Lecture 5: Simulation and Propensity Scores

Intro to Data Science for Public Policy, Spring 2016

by Jeff Chen & Dan Hammer, Georgetown University McCourt School of Public Policy

Contents

The objective of any good data science is to avoid work. Any work. And especially math. Don't spend time working out a closed-form solution when you can simulate the answer just as easily. Consider, for example, trying to calculate the standard errors on a complicated regression model? It is far easier to simulate the answer, rather than calculate it directly.

This lecture will offer an introduction to simulation in `R` and a cursory review of calculating a consistent estimate of policy impact, using simulation and propensity score matching. Entire books have been written about this subject, and this lecture doesn't come close to a full treatment. However, as an illustration of the value of simulation in data science, the propensity score applications are particularly expository.

Random number generator

Simulation is especially useful when complex interactions between random variables are at work. This is often the case in policy analysis, since the world is complicated and everything is related to everything else. We may not be able to calculate directly the behavior of a random process. Instead of an algebraic solution, we can set up a simulation and examine the outcomes. At it's core, then, a simulation depends on generating and using random variables.

Start with a random variable $X \sim N(0, 1)$, or a variable that is drawn from a normal distribution with mean 0 and standard deviation 1. We will draw from this distribution 1,000 times to create a vector `X`.

```
X <- rnorm(1000)
head(X)
```

```
[1]  0.87863683 -0.33234845  0.08551549 -0.93119338 -1.07885153  0.27847331
```

The mean of `X` should be close to zero:

```
mean(X)
```

```
[1] -0.04322571
```

If you run this over and over, you will get different values, since the random vector is generated over and over again.

```
X.1 <- rnorm(1000)
X.2 <- rnorm(1000)

mean(X.1)
```

```
[1] -0.05343959
```

```
mean(X.2)
```

```
[1] 0.02532748
```

Using truly random variables makes it difficult to figure out whether our code is actually doing what it is supposed to do. Is the difference the result of a typo or natural randomness. So, instead of a truly random

variable, we can peg the randomness to a reference value – creating a pseudo-random variable to use in our analysis. Use the `set.seed()` function to peg the draws to the same seed.

```
set.seed(124)
X.1 <- rnorm(1000)

set.seed(124)
X.2 <- rnorm(1000)

mean(X.1)
```

```
[1] -0.0653552
```

```
mean(X.2)
```

```
[1] -0.0653552
```

Note that the two random vectors have the same exact mean, even though they are two "random" pulls. Now we can be assured that any observed differences means that someone screwed up.

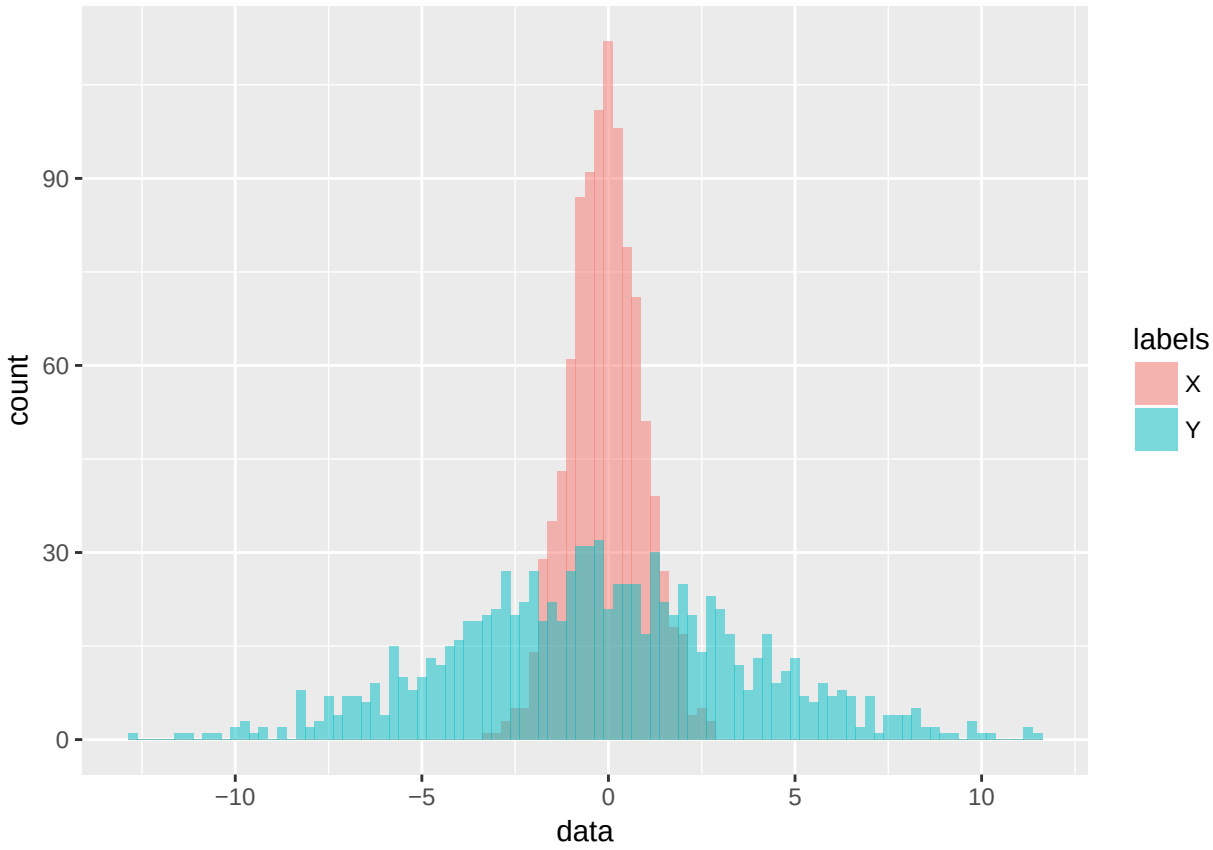
Notice that the spread of a random variable is indicated by the measure of standard deviation from the mean. Let's generate and plot the distributions of two normally distributed random variables with different standard deviations.

```
set.seed(124)
X <- rnorm(1000, 0, 1)

set.seed(124)
Y <- rnorm(1000, 0, 4)

# create labels for the data frame, first repeating "X" one thousand times,
# and then repeating "Y" one thousand times
labels.vec <- rep(c("X", "Y"), c(1000, 1000))
data <- data.frame(data=c(X, Y), labels=labels.vec)

# plot the histograms of each variable, where `alpha` is an opacity parameter.
library(ggplot2)
p <- ggplot(data, aes(x=data, fill=labels))
p + geom_histogram(position="identity", binwidth=0.25, alpha=0.5)
```



The `X` variable is distributed standard normal, while the `Y` variable is distributed normally with the same mean (zero) but a greater spread.

Now suppose you are asked to calculate the spread (or standard deviation) of a new random variable, generated from multiplying a random variable $X \sim N(0, 1)$ by 2 and adding 3. Call this new variable $T = 2X + 3$. You can calculate the answer directly, or you can get a quick estimate by drawing the composed variable thousands of times and observing the new distribution. First, let's calculate the variance directly, noting that $\text{var}(aX + b) = a^2\text{var}(X)$. Thus, the :

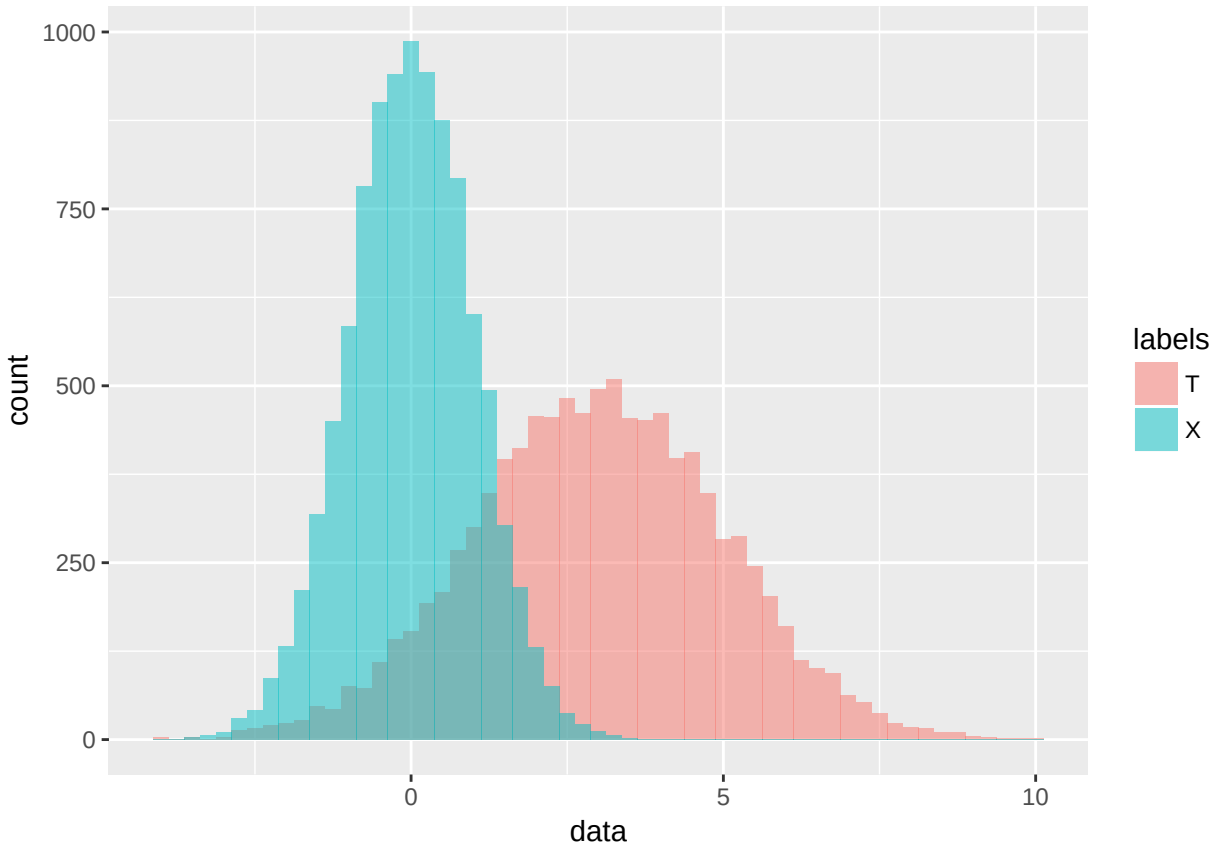
$$\text{sd}(T) = \sqrt{\text{var}(2X + 3)} = \sqrt{2^2\text{var}(X)} = \sqrt{4 \cdot 1} = 2 \tag{1}$$

The standard deviation of T is 2. This is too much work. You have to remember math, which is unacceptable.

```
set.seed(124)
X <- rnorm(10000, 0, 1)

T <- (2 * X) + 3
labels.vec <- rep(c("X", "T"), c(10000, 10000))
data <- data.frame(data=c(X, T), labels=labels.vec)

# plot the histograms of each variable, where `alpha` is an opacity parameter.
p <- ggplot(data, aes(x=data, fill=labels))
p + geom_histogram(position="identity", binwidth=0.25, alpha=0.5)
```



We can check that the simulated values are roughly equal to

```
sd(X)
```

```
[1] 0.9983922
```

```
sd(T)
```

```
[1] 1.996784
```

Impact assessment, understanding bias

Estimating the impact of a policy or program is not easy. Empirical economists spend lifetimes crafting new estimators and then debating their use in the wild. A data scientist may be able to short-circuit some of this work, however, by simulating outcomes. Much like the previous example, but with a **data generating process** that looks much more like a real-world application than simple draws from normal distributions.

A data generating process is a model of the random influences that yield observed outcomes. This model would include, for example, all of the random influences that led to the color of the shirt you put on this morning. The weather, your mood. In reality, we don't observe a data generating process. Instead, we attempt to infer some structure from meaures that we *can* observe, like gender, age, or hair color – all observable variables that may influence shirt color selection. We can use the structure to assess the impact of an exogenous change. Suppose a tax is imposed on wearing blue pants. What is the average impact of the blue-pants tax on the selection of red shirts?

Parsing structure from noise within social statistics is difficult – maybe impossible – with algebra. Simulation and slapdash coding can get you pretty far, fairly quickly. The objective of the following illustration is to demonstrate how simulation and a contrived data generating process can yield intuition that carries over to real life processes.

Suppose we are asked to estimate the impact of a policy D on an outcome variable Y. This could be the impact of charter schools on test scores or tax policy on clothing choices. Let D_i be the indicator of treatment for observation $i = 1, 2, \ldots, N$; let Y_i be the outcome variable; and let X_i be a vector of observable cofactors:

$$Y_i = \delta D_i + \beta X_i + \epsilon_i, \quad \text{with} \quad \epsilon_i \sim N(0,1) \tag{2}$$

Assume further that the three observable characteristics $x_1, x_2, x_3 \sim Unif(0,1)$ determine the propensity of receiving treatment:

$$D_i = \begin{cases} 1 & \text{if } 2(x_{1i} + x_{2i} + x_{3i}) + u_i > 4, \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $u_i \sim N(0,1)$ is just pure randomness. Note that if we run a linear regression without conditioning on the three observable cofactors, bundled in X_i, the estimated treatment effect will be biased. If we ignore these cofactors, the impact of these cofactors is falsely ascribed to the treatment, D. With this framework, we can construct a data set of size $N = 5000$ in order to examine the behavior of two estimation techniques: multiple regression and conditioning on propensity score.

```
N <- 5000; eps <- rnorm(N); u <- rnorm(N)
x1 <- runif(N); x2 <- runif(N); x3 <- runif(N)
D <- ifelse(2*(x1 + x2 + x3) + u > 4, 1, 0)
Y <- D + x1 + x2 + x3 + eps
summary(D); summary(Y)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   0.247   0.000   1.000

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -2.888   0.857   1.677   1.737   2.602   7.014
```

Roughly one quarter of the observations received treatment, and the outcome variable has about a ten unit spread, centered around 1.5 or 2.

For reference, we estimate to basic, linear models by ordinary least squares. First, we do not condition on the X covariates, which will yield biased estimates of the treatment effect. We bootstrap the distribution of the estimated treatment effect. We sample $n = 500$ observations from the distribution, estimate the impact effect, and repeat for $B = 5000$ iterations. Note that we do not iterate using a `for` loop, but rather by applying the `ols` function, defined below, to a range of indices using `sapply` to keep the code compact and readable.

```
n <- 500; B <- 5000
X <- cbind(1, D)

ols <- function(i) {
    idx <- sample.int(N,n)
    Xs <- X[idx,]
    b <- solve(t(Xs) %*% Xs) %*% t(Xs) %*% Y[idx]
    b[2]
}

res.ols <- data.frame(impact=sapply(1:B, ols), method=c("ols"))
```

Before we graph the distribution, let's perform the same process for the estimated impact, conditioning on X. This should yield a consistent estimator for the treatment effect δ, since by construction there is no three-way covariation between the error, outcome, *and* treatment, after conditioning on the observables.

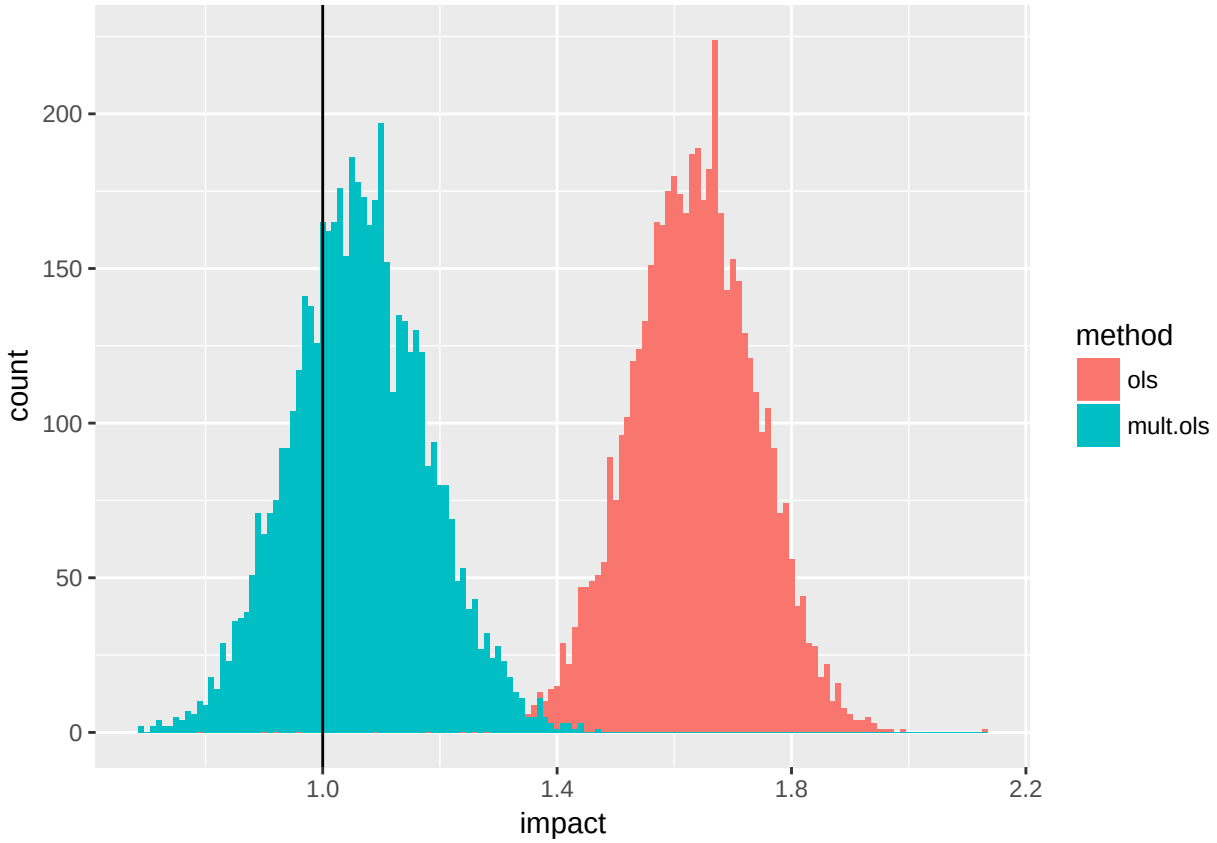
```
X.ext <- cbind(1, D, x1, x2, x3)

mult.ols <- function(i) {
    idx <- sample.int(N,n)
    Xs <- X.ext[idx,]
    b <- solve(t(Xs) %*% Xs) %*% t(Xs) %*% Y[idx]
    b[2]
}

res.mult <- data.frame(impact=sapply(1:B, mult.ols), method=c("mult.ols"))
total.res <- rbind(res.ols, res.mult)
```

Now we can plot the two distributions of impact estimates, based on the method of estimation. The vertical line in the figure below indicates the true, known impact effect. It is clear that the OLS estimates with omitted variables overstate the treatment effect, since there is selection into the treatment group.

```
p <- ggplot(total.res, aes(x=impact, fill=method))
p <- p + geom_histogram(position="identity", binwidth=0.01)
p + geom_vline(xintercept = 1)
```



Some of the positive bias is not totally removed. The reason for this is beyond the scope of the course, but it is clear that the bias is mitigated through conditioning on the additional cofactors.

Conditional on propensity score

Another way to view the same problem is that there is part of the treatment is explained by other, external variables. We are looking only for the causal impact of a policy, not the part that is correlated with variables

outside of the control of the policy maker. So we can separate out these two effects. We can, in effect, control for the likelihood of getting treatment in order to look *only* at the direct impact of the policy D.

We use a generalized linear model to predict the likelihood of each individual of receiving treatment. We then use that likelihood as a *new* variable p in the standard linear regression, separating the impact into p and the causal impact D. Examining the coefficient on D will give us the unbiased, causal policy impact.

```
data <- data.frame(cbind(Y, D, x1, x2, x3))

psm <- function(i) {
  idx <- sample.int(N,n)
  d <- data[idx,]
  logit <- glm(D ~ x1 + x2 + x3, data = d, family = "binomial")
  d$p <- logit$fitted.values
  ols <- lm(Y ~ D + p, data = d)
  ols$coefficients[["D"]]
}

res.psm <- data.frame(impact=sapply(1:B, psm), method=c("psm"))
summary(res.psm)
```

```
##      impact       method
##  Min.   :0.6809   psm:5000
##  1st Qu.:0.9975
##  Median :1.0744
##  Mean   :1.0767
##  3rd Qu.:1.1558
##  Max.   :1.4821
```

Matching in the wild

Simulation imparted a better intution of the mechanics of multivariate regression and conditioning on the propensity score. We can reference the output against a known data generating process – where we know the correct answer. This is invaluable for a coder, who mostly does work by trial and error.

For this section, we will use the Early Childhood Longitudinal Study, which tracks the learning outcomes for kindergarteners in 1998-1999. We will use a cleaned version of the data found in this repository (ecls.csv, 1.8MB). Our objective is to determine the causal impact of Catholic schools on a standardized math score (`c5r2mtsc_std`), relative to public schools. Within the cleaned data, we have defined a treatment variable `catholic`, which is equal to 1 if the student went to Catholic school and 0 if the student when to public school. However, the students that select into Catholic school may be systematically different from students that go to public schools. Income, race, mother's age, mother's education level, and recent residential moves may all determine the likelihood of going to Catholic school and, independently, math scores of students. A blunt and simple comparison of math scores in Catholic versus public schools will not capture the *causal* impact of Catholic school curriculums. It's not a fair comparison, without accounting for the students' backgrounds.

In the parlance of the previous section, `catholic` is `D`, `c5r2mtsc_std` is `Y`, and `race_white`, `p5hmage`, `w3income`, `p5numpla`, `w3momed_hsb` constitute `X`, where

- `race_white`: Is the student white (1) or not (0)?
- `p5hmage`: Mother's age
- `w3income`: Family income
- `p5numpla`: Number of places the student has lived for at least 4 months
- `w3momed_hsb`: Is the mother's education level high-school or below (1) or some college or more (0)?

First, the simple, dumb comparison. What is the average math scores

```
ecls.full <- read.csv("ecls.csv")

cols <- c(
    'catholic',
    'c5r2mtsc_std',
    'race_white',
    'p5hmage',
    'w3income',
    'p5numpla',
    'w3momed_hsb'
)

ecls <- na.omit(ecls.full[cols])
summary.stats <- aggregate(ecls, by=list(ecls$catholic), FUN=mean, na.rm=TRUE)
print(summary.stats[cols])
```

```
##   catholic c5r2mtsc_std race_white  p5hmage w3income p5numpla w3momed_hsb
## 1        0   0.03303204  0.5914087 37.56576 55485.02 1.129754   0.4608970
## 2        1   0.20966787  0.7411243 39.59320 82568.94 1.091716   0.2233728
```

It is clear that the standardized math scores at Catholic schools are far higher (about 0.17) than those at public schools. It is also clear, however, that the students in Catholic schools come from families that are wealthier, whiter, more educated, and more stable. We are interested in understanding the impact of Catholic schooling, *holding all else equal*. What happens if two students with the same backgrounds are sent to different schools, one Catholic and one public? What is the average effect on math scores, purely a result of the curriculum or facility?

Employ the same process as before. Predict and control for the propensity to attend Catholic school.

```
logit <- glm(
    catholic ~ race_white + p5hmage + w3income + p5numpla + w3momed_hsb,
    data=ecls,
    family="binomial"
)

ecls$p <- logit$fitted.values
summary(lm(c5r2mtsc_std ~ catholic + p, data=ecls))
```

```
##
## Call:
## lm(formula = c5r2mtsc_std ~ catholic + p, data = ecls)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4316 -0.5963  0.0494  0.6323  3.2990
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.59316    0.01863  -31.84  < 2e-16 ***
## catholic    -0.07245    0.02766   -2.62  0.00881 **
## p            4.54117    0.11257   40.34  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9161 on 9264 degrees of freedom
```

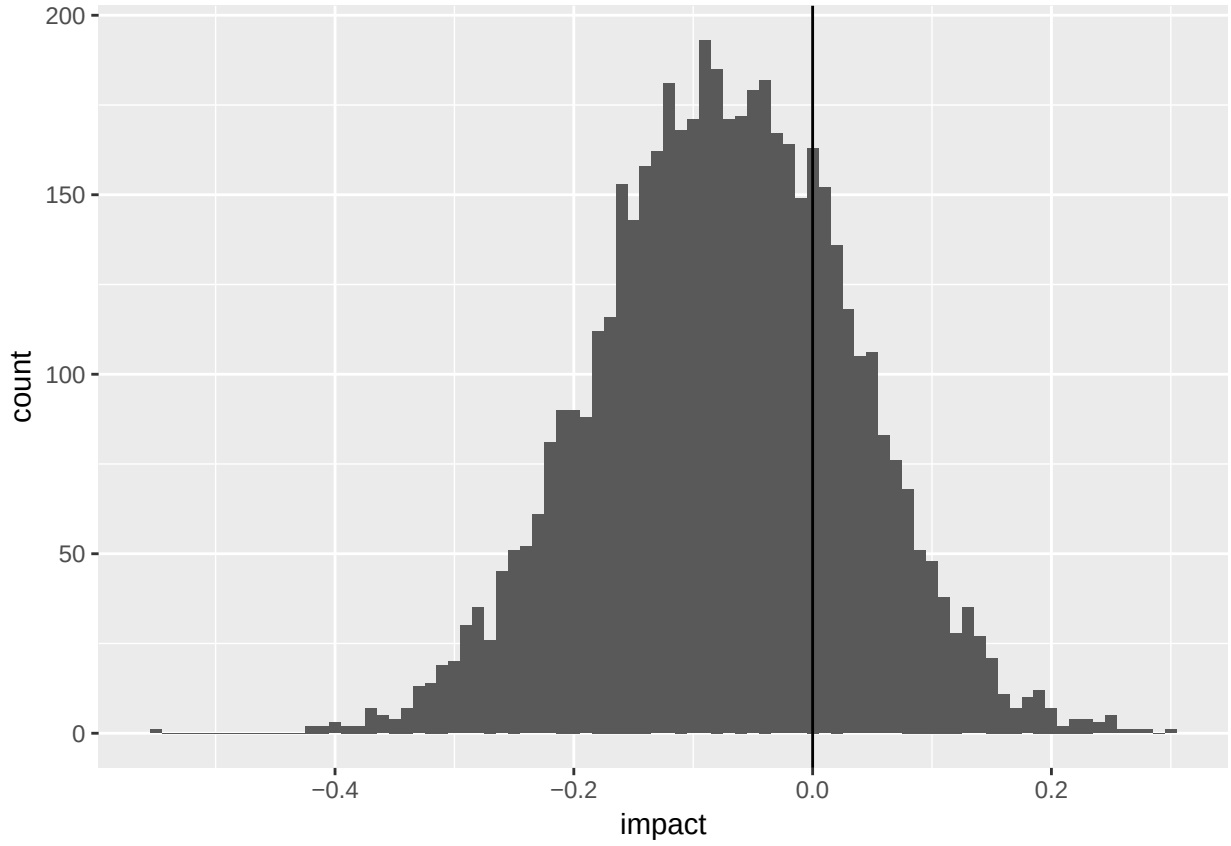
```
## Multiple R-squared:  0.1528, Adjusted R-squared:  0.1526
## F-statistic: 835.1 on 2 and 9264 DF,  p-value: < 2.2e-16
```

After controlling for the cofactors that influence the choice or ability to enter Catholic school, the estimated effect is actually *negative*. The naive results were reversed, after accounting for socioeconomics. We can explore the spread of the impact evaluation by bootstrapping the impact estimates. That is, we draw a random sample of 500 students, run the analysis, and repeat 10,000 times. The following displays the results as a histogram. It is pretty convincing that, on average, the impact of Catholic scores is less than zero (negative) – even when accounting for natural variation.

```
impact.assessment <- function(i) {
    idx <- sample.int(nrow(ecls), 500)
    d <- ecls[idx,]
    logit <- glm(
        catholic ~ race_white + p5hmage + w3income + p5numpla + w3momed_hsb,
        data=d,
        family="binomial"
    )
    d$p <- logit$fitted.values
    ols <- lm(c5r2mtsc_std ~ catholic + p, data = d)
    ols$coefficients[["catholic"]]
}

res <- data.frame(impact=sapply(1:B, impact.assessment))

p <- ggplot(res, aes(x=impact))
p <- p + geom_histogram(position="identity", binwidth=0.01)
p + geom_vline(xintercept=0) + theme(legend.position="none")
```



Is the revised estimate convincing? Probably not. The variables in our model likely don't account for all of the unobservable influences that affect both math scores and school choice. The missing values may be systematically related to math scores. The functional form is probably not linear. The error structure is not normal. There are ways to address each of these issues. The answer won't be as simple as a coefficient estimate and a p-value. However, telling stories with data and simulations can help articulate a narrative that is actionable and useful for designing policy.