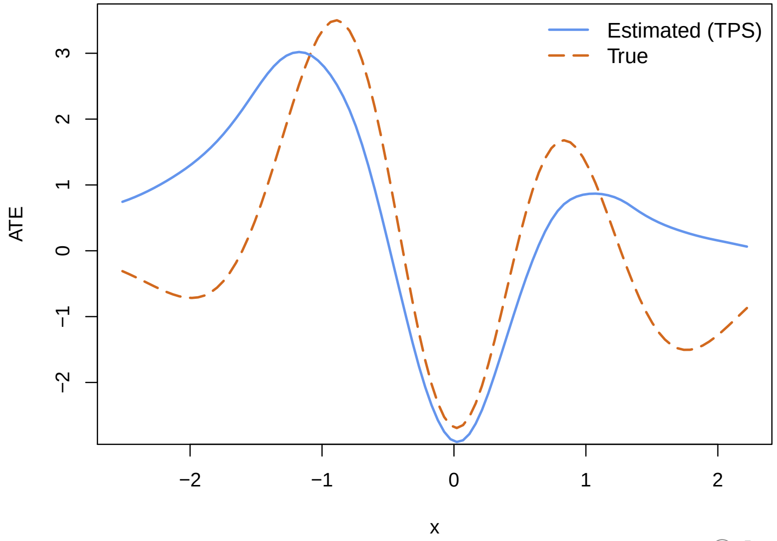
Simulation 6

```
#####################
# Thin-plate Spline #
#####################
fit = gam(y ~ s(x,z),family=gaussian)

# Estimated ATE
# here we first compute E[y|do(x)] = Integrate(E[y|x,z]) over z
# then we take the derivative of E[y|do(x)] to obtain ATE
require(numDeriv)
cef = function(a) mean(predict(fit,data.frame(x=a,z=z)))
atehat = sapply(xgrid,function(a) grad(cef,a))
```

Simulation 6



Simulation 6



Simulation 6

To choose the best statistical model for $E[y|x,z]$, we can split our data set into training and test sets and perform model selection.

```r
# Create Training and Test Sets
require(caret)
train = createDataPartition(y,p=0.5,list=F)
data = data.frame(x=x,z=z,y=y)
data_train = data[train,]
data_test = data[-train,]
```

Simulation 6

```
#####################
# Thin-plate Spline #
#####################
fit.tps = gam(y ~ s(x,z),data_train,family=gaussian)

# test err
yhat = predict(fit.tps,data_test)
mean((data_test$y - yhat)^2)

## [1] 0.07927297
```

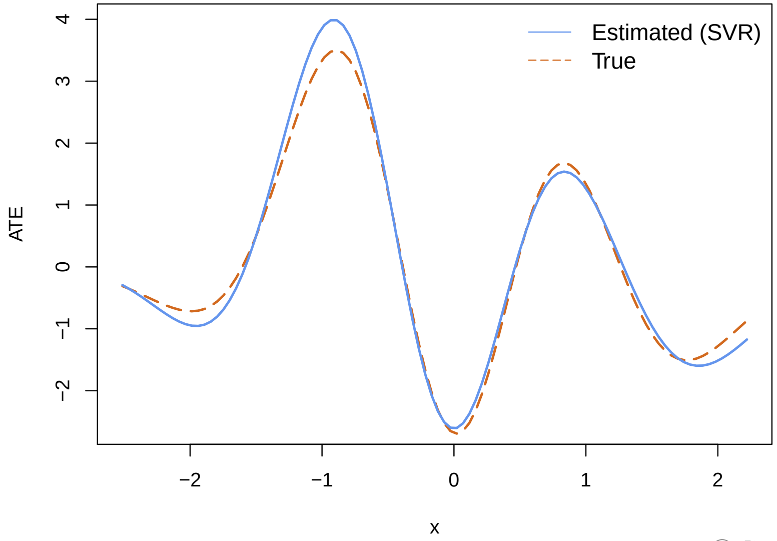
Simulation 6

```
##############################
# Support Vector Regression #
##############################
require(e1071)
fit.svm = svm(y~.,data_train,kernel="radial",
              cost=1e3,gamma=1) # tuning parameters chosen by cv

# test err
yhat = predict(fit.svm,data_test)
mean((data_test$y - yhat)^2)

## [1] 0.04508208
```

Simulation 6

```r
##############
# Best Model #
##############
# Fit the best model (here: svr) on the entire data set
# note: the "test set" here is technically still a validation set,
#       since we are using it to select the best model.
#       we can then re-fit the selected model on the combined data
fit = update(fit.svm,data=data)

# Estimated ATE
cef = function(a) mean(predict(fit,data.frame(x=a,z=data$z)))
atehat = sapply(xgrid,function(a) grad(cef,a))
```

Simulation 6

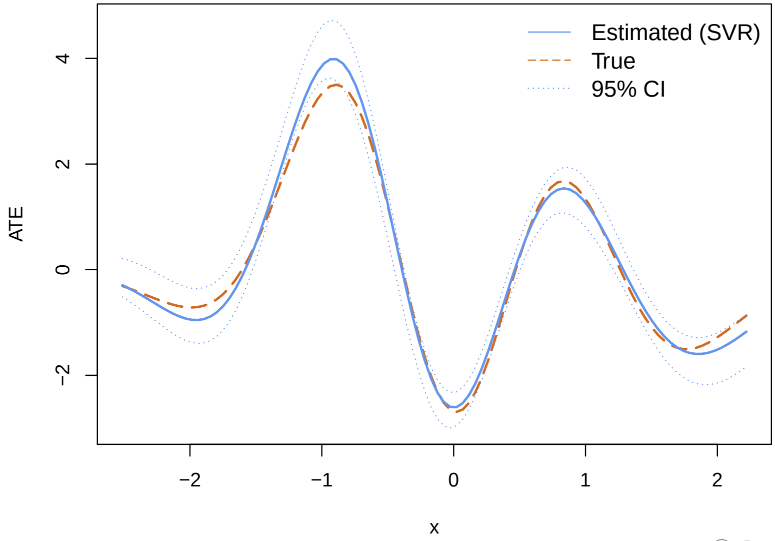


Simulation 6

We can use bootstrap to obtain standard errors and confidence intervals for estimated ATEs.

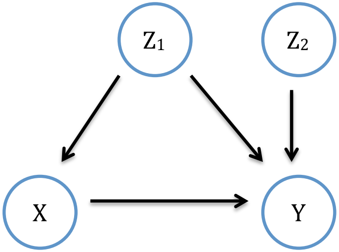
```r
# Function to calculate ATE
calcATE = function(data,index){
  data_i = data[index,]
  fit_i = update(fit,data=data_i)
  cef_i = function(a) mean(predict(fit_i,data.frame(x=a,z=data_i$z)))
  atehat_i = sapply(xgrid,function(a) grad(cef_i,a))
  return(atehat_i)
}

# Bootstrap
require(boot)
B = 1000 # number of bootstrap samples
bootATE = boot(data,calcATE,R=B,parallel="multicore") # use parallel

# Confidence Intervals
ateci = sapply(1:length(xgrid),
        function(i) boot.ci(bootATE,type="norm",index=i)$normal[2:3])
```

Simulation 6



Discussion: How Many Variables to Control?

Given the following causal diagram:



Both $\{z_1\}$ and $\{z_1, z_2\}$ are sufficient for controlling for confounding. Suppose we observe both z_1 and z_2[6], which set should we use?

[6] z_1 and z_2 can be sets of variables themselves.

Discussion: How Many Variables to Control?

$$E\left[y|\text{do}\left(x\right)\right] = \int E\left[y\left|x, z_1\right.\right] p\left(z_1\right) dz_1 \tag{10}$$

$$= \int\int E\left[y\left|x, z_1, z_2\right.\right] p\left(z_1, z_2\right) dz_1 dz_2 \tag{11}$$

- Given infinite data, (10) and (11) are equivalent ways of computing $E\left[y|\text{do}\left(x\right)\right]$.
- In finite sample, however, they are *not* equivalent.

Discussion: How Many Variables to Control?

Since z_1 is the set of confounders, causal reasoning dictates that we must control for z_1. In choosing whether to further control for z_2, we have the following considerations:

1. Given *nested* models $\mathcal{H}_1 = \{h(x, z_1)\} \subset \mathcal{H}_2 = \{h(x, z_1, z_2)\}$, if the integration in (10) and (11) can be done perfectly – if we know the true $p(z_1, z_2)$ – then choosing between $\mathcal{H}_1$ and $\mathcal{H}_2$ is a statistical variable selection problem: the best model generates the smallest prediction error[7].

[7] The reasoning is as follows: both $\mathcal{H}_1$ and $\mathcal{H}_2$ can be considered as models for $E[y|x, z_1]$ or $E[y|x, z_1, z_2]$, with $\mathcal{H}_1$ being the constrained version of $\mathcal{H}_2$ subject to the hard-order constraints that parameters associated with z_2 are zero. Since the hypothesis that minimizes the prediction error also provides the best approximation to the CEF – which is what we want, i.e., $\min_h E\left[(y-h)^2\right] = \min_h E\left[(E[y|x, z_1] - h)^2\right]$, the problem is essentially a variable selection problem.

Discussion: How Many Variables to Control?

2. In practice, adding more control variables means we have to integrate over higher dimensions to obtain $E\left[y|\text{do}\left(x\right)\right]$ (see (11)). Doing so increases the variance of the estimator in finite sample[8].

[8]If our goal is to obtain an estimate of the average causal effect, and if, in choosing control variables, we restrict our models to a class of additive models:

$$E\left[y|x,z\right]=\phi_1\left(x\right)+\phi_2\left(z\right)$$

, then by (5), we have:

$$\text{ATE}\left(x\right)=\frac{\partial E\left[y|x,z\right]}{\partial x}$$

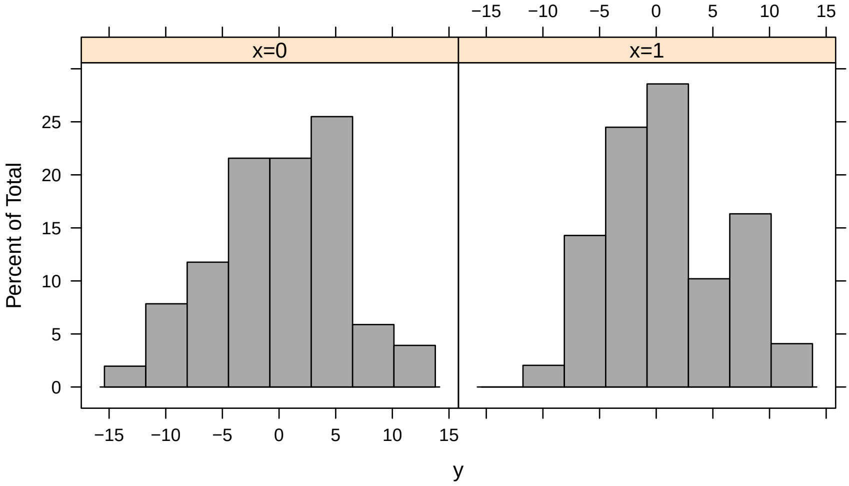
In this case, no integration is necessary to obtain the ATE, so we are left with only consideration 1.

Simulation 7

$$
\begin{aligned}
z_1 &\sim \mathcal{N}(0,1) \\
z_2 &\sim \mathcal{N}(0,1) \\
x &\leftarrow \text{Bernoulli}\left((1+\exp(-z_1))^{-1}\right) \\
y &\leftarrow 0.5x + z_1 + 5z_2 + \mathcal{N}(0, 0.01)
\end{aligned}
$$

```r
# Simulation
n = 100
z1 = rnorm(n)
z2 = rnorm(n)
x = rbinom(n,1,sigmoid(z1))
y = 0.5*x + z1 + 5*z2 + 0.1*rnorm(n)
```

Simulation 7



Simulation 7

```
fit = lm(y ~ x + z1)
coeftest(fit)

##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.58557    0.76987  0.7606 0.448737
## x            -0.60867    1.17055 -0.5200 0.604257
## z1            1.91355    0.62215  3.0757 0.002729 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulation 7

```
fit = lm(y ~ x + z1 + z2)
coeftest(fit)

##
## t test of coefficients:
##
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -0.0237201  0.0144846  -1.6376   0.1048
## x            0.5265737  0.0220585  23.8717   <2e-16 ***
## z1           0.9762877  0.0118034  82.7125   <2e-16 ***
## z2           5.0071967  0.0095362 525.0749   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulation 8

$$\begin{aligned} z_1, \ldots, z_{60} &\sim \mathcal{N}(0,1) \\ x &\leftarrow \text{Bernoulli}\left(\left(1+\exp\left(-z_1\right)\right)^{-1}\right) \\ y &\leftarrow x + \alpha' z + \mathcal{N}(0,1) \end{aligned}$$

, where $\alpha_1, \ldots, \alpha_{20} \sim U(0.5, 1)$, $\alpha_{21}, \ldots, \alpha_{40} \sim U(0, 0.3)$, $\alpha_{41}, \ldots, \alpha_{60} = 0$, $\alpha = (\alpha_1, \ldots, \alpha_{99})$, and $z = (z_1, \ldots, z_{60})$.

Simulation 8

```r
# Simulation
n = 1000 #number of data points
p = 60 #number of z variables
z = matrix(rnorm(n*p),nrow=n)
x = rbinom(n,1,sigmoid(z[,1]))
beta = c(runif(20,0.5,1),runif(20,0,0.3),rep(0,20))
y = x + z%*%beta + rnorm(n)

# Preparing Training and Test Sets
require(dplyr)
data = data.frame(x=x,z=z,y=y)
data_train = data %>% sample_frac(0.1)
data_test = data %>% setdiff(data_train)
Xr <- model.matrix(y ~.,data_train)[,-1]
Xe <- model.matrix(y ~.,data_test)[,-1]
yr = data_train$y
ye = data_test$y
```

Simulation 8

```
#####################
# Linear Regression #
#####################
# Here z1 is the only confounder
# so minimally, we need to control for z1
fit = lm(y ~ x + z.1, data_train)

# test err
yhat = predict(fit,data_test)
mean((yhat-ye)^2)

## [1] 12.24483
```

Simulation 8

```
coeftest(fit)

##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.63950    0.55225 -1.1580  0.24971
## x            1.38262    0.80823  1.7107  0.09034 .
## z.1          0.74650    0.43417  1.7194  0.08873 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

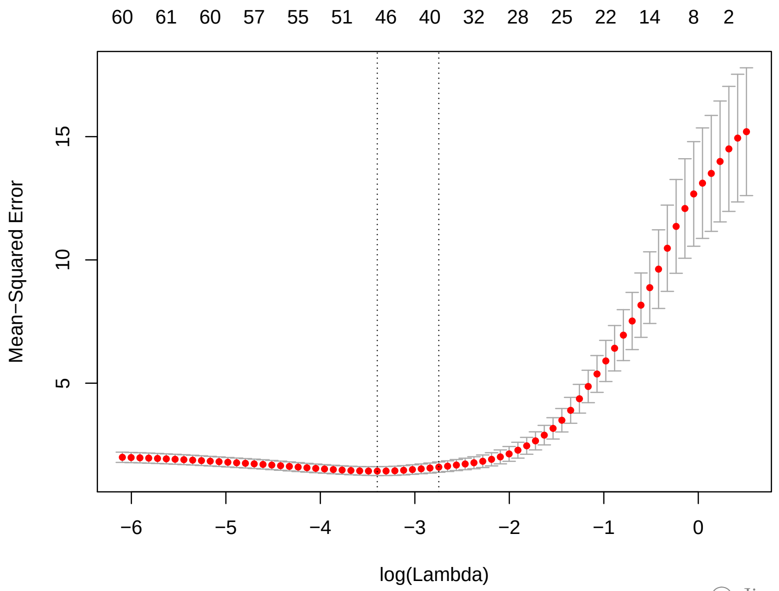
Simulation 8

```
######################
# Linear Regression #
######################
# regress on all z
fit = lm(y ~ ., data_train)
coeftest(fit)[2,] # coefficient of x

##     Estimate   Std. Error      t value     Pr(>|t|)
## 1.1705929767 0.2961487071 3.9527202005 0.0003248338

# test err
yhat = predict(fit,data_test)
mean((yhat-ye)^2)

## [1] 2.2757
```

Simulation 8

```
#########
# Lasso #
#########
require(glmnet)
fit.cv = cv.glmnet(Xr,yr,alpha=1)
fit = glmnet(Xr,yr,alpha=1,lambda=fit.cv$lambda.min)

nrow(predict(fit,type="nonzero")) # number of nonzero coefficients

## [1] 46

# test err
yhat = predict(fit,Xe)
mean((yhat-ye)^2)

## [1] 1.863265
```

Simulation 8



Simulation 8

```
coef(fit)[2] # coefficient of x
```

```
## [1] 1.070317
```

```
# Bootstrap confidence intervals for lasso coefficients
require(HDCI)
r = bootLasso(Xr,yr) # residual bootstrap lasso
```

```
r$interval[,1] # 95%CI for coefficient of x
```

```
## [1] 0.8188214 1.4670778
```

Discussion: Model Selection in Causal Inference

- The examples show that estimates of the causal effect of x on y using the back-door criterion can suffer from significant inaccuracy if (a) we do not sufficiently control for confounders z, or (b) we do not accurately model $E[y|x,z]$.
- Traditional econometrics focus on (a) and pay little attention to (b). In addition, by relying on models that are typically linear in x, traditional econometrics can be thought of focusing only on first-order effects.

Discussion: Bias-Variance Tradeoff in Causal Inference

- Should our goal be obtaining the most accurate causal effect estimates or unbiased causal effect estimates?

- There exists a popular narrative that the goal of machine learning is prediction, hence machine learning methods are not applicable to causal inference. However, as we have discussed, causal effect estimation can be thought of as a type of prediction as well, which we have called causal prediction – predicting the effect of $\text{do}(x)$. Obtaining the best causal effect estimate *should* mean obtaining the most *accurate* causal prediction.

Discussion: Bias-Variance Tradeoff in Causal Inference

- Modern machine learning methods explore the bias-variance tradeoff to improve the predictive performance of statical models. In causal inference – in particular, in the field of econometrics today, whether we should explore the same bias-variance tradeoff in causal effect estimation remains a topic of discussion.

- Part of the reason that the bias-variance tradeoff has not been emphasized in econometrics is that traditionally, statistical methods are evaluated based on their asymptotic properties, while the bias-variance tradeoff exists only in finite samples. Focusing on finite-sample performance and using the training-validation-test approach for model selection is one of the main innovations of modern machine learning.