Data Science for Economists

Randomized Controlled Trials and Simulations

Kyle Coombs

Table of contents

- Introduction to RCTs
- Direct Rental Assistance Example
- Power Analysis
- Simulation

Introduction to RCTs

Introduction to RCTs

- Randomized Controlled Trials (RCTs) are a gold standard in research for establishing causal relationships.
- They allow researchers to control potential outcomes by randomly assigning subjects to treatment and control groups.
- This randomization helps ensure that any differences in outcomes can be attributed to the treatment itself.

Direct Rental Assistance Example

Direct Rental Assistance Example

- On problem set, several of note DRA could help people have more control over their rent and ability to move.

- That is a great idea and better yet it is empirically testable.

- Let's say we give every low-income household in the U.S. direct rental assistance.

- We can't just give it to all of them and look at what they do:

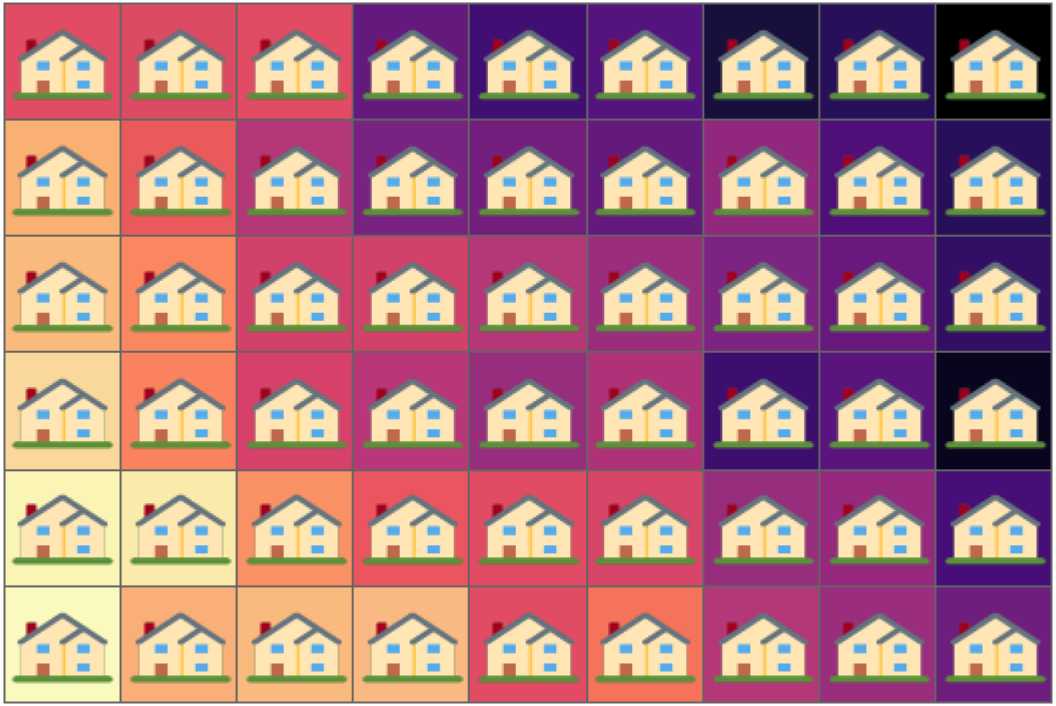
1. That's too expensive
2. We wouldn't see the counterfactual

- High income households are a bad counterfactual too

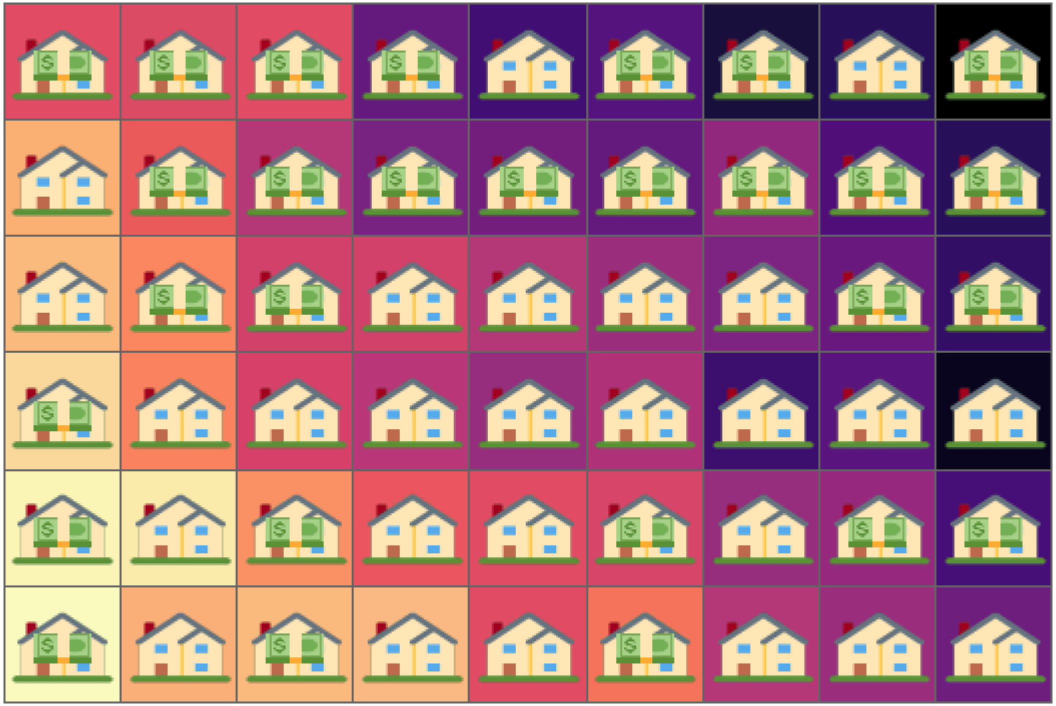
- Given we likely won't give it to everyone immediately (budget constraints):

- Why not randomize the stages?

54 households of varying starting rents (the color of the squares)



Randomly assign treatment in period 1



Give to the rest in period 2

Balance test

- After a randomization, we can check if the treated and control groups are balanced.
- We can do this by looking at the distribution of the covariates between the two groups.
- If there is no balance, we may need to re-randomize.
- Easiest way is to do a two-sample t-test

Table: Two-sample t-test balance test between treated and control households (hundreds of $)

variable	estimate	p.value	Treated	Untreated
rent	0.30	0.7367448	1.131335e+01	1.101247e+01
income	1736.65	0.6323698	5.173245e+04	4.999580e+04
age	-0.01	0.9969637	4.464286e+01	4.465385e+01
family_size	0.19	0.5572725	2.571429e+00	2.384615e+00
years_in_residence	-0.91	0.2420371	4.421429e+00	5.330769e+00
black	0.10	0.4501555	6.785714e-01	5.769231e-01

Easiest analysis two-sample t-test

- Two-sample t-test is the easiest analysis to compare means between two groups

```
ttest <- t.test(rent ~ trt, data = filter(house_did,after==1))
ttest %>%
  tidy() %>%
  mutate(estimate = -round(estimate,2),
    Difference = case_when(p.value<0.01 ~ paste0(estimate,"***"),
                           p.value<0.05 ~ paste0(estimate,"**"),
                           p.value<0.1 ~ paste0(estimate,"*"),
                           TRUE ~ as.character(estimate)),
     Treated = estimate2,
     Untreated = estimate1) %>%
  select(Difference, Untreated, Treated) %>%
  kable(caption = "Two-sample t-test of neighborhood quality between treated and control household
```

Table: Two-sample t-test of neighborhood quality between treated and control households (hundreds of $)

Difference	Untreated	Treated
-4.7*	13.01247	8.313347

Power Analysis

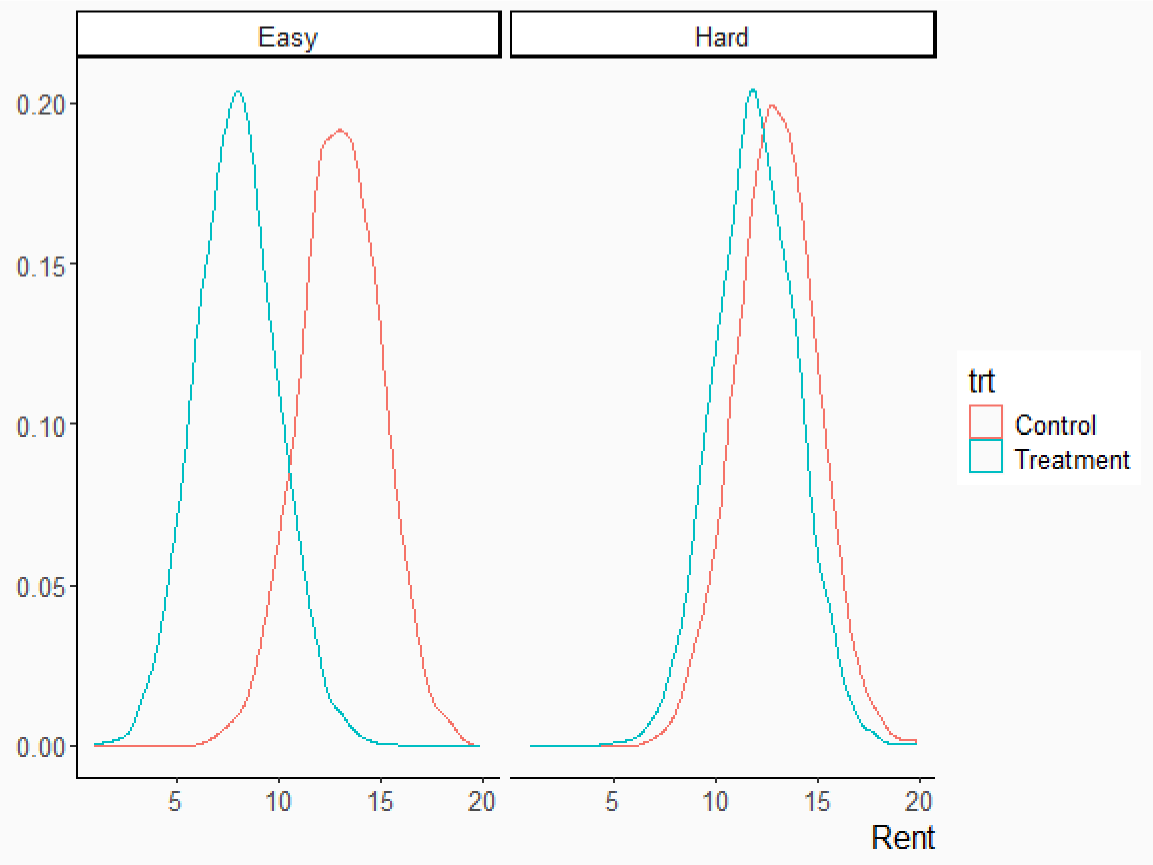
Power Analysis

- We have statistical significance, but how much can we trust it?
- How likely is it that this is a fluke and the true effect is actually zero?
    - A false positive or Type I error, let's say $\alpha = Pr(\text{reject } H_0 | H_0 \text{ is true})$
- If we had failed to reject the null hypothesis of 0, how likely is it that the true effect is actually 0?
    - A false negative or Type II error, let's say $\beta = Pr(\text{fail to reject } H_0 | H_0 \text{ is false})$
- Statistical power is defined as: $1 - \beta$
- Power analysis helps understand the probability of these errors -- given some assumptions:
- We can use power analysis to determine the sample size required to detect an effect of a given size with a certain degree of confidence.
- Alternatively, we can use power analysis to determine the power of a given sample size to detect an effect of a given size.

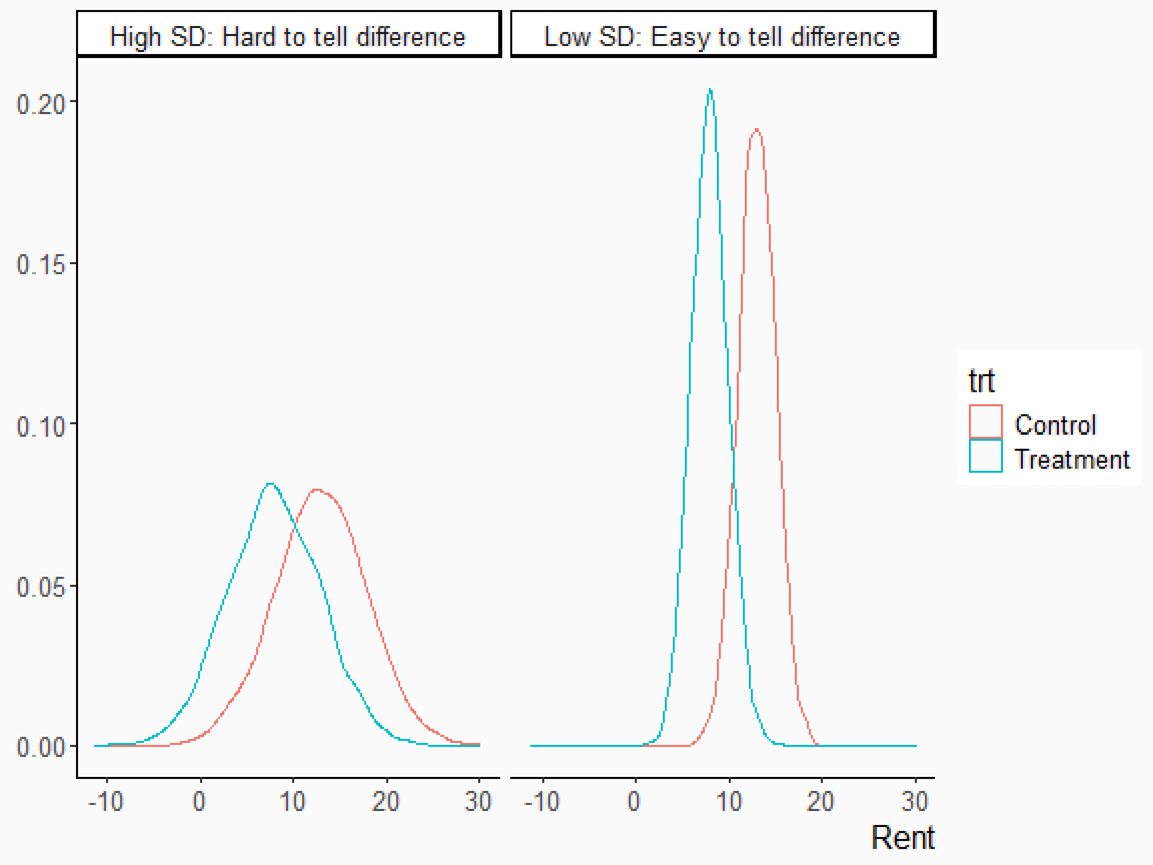
How does it work?

- Scientists determine the type I error rate, α -- that's
- But type II errors depend on a ton of factors:
  - Effect size
  - Sample size
  - Variability of the outcome
  - Type I error rate

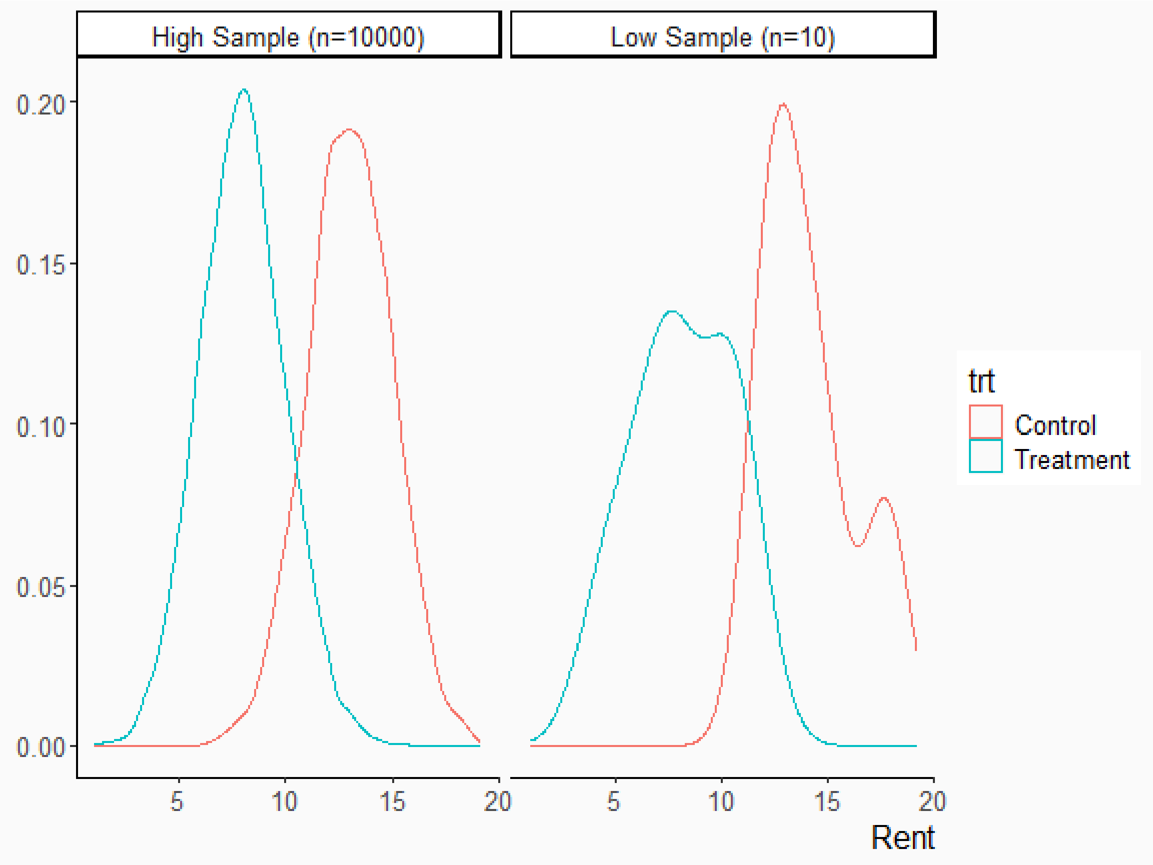
Effect: Magnitude of difference



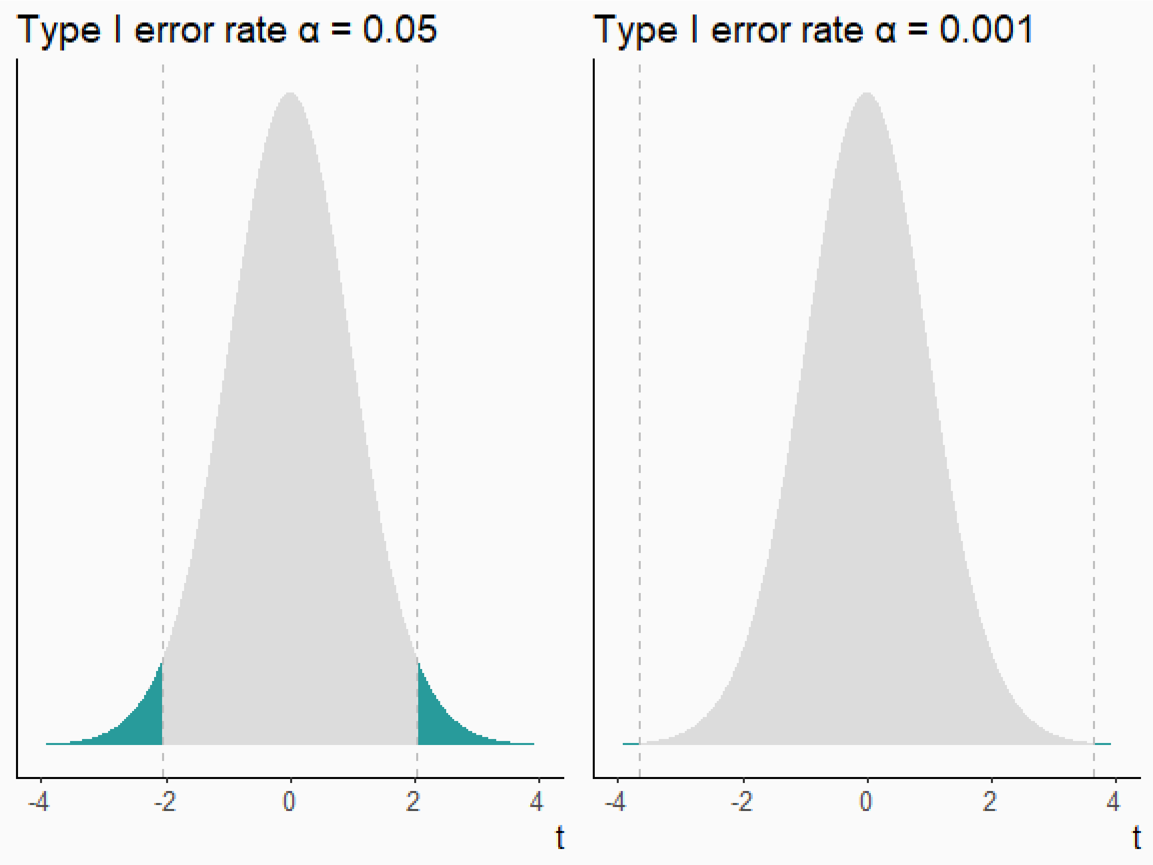
Standard deviation: Variability of the outcome



Sample Size:


High Sample (n=10000)
Low Sample (n=10)
0.20
0.15
0.10
0.05
0.00
5
10
15
20
5
10
15
20
Rent
trt
Control
Treatment

Type I error rate: $\alpha = 0.05$ vs. $\alpha = 0.001$



How do we calculate power?

The power calculations vary by test statistic, which can make them pretty tedious!

T-test:

$$1 - \beta \approx 1 - \Phi\left(1.64 - \frac{\text{Effect}}{\text{SD}/\sqrt{n}}\right)$$

where Φ is the cumulative distribution function of the standard normal distribution.

You can invert that to solve for the sample size for a given power:

$$n \approx \left(\frac{\text{Effect}}{\text{SD}}\right)^2 \left(1.64 + \Phi^{-1}(1 - \beta)\right)^2$$

Using tools like `pwr`

Thankfully, you don't have to do this by hand! `pwr` is a package that can help you calculate power for a variety of tests (but not all of them).

```
pwr::pwr.t.test(d = 0.5, power = 0.8, sig.level = 0.05, type = "two.sample", alternative = "two.si
```

```
##
##      Two-sample t test power calculation
##
##               n = 63.76561
##               d = 0.5
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

- A software called G*Power can do a lot more, but it is limited by what is pre-programmed
- But what happens when it gets more complex?

Power Analysis for more Complex Designs

- So before we did a two-way t-test and got a difference of
- But that may be insufficient if the households were not perfectly balanced too start.
- Plus, maybe household rents are correlated over time (need clustering)
- Then we'd do a diff-in-diff analysis

```
feols(rent ~ after*trt | hh + t, data = house_did,cluster=~hh) %>%
  etable() %>%
  kable(caption = "Diff-in-diff analysis of neighborhood\nquality between treated and control hous
```

Table: Diff-in-diff analysis of neighborhood quality between treated and control households (hundreds of $)

	.
Dependent Var.:	rent
after x trt	-5.0* (1.1e-15)

Power Analysis

- Uh oh, how do you do power analysis for diff-in-diff with clustering?
- The calculation is going to more complex
- What if we just simulated a bunch of fake datasets and saw how often we commit a type II error?
- We could set the standard deviation, effect size, sample size, and significance level (α)
- We could then simulate a bunch of fake datasets and see how often we reject the null hypothesis
- This is a simulation!

Simulation

Simulation Basics in R

- Simulations are a powerful tool for modeling complex systems and testing hypotheses.
- In R, you can generate random data using functions like `rnorm()`, `runif()`, and `rbinom()`.
- Simulations often involve running multiple iterations to assess variability and robustness.
- Results can be analyzed using summary statistics, visualizations, and statistical tests.
- I've literally been using them all semester

Simple Simulation Example

Here I make a dataset of 1000 observations using `tibble()` and define various parameters of the model. Let me grab the p-value from the model.

```
# Load libraries
set.seed(123)
n <- 54
effect_size <- -.5
untreated_before_mean <- 10
treated_before_mean <- 11
untreated_after_mean <- 12
treated_after_mean <- treated_before_mean+effect_size
sigma <- 5
alpha <- 0.05

# Make a tibble (or data.frame) to contain our data
tib <- tibble(
  # Create 54 units observed over 2 periods
  unit = rep(1:n, each = 2),
  time = rep(1:2, n),
  # Treatment indicator for 27 units
```

```
## P-value: 0.2243965
```

```
cat("Significance:", sig, "\n")
```

```
## Significance: Not Significant
```

Simulate it 2000 times!

R has a bunch of great ways to iterate, there's `for` loops, `lapply` and `purrr::map`. I'll cover them more latter, for now let's use a `for` loop to simulate it 2000 times![1]

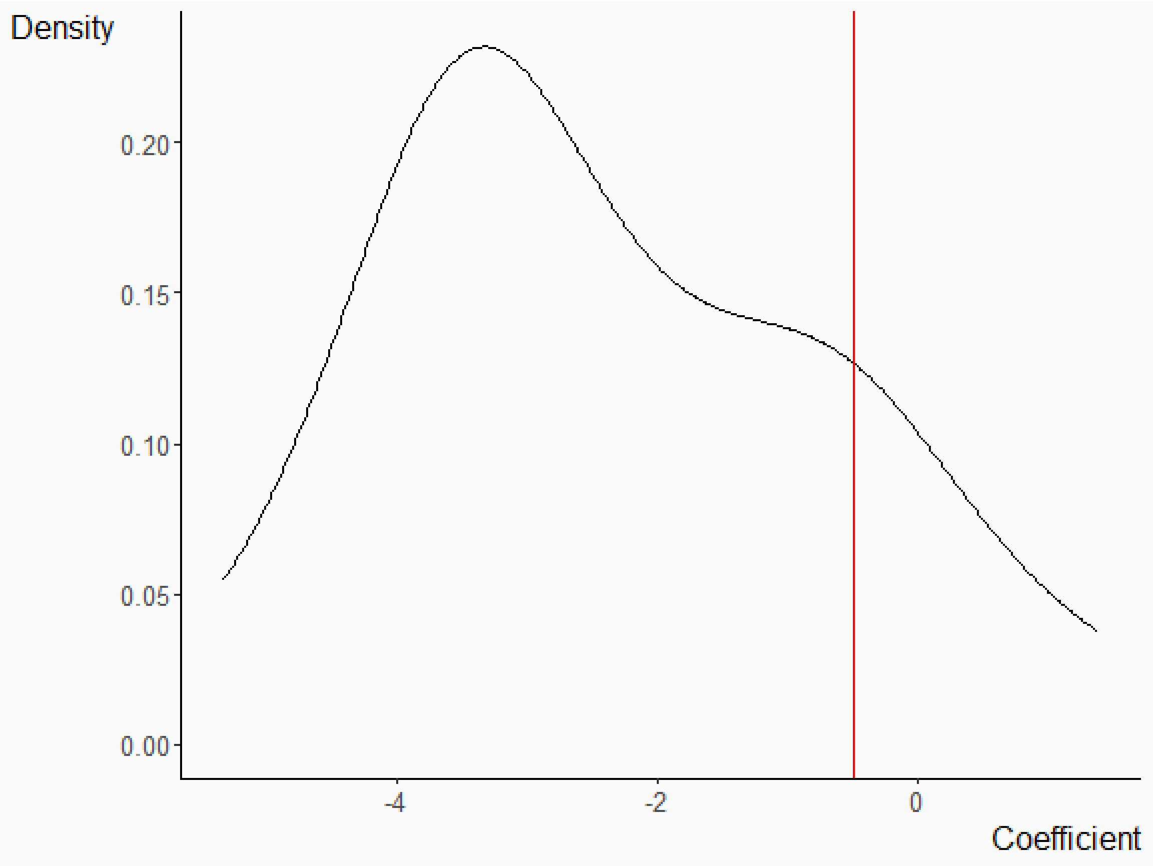
```r
# initialize a vector to store the p-values
coef_results <- c()
sig_results <- c()

for (i in 1:20) {
  tib <- tibble(
    unit = rep(1:n, each = 2),
    time = rep(1:2, n),
    treatment = rep(c(0,1), each = n/2)[unit],
    post = time == 2
  ) %>%
    mutate(
      Y = case_when(treatment==1 & post==1 ~ treated_after_mean,
                    treatment==1 & post==0 ~ treated_before_mean,
                    treatment==0 & post==1 ~ untreated_after_mean,
                    treatment==0 & post==0 ~ untreated_before_mean)+
                    rnorm(n*2, mean=0, sd=sigma)
```

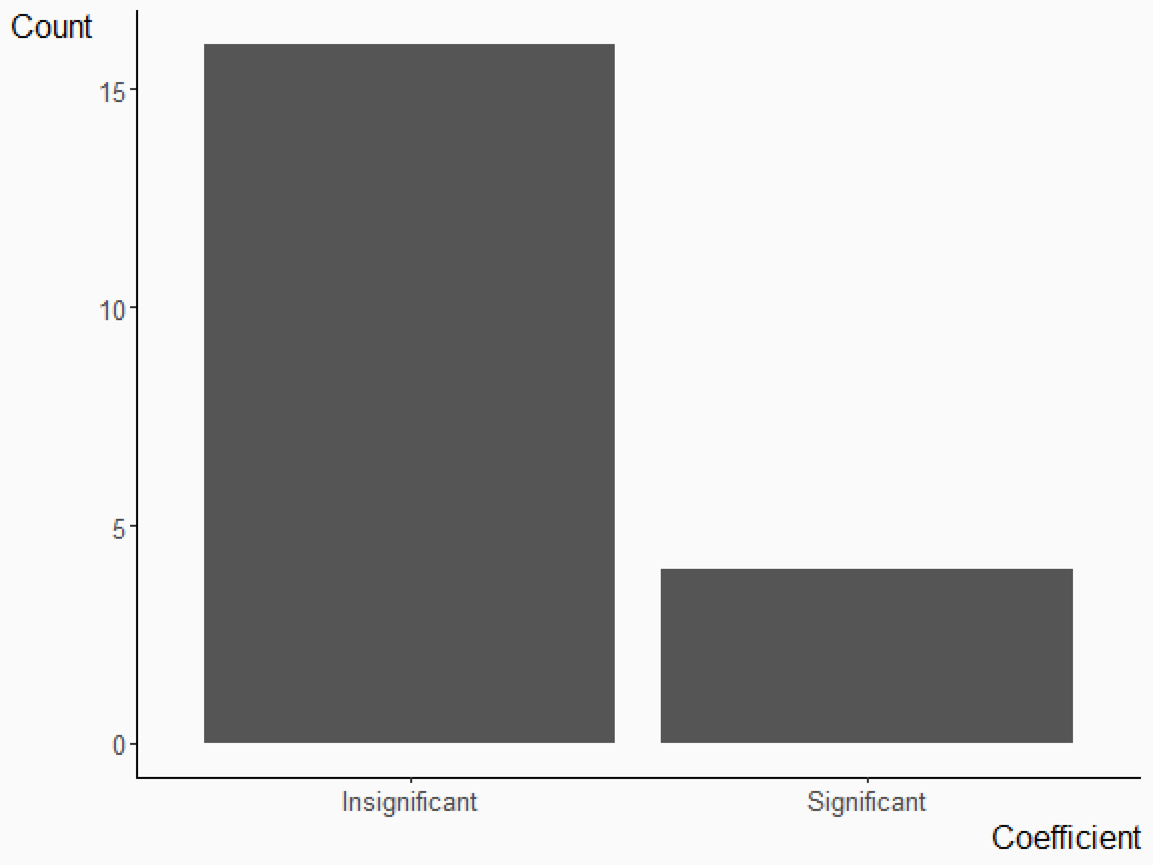
The mean of the statistically significant results is 0.2, meaning we have `mean(sig_results, na.rm=FALSE)*100` % statistical power.

[1] `for` loops are the worst way to do things in `R`, but this syntax is the most intuitive.

Let's check the distribution of coefficients



What about the share that are statistically significant?



Then fiddle with the parameters

- How does power change with sample size?
- How does power change with the effect size?
- How does power change with the significance level?
- How does power change with the standard deviation of the outcome?
- How does power change with the standard deviation of the predictor?
- What if we added square terms? Or made it an RDD design or a diff-in-diff design?
- How does power change with the number of covariates?
- How does power change with the number of clusters?

Note: Always set your seed!

- The seed is the starting point for the random number generator
- If you don't set it, you'll get a different answer every time you run the code
- This is why we set the seed to 123

```
set.seed(123)
print(rnorm(10))
```

```
##  [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774  1.71506499
##  [7]  0.46091621 -1.26506123 -0.68685285 -0.44566197
```

```
set.seed(123)
print(rnorm(10))
```

```
##  [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774  1.71506499
##  [7]  0.46091621 -1.26506123 -0.68685285 -0.44566197
```

- Got the same thing!

AI in Simulation

- AI can enhance simulation by automating the writing and execution of simulation code.
- With carefully worded instructions, AI can generate complex simulations efficiently.
- This approach can improve the robustness and reliability of research findings by exploring a wider range of scenarios.
- I might ask it:

> I want to simulate a diff-in-diff design with a treatment effect of -0.5, a sample size of 54, and a significance level of 0.05. How many times will I reject the null hypothesis if I run this simulation 2000 times?

- It could return code or a simulation function

Now go try this yourself!

- Go out and simulate the power of your favorite design!
- This is a great way to understand the power of your design!
- It's also a great way to understand the behavior of your estimator
- And it's a great way to understand the behavior of your data
- Or really to understand any tricky bit of econometrics or data analysis