

# Randomized Experiments

DPI 610

This exercise helps us understand randomized experiment basics, particularly why they allow unbiased estimates of treatment effects in the target population – even without observing both potential outcomes for any individual.

---

## Pre-Lab Exercise (Complete Before Class)

Before coming to the lab session, complete the following:

### Step 1: Load the Data

Open RStudio/Posit Cloud and load the experiment data. This dataset contains 10,000 voters where we record the potential outcomes –  $Y(1)$  and  $Y(0)$  – for each voter.

- 1 indicates they turned out to vote and;
- 0 indicates they did not.

We also record treatment status indicating *whether* or *not* each voter received a flyer with polling place information.

```
library(tidyverse)
votes <- read_csv("votes.csv")
glimpse(votes)
```

Rows: 10,000

Columns: 3

```
$ y1    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1~
$ y0    <dbl> 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1~
$ treat <dbl> 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1~
```

You'll see three columns: `y1`, `y0`, and `treat`.

- `y1` represents what would happen (voted = 1, didn't vote = 0) **if this voter received the treatment**
- `y0` represents what would happen **if this voter did not receive the treatment**
- `treat` indicates whether they actually received treatment (1) or not (0)

Note that in reality, we would never observe both potential outcomes for any single voter.

### Step 2: Reflection (2 sentences max)

In a real experiment, why can't we observe both `y1` AND `y0` for the same person?

- We can't we observe both `y1` AND `y0` for the same person because there is only one reality. Once we (don't) assign a treatment condition to an individual, we can only observe resulting outcome of that condition; the other potential outcome remains unobserved.

*Note: This impossibility—that we cannot observe both potential outcomes for the same individual simultaneously—is known as the **Fundamental Problem of Causal Inference**.*

### Step 3: Write Some Code

For how many voters in this dataset does the treatment actually make a difference (i.e., `y1` is different from `y0`)? Write 1-2 lines of code to find out.

*Hint: You can use `filter()` and `nrow()`, or `summarize()` with `sum()`.*

```
# Your code here:

votes %>%
  group_by(y1, y0) %>%
  count()
```

```
# A tibble: 4 x 3
# Groups:   y1, y0 [4]
   y1    y0     n
<dbl> <dbl> <int>
1     0     0  2087
2     0     1   895
3     1     0  4896
4     1     1  2122
```

```
votes %>%  
  filter(y1 != y0) %>%  
  nrow()
```

[1] 5791

- 5791 voters have different potential outcomes.

**ATE:** Average Treatment Effect across everyone **ATT:** Average Treatment Effect across treated group **Selection bias:**

---

## Calculating the True ATE

Now let's calculate the average treatment effect (ATE) in our sample. We can do this because we know the potential outcomes – again, in a real-world setting we would not have this information.

```
# Average treatment effect

ATE <- votes %>%
  summarize(ATE = mean(y1 - y0)) %>% # calculate ATE
  pull(ATE)
```

```
ATE
```

```
[1] 0.4001
```

---

## Worked Example: Individual Treatment Effects

Before calculating treatment effects for everyone, let's work through one example. What is the treatment effect for the first voter in our dataset?

```
# Example: What is the treatment effect for the first voter?
votes %>%
  slice(1) %>% # slice df to first observation
  mutate(ite = y1 - y0) # calculate ITE (difference b/t potential outcomes)
```

```
# A tibble: 1 x 4
   y1    y0 treat  ite
<dbl> <dbl> <dbl> <dbl>
1     1     0     1     1
```

If `ite = 1`, this voter would vote if treated but not otherwise (the treatment helps). If `ite = 0`, the treatment makes no difference for this voter. If `ite = -1`, this voter would vote if untreated but not if treated (the treatment hurts).

---

## Question 1a

Calculate an individual treatment effect for each individual. Then calculate the average of individual treatment effects. Does it match the ATE from above?

*Hint: Use `mutate()` to create a new column for the individual treatment effect, following the pattern in the worked example above.*

## Answer 1a

```
# define function to find individual treatment effect (ITE)
find_ite <- function(df) {
  df %>%
    mutate(ite = y1 - y0) # calculate ITE (difference b/t potential outcomes)
}

# count of treatment groups
find_ite(votes) %>%
  group_by(ite) %>%
  count()

# A tibble: 3 x 2
# Groups:   ite [3]
   ite     n
<dbl> <int>
1   -1   895
2    0  4209
3    1  4896

# calculate average of individual treatment effects
ITE <- votes %>%
  find_ite() %>%
  summarize(ITE = mean(ite)) %>%
  pull(ITE)

ITE

[1] 0.4001
```

- The average of individual treatment effects, 0.4001 does match the ATE calculated earlier. This makes sense looking out our treatment (-1, 0, and 1) group counts of 895, 4209, and 4896. Since an ITE of 0 indicates the treatment would have no effect on our voter, we only consider it in our sum when calculating the average ITE. Thus, the ATE is calculated as  $ITE = \frac{4896-895}{10000} = 0.4001$ .

## Question 1b

Can you calculate the average treatment effect on the treated (ATT)? ATT is the average treatment effect among those who actually receive the treatment. How does it compare to the ATE? (Hint: use the `filter()` function to restrict your data to only treated observations!)

## Answer 1b

```
ATT <- votes %>%
  filter(treat == 1) %>% # filter to treated units
  summarize(ATT = mean(y1 - y0)) %>% # calculate ATT (difference b/t potential outcomes)
  pull(ATT)

round(ATT, 4)
```

```
[1] 0.6759
```

**Interpretation:** The ATT is higher than the ATE. Why might this be? What does it suggest about who received treatment in this data?

- The ATT of 0.6759 is higher than the ATE of 0.4001. This suggests that members of the treatment group either **1)** had a higher baseline propensity to vote than member of the control group or **2)** the treatment (flyer) was more effective for them than it would've been for control members.

## Question 2: What Would an Analyst Observe?

Now let's investigate what happens when treatment is not randomly assigned. As an analyst, you can only observe outcomes for people in the condition they were actually assigned to.

Calculate the observed difference in means based upon treatment assignment: the mean of  $Y(1)$  for units where  $T = 1$  minus the mean of  $Y(0)$  for units where  $T = 0$ .

*Hint: It's easier to calculate these values separately. Create an object for each mean (e.g., `mean_y1_treated` and `mean_y0_control`), then subtract.*

## Answer 2

```
# observed difference in means, by treatment assignment
ODM <- votes %>%
  summarize(mean_y1_treated = mean(y1[treat==1]), # mean(y1) where treat==1
            mean_y0_control = mean(y0[treat==0]), # mean(y0) where treat==0,
            obs_diff_means  = mean_y1_treated - mean_y0_control) %>%
  pull(obs_diff_means)

#' using conditional indexing where the function looks at
#' only the rows where the condition is true, in this case where
#' [treat == 1] or [treat == 0]
```

**Interpretation:** Compare the observed difference in means to the true ATE you calculated earlier. Are they the same? What does this suggest about whether treatment was randomly assigned in this dataset?

- The observed difference in means is 0.2422 is lower than the true ATE of 0.4001. This suggests that treatment was not randomly assigned in this dataset, as non-random assignment can lead to biased estimates of the treatment effect.

---

## The sample function

In Question 2, we saw that non-random treatment assignment can lead to biased estimates of the treatment effect. How do we solve this problem? By assigning treatment **randomly**! The next section introduces tools to simulate randomness in R.

The `sample()` function (and `slice_sample()`, which applies to tibbles instead of just vectors) is a powerful tool in R. Let's use some examples to understand randomized experiments better. The function takes a random number of draws from a vector of things (e.g., numbers, characters, etc.). For example, suppose we wanted to randomly choose from "rock", "paper", or "scissors".

```
rps <- c("rock", "paper", "scissors")

set.seed(100) #set a seed to make the random draw reproducible

sample(rps, size = 1) # size specifies how many draws to take
```

```
[1] "paper"
```

```
sample(rps, size = 10, replace=T) # if we take more than 1
```

```
[1] "scissors" "paper"      "scissors" "rock"      "paper"      "paper"
[7] "scissors" "paper"      "paper"      "scissors"
```

```
# draw then specify whether
# you want to draw with or
# without replacement
```

## Example: Simulating Random Outcomes (Darts)

Ella has a dartboard with 10 squares on it, each worth a different point value ranging from 1 to 10. Suppose she plays a game of darts where she throws three darts, and her score is the sum of the points earned from her three throws. Let's use `sample()` to simulate 10,000 games and find her average score:

```
set.seed(1)

# Simulate 10,000 games
ella_games <- tibble(game = 1:10000) %>%
  rowwise() %>%
  mutate(score = sum(sample(1:10, 3, replace = TRUE))) %>%
  ungroup()
```



```

# Calculate average score
ella_games %>%
  summarize(average_score = mean(score))

# A tibble: 1 x 1
  average_score
      <dbl>
1          16.5

```

This simulation shows how `sample()` lets us model random processes. Each game randomly draws three dart throws, and we can calculate summary statistics across many simulations.

---

## Example: From Sampling to Experiments

Suppose we're studying a city with 100,000 people where 55% are Democrats and 45% are Republicans. We can only survey 1,000 people. Can we get an unbiased estimate of overall partisanship?

```

# Define city
city <- tibble(
  id = 1:100000,
  party = c(rep("D", 55000), rep("R", 45000))
)

set.seed(02138)

# Sample one draw of 1000 people
city_sample <- city %>%
  slice_sample(n = 1000)

# Calculate statistics
city_sample %>%
  summarize(
    mu = sum(party=="D")/n(),
    se = sqrt(mu * (1 - mu) / n()),
    ci_lower = mu - 1.96 * se,
    ci_upper = mu + 1.96 * se
  )

```

```
# A tibble: 1 x 4
      mu      se ci_lower ci_upper
<dbl> <dbl>   <dbl>   <dbl>
1 0.532 0.0158   0.501   0.563
```

---

### Key Insight: Experiments as Random Sampling

Just as we sampled 1,000 people from the city to estimate partisanship, a **randomized experiment** samples from the  $Y(1)$  column (for treated units) and the  $Y(0)$  column (for control units).

Even though we never see both  $Y(1)$  and  $Y(0)$  for any individual, randomization ensures our samples are representative of the full columns. This is why randomized experiments can provide unbiased estimates of causal effects!

Think of it this way:

- The **darts simulation** taught you how `sample()` works
  - The **city example** showed that random sampling gives unbiased estimates of population quantities
  - A **randomized experiment** combines these ideas: we randomly “sample” which people show us their  $Y(1)$  and which show us their  $Y(0)$
- 

### Question 3

Can you use the `sample()` function or `slice_sample()` function to show how running a randomized experiment leads to an unbiased estimate of the ATE in the `votes` data set?

**Hints:**

1. Create a new column with randomly assigned treatment (use `sample(0:1, n(), replace=TRUE)`)
2. Calculate mean  $Y(1)$  among those randomly assigned  $T = 1$
3. Calculate mean  $Y(0)$  among those randomly assigned  $T = 0$
4. Compare the difference to the true ATE we calculated at the beginning

### Answer 3

#Insert Answer Here

**Interpretation:** How does your estimated ATE from the randomized assignment compare to the true ATE? How does it compare to the biased estimate from Question 2?

*Your answer:*

*Optional extension: Repeat the randomization 1,000 times and plot the distribution of estimated ATEs.*