Regresion and ATE

Tomoya Sasaki

May 26, 2020

```r
library(tidyverse)
```

Binary treatment, no control variables

Constant additive unit causal effect

- The assumption: the causal effect is the same across all the units
- $\beta = \mathbb{E}[Y_i(1)] - \mathbb{E}[Y_i(0)]$ for $\forall i$
- Data generating process

$$Y_i = \alpha + \beta D_i + \epsilon_i \tag{1}$$

- Treatment variable D_i, $\mathbb{E}[Y_i(0)] = \alpha$, and $\mathbb{E}[\epsilon_i] = 0$
- Note that we have α so $\mathbb{E}[\epsilon_i] = 0$ is not really an assumption, but by construction

Simulation Setting

```r
N <- 1000
M <- 1000
```

Generate data

```r
set.seed(123)
b <- 2 # treatment effect (constant across all units)
e <- rnorm(mean = 0, sd = 4, n = N)
# potential outcomes
y1 <- b + e
y0 <- e
# treatment asignment
x <- sample(0:1, prob = c(1, 1), size = N, replace = TRUE)
# observed outcomes
y <- ifelse(x == 1, y1, y0)
```

Estimate via regression

```r
mod1 <- lm(y ~ x)
coefficients(summary(mod1))
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.1949345  0.1784689 1.092260 2.749824e-01
## x           1.7422470  0.2508923 6.944203 6.850761e-12
```

Estimate via difference in means

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

```
## [1] 1.742247
```

Regression VS difference in means

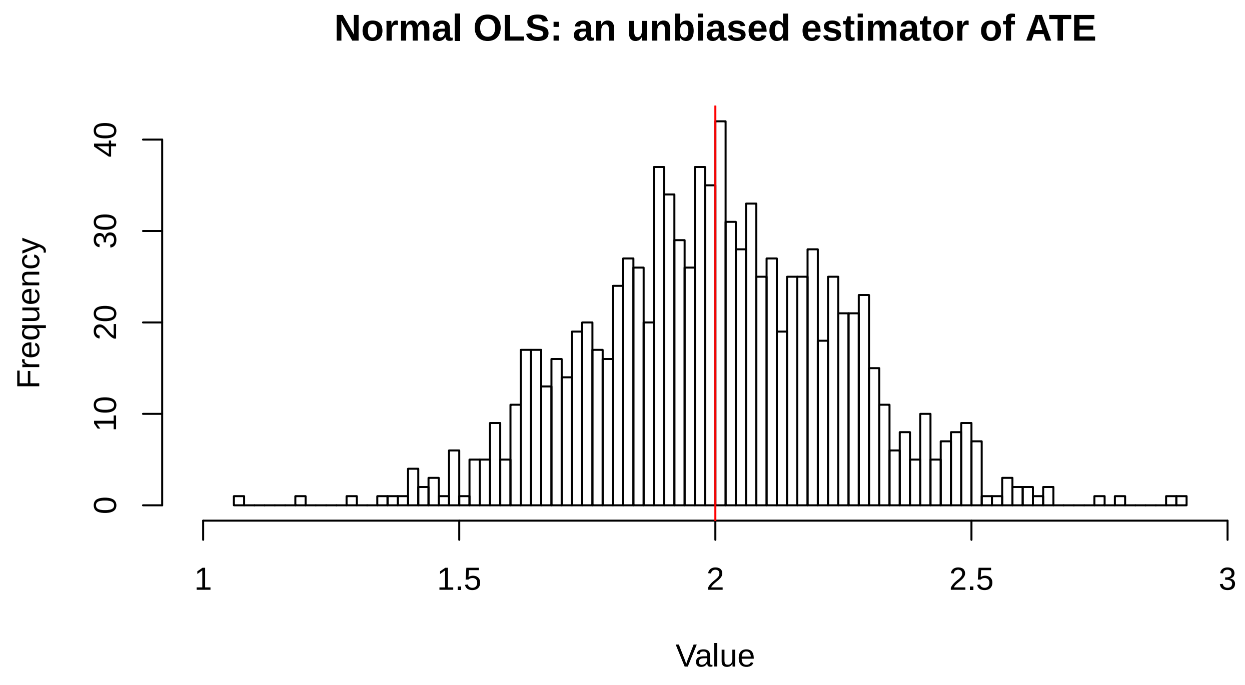
- In the simulation, I only change the treatment assignment
- Results of two estimators are identical
- A regression estimator is a unbiased estimator of the difference in means estimator as long as the treatment varaiable is binary

```r
# create function
sim_reg_diff_mean <- function(.y1, .y0) {
  # random assignment of treatment
  x <- sample(0:1, prob = c(1, 1), size = N, replace = TRUE)
  # observed y
  y <- ifelse(x == 1, .y1, .y0)
  # regression
  mod <- lm(y ~ x)
  # diff-in-means
  dif <- mean(y[x == 1]) - mean(y[x == 0])
  return( c(mod$coef[2], dif) )
}

set.seed(123)
# simulation
# use replicate and generated data above
res <- replicate(n = M, sim_reg_diff_mean(y1, y0))
# two estimators are the same
all.equal(res[1, ], res[2, ])
```

```
## [1] TRUE
```

Plot results

```r
xup <- 3
xdown <- 1
{
  hist(res[1, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown, xup), xaxt = "n")
  axis(1, at = seq(xdown, xup, by = 0.5), labels = seq(xdown, xup, by = 0.5))
  abline(v = 2, col = "red")
  title(main = "Normal OLS: an unbiased estimator of ATE")
}
```



Calculate RMSE

```r
rmse1 <- apply(res, 1, function(x) sqrt(mean((x - 2)^2)))
```

Treatment effect heterogeneity

- Allow for heterogeneous treatment effects: each unit has different causal effects
- $\beta + \epsilon_i(1) - \epsilon_i(0) = \mathbb{E}[Y_i(1)] - \mathbb{E}[Y_i(0)]$ for $\forall i$
- β is ATE
- Data generating process

$$Y_i = \alpha + \beta D_i + \epsilon_i(D_i) \tag{2}$$

- Treatment variable D_i, $\mathbb{E}[Y_i(0)] = \alpha$, and $\mathbb{E}[\epsilon_i(1)] = \mathbb{E}[\epsilon_i(0)] = 0$
- Relax the assumption of $\epsilon_i = \epsilon_i(1) = \epsilon_i(0)$
- The example below shows that the OLS estimator is an unbiasesd estimator of the ATE.

Generate data

```r
set.seed(123)
# this time with different error term structure
e1 <- rnorm(mean = 0, sd = 4, n = N)
e0 <- rnorm(mean = 0, sd = 4, n = N)
# potential outcomes
y1 <- b + e1
y0 <- e0
# treatment asignment
x <- sample(0:1, prob = c(1, 1), size = N, replace = TRUE)
# observed outcomes
y <- ifelse(x == 1, y1, y0)
```

Estimate via difference in means

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

```
## [1] 2.099449
```

Estimate via regression

```r
mod1 <- lm(y ~ x)
coefficients(summary(mod1))
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 0.09649083  0.1780082 0.5420583 5.878993e-01
## x           2.09944879  0.2490173 8.4309347 1.188262e-16
```

Regression VS difference in means under heterogenous traetment effects

- Results of two estimators are identical
- Again, a regression estimator is a unbiased estimator of the difference in means estimator as long as the treatment varaiable is binary

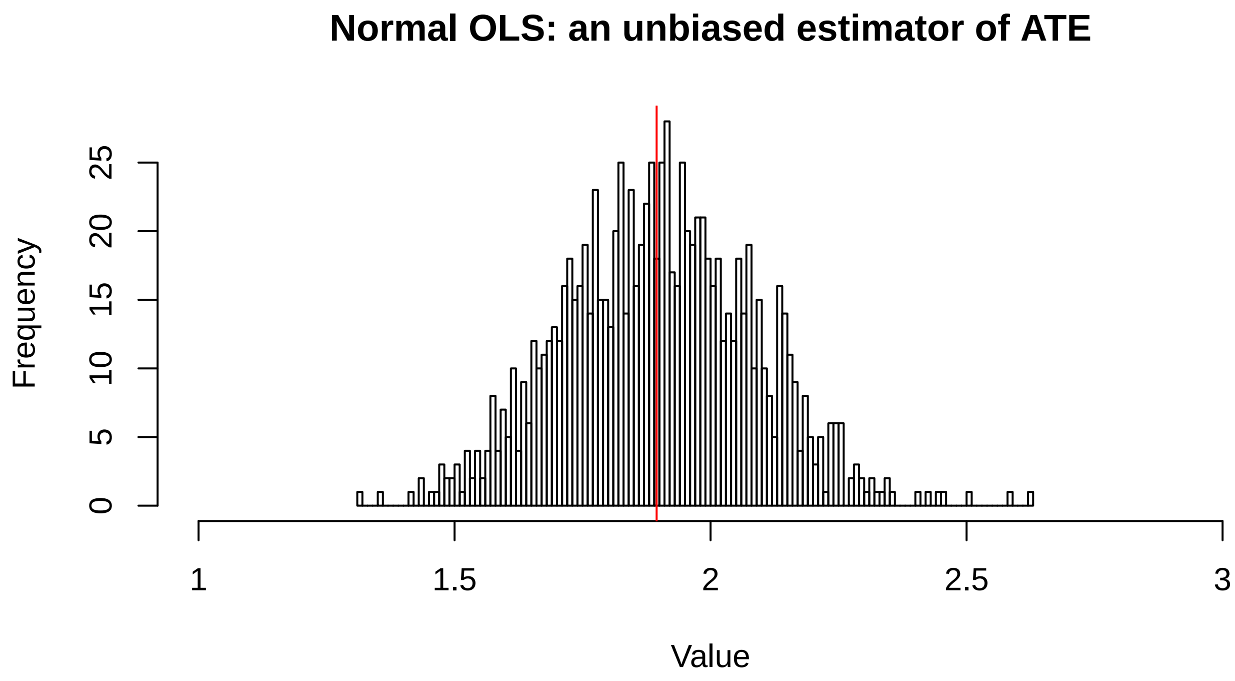
```r
set.seed(123)
# simulation
# use replicate and generated data
res <- replicate(n = M, sim_reg_diff_mean(y1, y0))
# two estimators are the same
all.equal(res[1, ], res[2, ])
```

```
## [1] TRUE
```

Plot results

- Note that this time true ATE is not 2 because of heterogeneity

```r
xup <- 3
xdown <- 1
{
  hist(res[1, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown, xup), xaxt = "n")
  axis(1, at = seq(xdown, xup, by = 0.5), labels = seq(xdown, xup, by = 0.5))
  abline(v = mean(y1 - y0), col = "red")
  title(main = "Normal OLS: an unbiased estimator of ATE")
}
```



Calculate RMSE

```
rmse2 <- apply(res, 1, function(x) sqrt(mean((x - mean(y1 - y0))^2)))
```

Binary treatment, with control variables

Heterogeneous treatment effect

- If you assume constant treatment effect, you can just regression with control variables
- However, if you assume heterogeneous treatment effect, you need to use a saturated model:

$$Y_i = \sum_x B_{xi}\gamma_x + \beta D_i + \epsilon_i \tag{3}$$

where B_{xi} is a dummy variable for unique combination of x_i.
- We compare results from a saturated model with results from a normal OLS:

$$Y_i = \gamma^\top x_i + \beta D_i + e_i \tag{4}$$

- This model is linear in covariates by construction whereas the linear assumption does not necessarily holds for a normal OLS

Generate data

```
set.seed(123)
b1 <- 2
# control variables
lenx2 <- 5 # 1 to 5
lenx3 <- 4 # 1 to 3
```

```r
lenx4 <- 3 # 1 to 4
lenx5 <- 4 # 1 to 4
# probability of covariates
probx2 <- sample(1:5, size = lenx2, replace = TRUE)
probx3 <- sample(1:5, size = lenx3, replace = TRUE)
probx4 <- sample(1:5, size = lenx4, replace = TRUE)
probx5 <- sample(1:5, size = lenx5, replace = TRUE)
# generate covariates
x2 <- sample(1:lenx2, prob = probx2, size = N, replace = TRUE)
x3 <- sample(1:lenx3, prob = probx3, size = N, replace = TRUE)
x4 <- sample(1:lenx4, prob = probx4, size = N, replace = TRUE)
x5 <- sample(1:lenx5, prob = probx5, size = N, replace = TRUE)
raw_mat <- cbind(x2, x3)
# different error term structure
e1 <- rnorm(mean = 0, sd = 4, n = N)
e0 <- rnorm(mean = 0, sd = 4, n = N)
```

Prepare to run saturated model: create dummy variables

```r
# possible combination of control variables
comb <- expand.grid(1:lenx2, 1:lenx3)#, 1:lenx4)#, 1:lenx5)
colnames(comb) <- c("x2", "x3")#, "x4")#, "x5")
# create unique ID for each combination
comb <- cbind(comb, id = 1:nrow(comb))
# control variables for each observation is mapped to IDs
comb_join <- inner_join(data.frame(x2 = x2, x3 = x3), data.frame(comb),
                        by = c("x2" = "x2", "x3" = "x3"))
# create a matrix of dummy variables
matXs <- matrix(0, nrow = N, ncol = nrow(comb))

for (i in 1:nrow(comb_join)){
    .id <- comb_join$id[i]
    matXs[i, .id] <- 1
}
```

Pattern 1

- Potential outcomes: linear

```r
# potential outcomes
y1 <- e1 + b1 + 3 * x2 + -2 * x3# + b4 * x4 # + b5 * x5
y0 <- e0 + 3 * x2 + -2 * x3# + b4 * x4 # + b5 * x5

# random assignment of treatment
# p <- apply(raw_mat, 1, sum)/max(apply(raw_mat, 1, sum) + 1)
p <- 1 / (1 + exp(raw_mat[, 1] - 0.5 * raw_mat[, 2] + 0.25))
x <- as.numeric(runif(length(p)) < p)
df <- data.frame(x = x, matXs)
# observed y
y <- ifelse(x == 1, y1, y0)
# regression
mod <- lm(y ~ . -1, df); coef(mod)[1]
```

```
##         x
## 2.327662
```

```r
# normal OLS
df <- data.frame(x = x, raw_mat)
mod2 <- lm(y ~ ., df); coef(mod2)[2]
```

```
##         x
## 2.249546
```

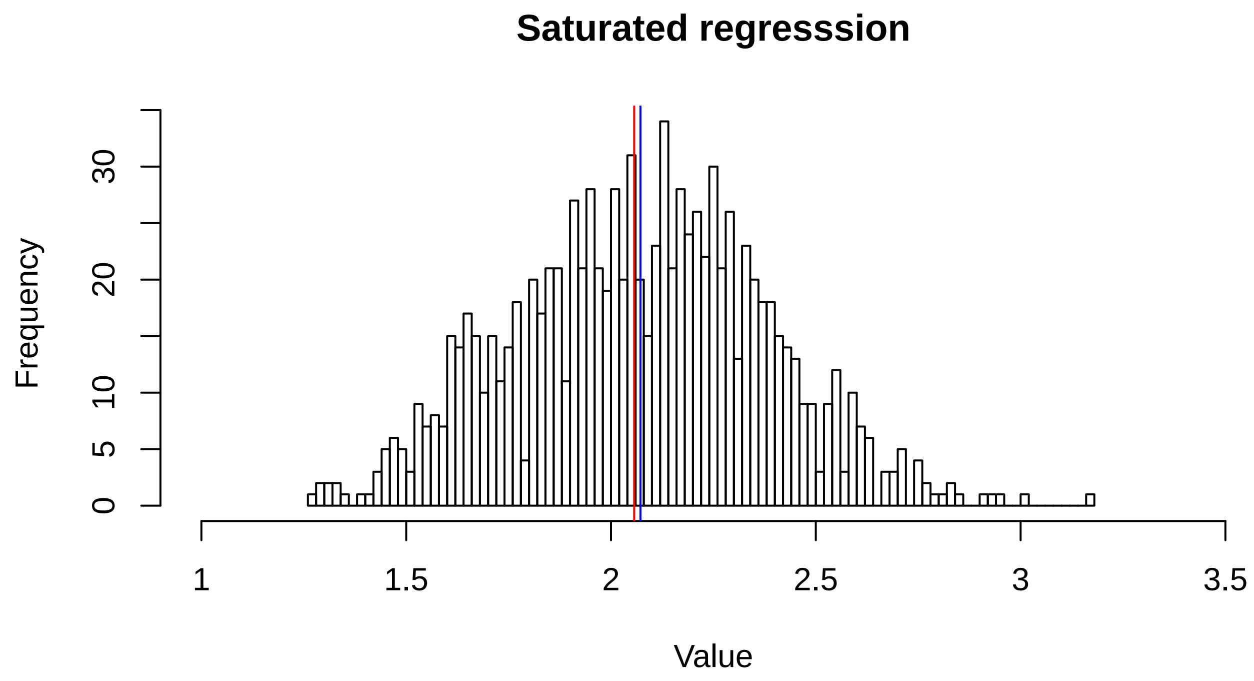
```r
mean(y1 - y0)
```

```
## [1] 2.056572
```

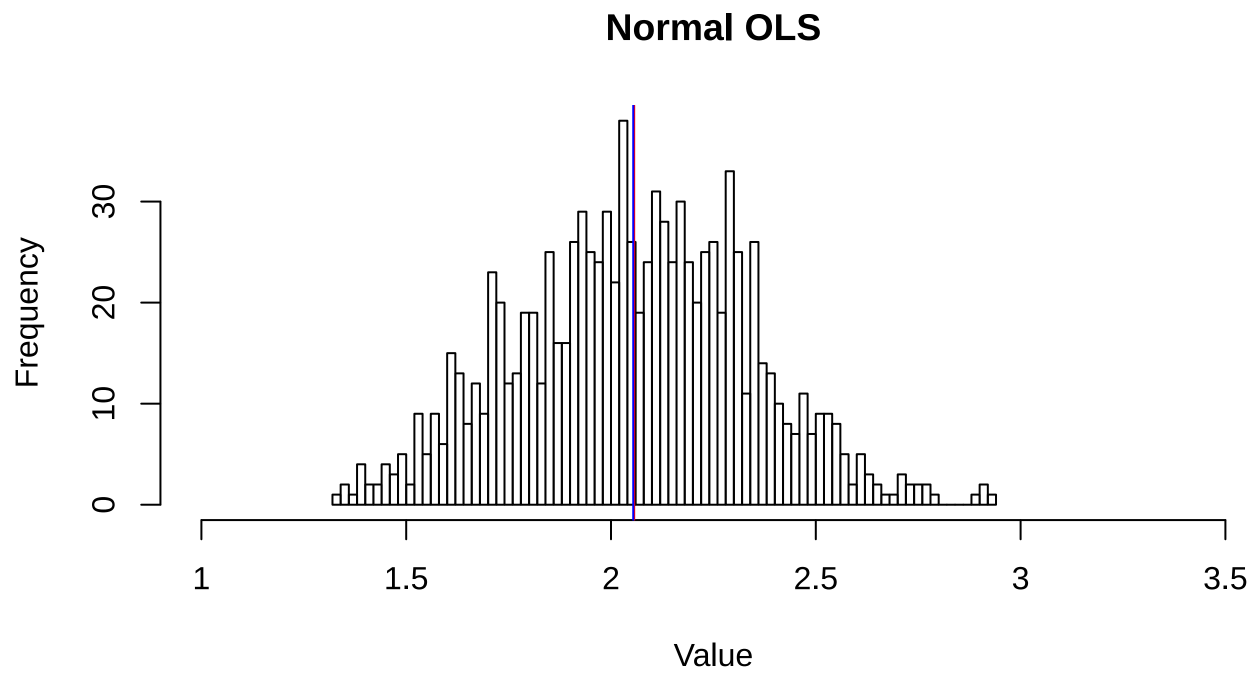
```r
# create function
sim_reg_saturated <- function(.y1, .y0, .mat, .N, .raw) {
  # random assignment of treatment
  # .p <- apply(.raw, 1, sum)/max(apply(.raw, 1, sum) + 1)
  .p <- 1 / (1 + exp(.raw[, 1] - 0.5 * .raw[, 2] + 0.25))
  x <- as.numeric(runif(length(.p)) < .p)
  df <- data.frame(x = x, .mat)
  # observed y
  y <- ifelse(x == 1, .y1, .y0)
  # satruated regression
  mod <- lm(y ~ . -1, df)

  # normal regression
  df2 <- data.frame(x = x, .raw)
  mod2 <- lm(y ~ ., df2)

  return( c(mod$coef[1], mod2$coef[2]) )
}

set.seed(123)
# simulation
# use replicate and generated data above
res <- replicate(n = M, sim_reg_saturated(y1, y0, matXs, N, raw_mat))
```

- Plot results
- Note that this time true ATE is not 2 because of heterogeneity

```r
xup <- 3.5
xdown <- 1
{
  hist(res[1, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown , xup), xaxt = "n")
  axis(1, at = seq(xdown , xup, by = 0.5), labels = seq(xdown , xup, by = 0.5))
  abline(v = mean(y1 - y0), col = "red") # true ATE is not 2
  abline(v = mean(res[1, ]), col = "blue")
  title(main = "Saturated regresssion")
}
```



```r
{
  hist(res[2, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown , xup), xaxt = "n")
  axis(1, at = seq(xdown , xup, by = 0.5), labels = seq(xdown , xup, by = 0.5))
  abline(v = mean(y1 - y0), col = "red") # true ATE is not 2
  abline(v = mean(res[2, ]), col = "blue")
  title(main = "Normal OLS")
}
```



- Calculate RMSE

```r
rmse3 <- apply(res, 1, function(x) sqrt(mean((x - mean(y1 - y0))^2)))
```

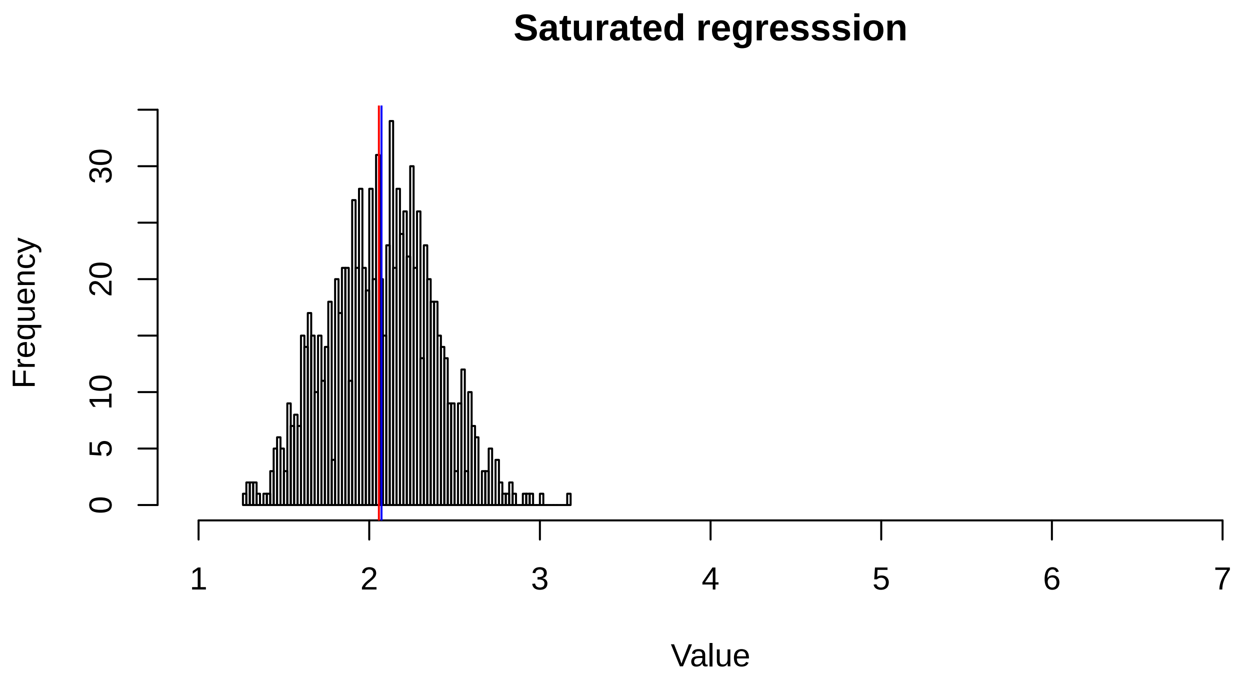
Generate data 2

- Potential outcomes: nonlinear

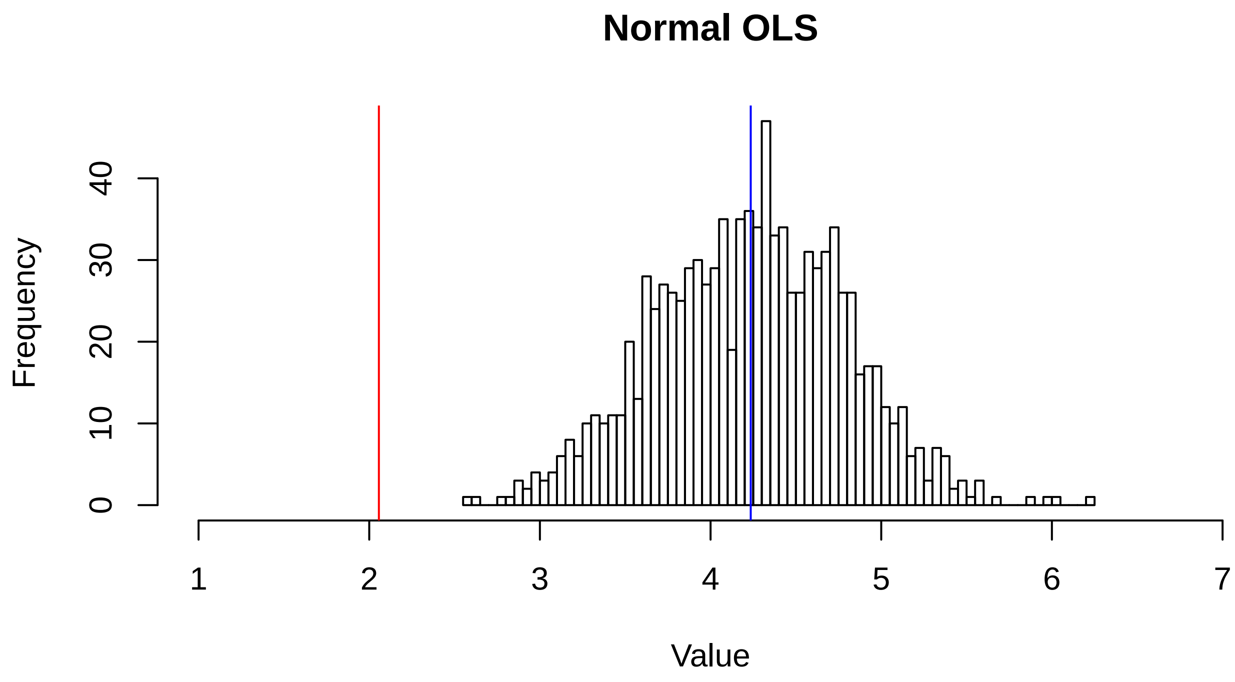
```r
# potential outcomes
y1 <- e1 + b1 + 3 * x2^2 + -2 * 1/x3# + b4 * x4 # + b5 * x5
y0 <- e0 + 3 * x2^2 -2 * 1/x3# + b4 * x4 # + b5 * x5

set.seed(123)
# simulation
# use replicate and generated data above
res <- replicate(n = M, sim_reg_saturated(y1, y0, matXs, N, raw_mat))
```

- Plot results

```r
xup <- 7
xdown <- 1
{
  hist(res[1, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown, xup), xaxt = "n")
  axis(1, at = seq(xdown, xup, by = 1), labels = seq(xdown, xup, by = 1))
  abline(v = mean(y1 - y0), col = "red") # true ATE is not 2
  abline(v = mean(res[1, ]), col = "blue")
  title(main = "Saturated regresssion")
}
```



```r
{
  hist(res[2, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown , xup), xaxt = "n")
  axis(1, at = seq(xdown , xup, by = 1), labels = seq(xdown , xup, by = 1))
  abline(v = mean(y1 - y0), col = "red") # true ATE is not 2
  abline(v = mean(res[2, ]), col = "blue")
  title(main = "Normal OLS")
}
```



- Calculate RMSE

```
rmse4 <- apply(res, 1, function(x) sqrt(mean((x - mean(y1 - y0))^2)))
```

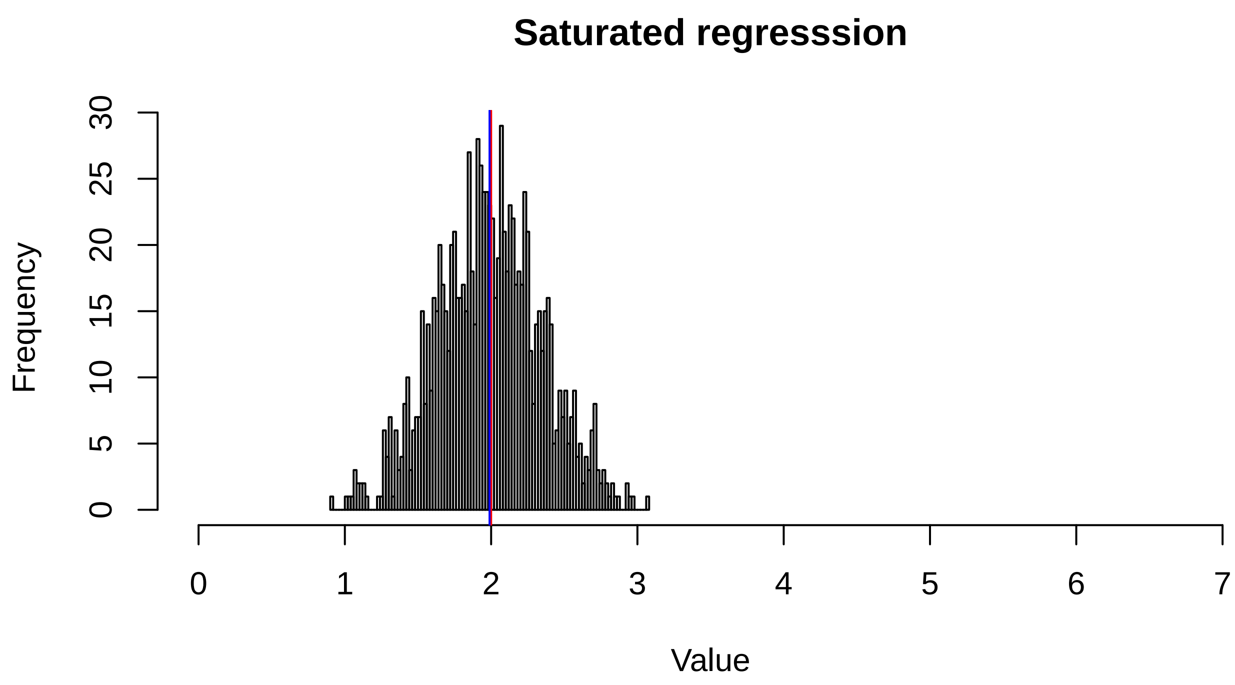
Constant effect revisited

- Potential outcomes: nonlinear
- But treatment effect is constant

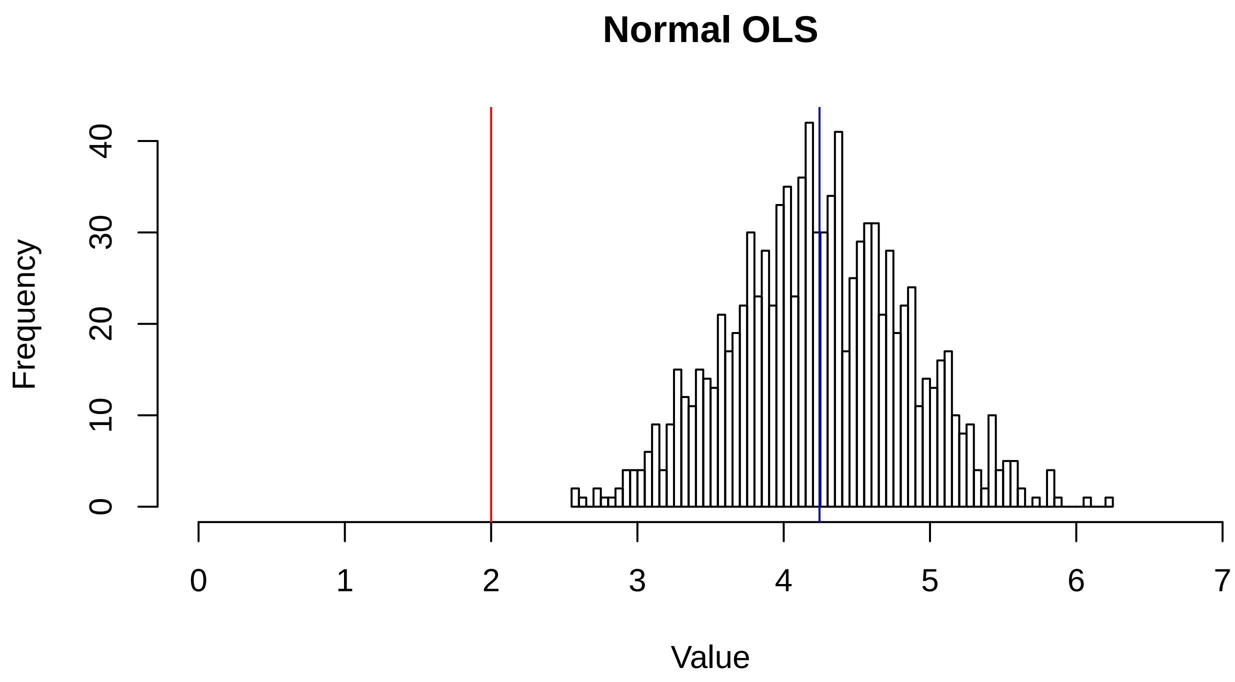
```
# potential outcomes
set.seed(123)
e <- rnorm(mean = 0, sd = 4, n = N)
y1 <- e + b1 + 3 * x2^2 + -2 * 1/x3# + b4 * x4 # + b5 * x5
y0 <- e + 3 * x2^2 + -2 * 1/x3# + b4 * x4 # + b5 * x5

set.seed(123)
# simulation
# use replicate and generated data above
res <- replicate(n = M, sim_reg_saturated(y1, y0, matXs, N, raw_mat))
```

- Plot results

```
xup <- 7
xdown <- 0
{
  hist(res[1, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown, xup), xaxt = "n")
  axis(1, at = seq(xdown, xup, by = 1), labels = seq(xdown, xup, by = 1))
  abline(v = mean(y1 - y0), col = "red") # true ATE is not 2
  abline(v = mean(res[1, ]), col = "blue")
  title(main = "Saturated regresssion")
}
```



```r
{
  hist(res[2, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown, xup), xaxt = "n")
  axis(1, at = seq(xdown, xup, by = 1), labels = seq(xdown , xup, by = 1))
  abline(v = mean(y1 - y0), col = "red") # true ATE is not 2
  abline(v = mean(res[2, ]), col = "blue")
  title(main = "Normal OLS")
}
```



- Calculate RMSE

```r
rmse7 <- apply(res, 1, function(x) sqrt(mean((x - mean(y1 - y0))^2)))
```

Saturated model when N is small

- Note that a saturatede model has more predictors than a model with the same set of control variables so the performance could be worse when the sample size is small
- The example below shows the same simulation with different number of units

```r
N <- 100

set.seed(123)
b1 <- 2
# control variables
lenx2 <- 5 # 1 to 5
lenx3 <- 4 # 1 to 3
lenx4 <- 3 # 1 to 4
lenx5 <- 3 # 1 to 4
# probability of covariates
probx2 <- sample(1:5, size = lenx2, replace = TRUE)
probx3 <- sample(1:5, size = lenx3, replace = TRUE)
probx4 <- sample(1:5, size = lenx4, replace = TRUE)
probx5 <- sample(1:5, size = lenx5, replace = TRUE)
# generate covariates
x2 <- sample(1:lenx2, prob = probx2, size = N, replace = TRUE)
x3 <- sample(1:lenx3, prob = probx3, size = N, replace = TRUE)
x4 <- sample(1:lenx4, prob = probx4, size = N, replace = TRUE)
x5 <- sample(1:lenx5, prob = probx5, size = N, replace = TRUE)
raw_mat <- cbind(x2, x3)
# different error term structure
e1 <- rnorm(mean = 0, sd = 4, n = N)
e0 <- rnorm(mean = 0, sd = 4, n = N)
```

Prepare to run saturated model: create dummy variables

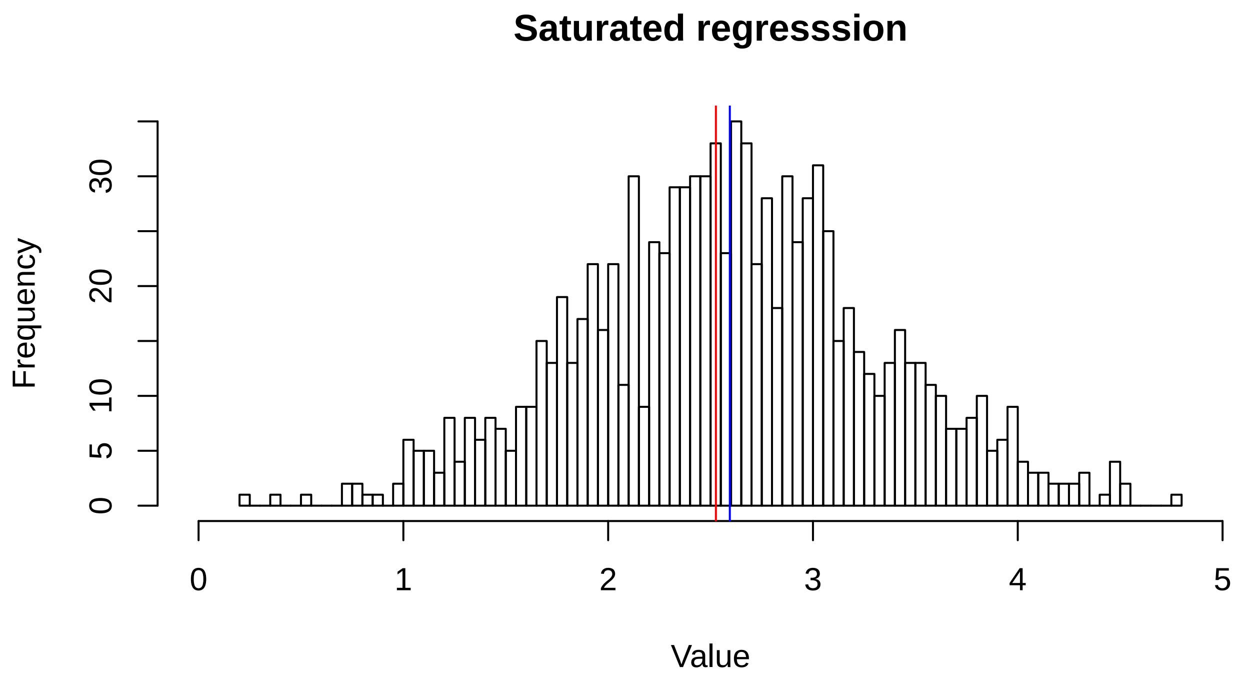
```r
# possible combination of control variables
comb <- expand.grid(1:lenx2, 1:lenx3)#, 1:lenx4)#, 1:lenx5)
colnames(comb) <- c("x2", "x3")#, "x4")#, "x5")
# create unique ID for each combination
comb <- cbind(comb, id = 1:nrow(comb))
# control variables for each observation is mapped to IDs
comb_join <- inner_join(data.frame(x2 = x2, x3 = x3), data.frame(comb),
                        by = c("x2" = "x2", "x3" = "x3"))
# create a matrix of dummy variables
matXs <- matrix(0, nrow = N, ncol = nrow(comb))

for (i in 1:nrow(comb_join)){
    .id <- comb_join$id[i]
    matXs[i, .id] <- 1
}
```

- Probability of receiving treatment: linear
- Potential outcomes: linear

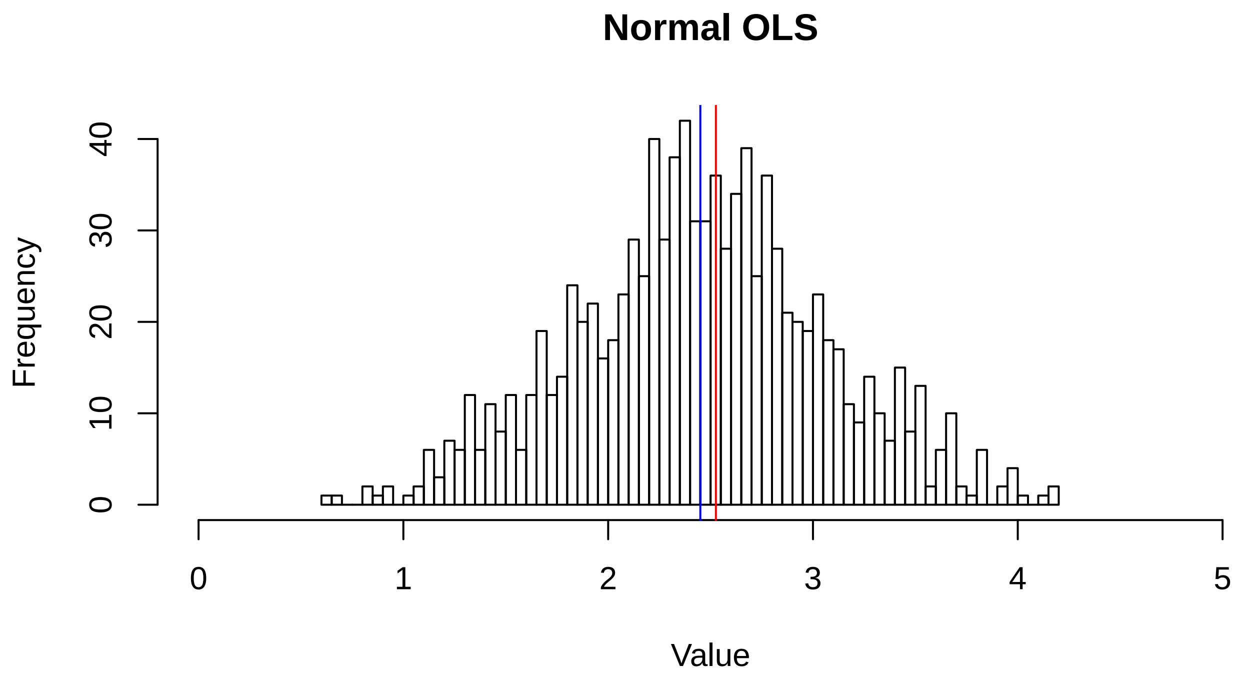
```r
# create function
sim_reg_saturated <- function(.y1, .y0, .mat, .N, .raw) {
  # random assignment of treatment
  .p <- apply(.raw, 1, sum)/max(apply(.raw, 1, sum) + 1)
  x <- as.numeric(runif(length(.p)) < .p)
  df <- data.frame(x = x, .mat)
  # observed y
  y <- ifelse(x == 1, .y1, .y0)
  # satruated regression
  mod <- lm(y ~ . -1, df)

  # normal regression
  df2 <- data.frame(x = x, .raw)
  mod2 <- lm(y ~ ., df2)

  return( c(mod$coef[1], mod2$coef[2]) )
}

# potential outcomes
y1 <- e1 + b1 + 3 * x2 + -2 * x3# + b4 * x4 # + b5 * x5
y0 <- e0 + 3 * x2 + -2 * x3# + b4 * x4 # + b5 * x5

set.seed(123)
# simulation
# use replicate and generated data above
res <- replicate(n = M, sim_reg_saturated(y1, y0, matXs, N, raw_mat))
```

- Plot results
- Note that this time true ATE is not 2 because of heterogeneity

```r
xup <- 5
xdown <- 0
{
  hist(res[1, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown , xup), xaxt = "n")
  axis(1, at = seq(xdown , xup, by = 1), labels = seq(xdown , xup, by = 1))
  abline(v = mean(y1 - y0), col = "red") # true ATE is not 2
  abline(v = mean(res[1, ]), col = "blue")
  title(main = "Saturated regresssion")
}
```



```
{
  hist(res[2, ], breaks = 100, xlab = "Value", main = "", xlim = c(xdown, xup), xaxt = "n")
  axis(1, at = seq(xdown, xup, by = 1), labels = seq(xdown , xup, by = 1))
  abline(v = mean(y1 - y0), col = "red") # true ATE is not 2
  abline(v = mean(res[2, ]), col = "blue")
  title(main = "Normal OLS")
}
```



- Calculate RMSE

```r
rmse8 <- apply(res, 1, function(x) sqrt(mean((x - mean(y1 - y0))^2)))
```

Compare RMSE for models with multiplc control variables

```r
rmses <- rbind(rmse3, rmse4, rmse7, rmse8)
rmses <- data.frame(c("linear", "nonlinear", "nonlinear", "linear"), rmses)
rownames(rmses) <- c("Pattern 1", "Pattern 2", "Constant Effect", "Small N")
colnames(rmses) <- c("Potential Outcomes", "Saturated Model", "Normal OLS")
knitr::kable(rmses)
```

	Potential Outcomes	Saturated Model	Normal OLS
Pattern 1	linear	0.3084701	0.2834243
Pattern 2	nonlinear	0.3084701	2.2488918
Constant Effect	nonlinear	0.3578689	2.3252215
Small N	linear	0.7330728	0.6174979