Week 2: Randomized Experiments

Tanisha Mohapatra & Fernando Sanchez Monforte

HT 2025

1 Introduction

In this week's lecture, we delved into the core concepts of causal inference, focusing particularly on the role of random assignment in experimental design. We discussed how random assignment is pivotal in addressing **The Fundamental Problem of Causal Inference**. By equitably distributing both observed and unobserved confounders across treatment and control groups, random assignment strengthens the internal validity of our causal inferences.

We also established that randomized experiments are often considered the 'gold standard' of causal research due to their robustness in establishing causal relationships. The analysis of such experiments is typically straightforward, employing tools like t-tests and linear regression for estimations. However, it's crucial to remember the potential trade-offs between internal and external validity in experimental research.

In this lab, we will bring these concepts to life. Specifically, we will:

1. Create and analyse a hypothetical randomised experiment to illustrate the efficacy of random assignment in overcoming the fundamental problem of causal inference.
2. Use real data from a field experiment by Gerber, Green, and Larimer to explore the motivations behind voting behavior, employing regression analyses to estimate Average Treatment Effects (ATE).
3. Assess the impact of different treatments and the role of extrinsic and intrinsic motivations in voter turnout.
4. Explore how to check for covariate balance in treatment and control groups and discuss the implications for selection bias and randomisation efficacy.

Before starting this seminar

1. Create a folder called `lab2`.
2. Download the data and the empty RMarkdown file (available on Canvas).
3. Save both in our `lab2` folder.
4. Open the RMarkdown file.
5. Set your working directory using the `setwd()` function or by clicking on "More". For example, this may look like this: `setwd(\"\~/Desktop/Causal Inference/2024/lab2\")`.

2 How functions function

One of the great strengths of R is the user's ability to write their own functions. Sometimes there is a small task (or series of tasks) you need done and you find yourself having to repeat it multiple times. In these types of situations it can be helpful to create your own custom function. The structure of a function is given below:

When defining the function you will want to provide the list of arguments required (inputs and/or options to modify behaviour of the function), and wrapped between curly brackets place the tasks that are being executed on/using those arguments. The argument(s) can be any type of object (like a scalar, a matrix, a dataframe, a vector, a logical, etc), and it's not necessary to define what it is in any way.

Finally, you can "return" the value of the object from the function, meaning pass the value of it into the global environment. The important idea behind functions is that objects that are created within the function are local to the environment of the function – they don't exist outside of the function.

> **Note**: You can also have a function that doesn't require any arguments, nor will it return anything.

2.1 Toss a coin

2.1.1 Write a basic function

Let's try creating a simple example function. This function will flip a coin for us and give us the result.

```r
toss_a_coin <- function(n) {
    sample(c("heads", "tails"), n, replace = T)
}
```

Now, we can use the function as we would any other function. We type out the name of the function, and inside the parentheses we provide a numeric value `n`:

```r
toss_a_coin(5)
```

```
## [1] "tails" "tails" "tails" "heads" "heads"
```

2.1.2 Replicate functions

We can also create a dataset with multiple draws. You can do this using the `replicate()` command, with two arguments: the first is the number of simulations and the second is the function being evaluated. The `replicate()` function in R is crucial for statistical simulations, like Monte Carlo methods or bootstrapping, allowing for repeated execution of a function. For instance, replicate(10, toss_a_coin(5)) performs ten sets of five coin tosses, providing a dataset to analyse patterns and probability distributions.

```
replicate(10, toss_a_coin(5))
```

```
##      [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]
## [1,] "tails" "tails" "tails" "heads" "tails" "heads" "heads" "tails" "tails"
## [2,] "heads" "tails" "tails" "heads" "heads" "heads" "tails" "heads" "heads"
## [3,] "heads" "heads" "tails" "heads" "heads" "tails" "heads" "heads" "heads"
## [4,] "heads" "heads" "heads" "tails" "tails" "tails" "tails" "heads" "tails"
## [5,] "tails" "tails" "heads" "tails" "tails" "tails" "tails" "tails" "heads"
##      [,10]
## [1,] "tails"
## [2,] "tails"
## [3,] "tails"
## [4,] "heads"
## [5,] "tails"
```

2.2 Find the mean

Let's try for another simple example. This code gives us the numbers 1 to 15.

```
numbers <- 1:15 # Save numbers 1 to 15 in a vector
numbers # Check that it worked
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

If we calculated the mean by hand, we would sum all values up and divide the sum by the number of observations.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Or in R code:

```
sum(numbers)/length(numbers)
```

```
## [1] 8
```

2.3 Write a basic function

But we might want to do this operation more than once. So, let's put this into a function:

```
find_mean <- function(vector){
  # Store the sample size
  n <- length(vector)
  # Sum up all values
  sum <- sum(vector)
  # Divide sum by n
  mean <- sum/n
  # Return the mean
  return(mean)
}
```

And let's try our `find_mean` function out:

```
find_mean(numbers)
```

```
## [1] 8
```

Pretty simple, right? In this case, we only had one line of code that was run, but in theory you could have many lines of code to get obtain the final results that you want to "return" to the user.

But isn't there a function for this already? Let's see what the inbuilt base R `mean()` function does.

```
?mean
```

It seems like the function does the exact same. So let's see if we get the same result by using a basic call like we did before.

```
mean(numbers)
```

```
## [1] 8
```

So far, so good. But what is going on under the hood?

```
getS3method("mean", "default")
```

```
## function (x, trim = 0, na.rm = FALSE, ...)
## {
##     if (!is.numeric(x) && !is.complex(x) && !is.logical(x)) {
##         warning("argument is not numeric or logical: returning NA")
##         return(NA_real_)
##     }
##     if (isTRUE(na.rm))
##         x <- x[!is.na(x)]
##     if (!is.numeric(trim) || length(trim) != 1L)
##         stop("'trim' must be numeric of length one")
##     n <- length(x)
##     if (trim > 0 && n) {
##         if (is.complex(x))
##             stop("trimmed means are not defined for complex data")
##         if (anyNA(x))
##             return(NA_real_)
##         if (trim >= 0.5)
##             return(stats::median(x, na.rm = FALSE))
##         lo <- floor(n * trim) + 1
##         hi <- n + 1 - lo
##         x <- sort.int(x, partial = unique(c(lo, hi)))[lo:hi]
##     }
##     .Internal(mean(x))
## }
## <bytecode: 0x1081fb740>
## <environment: namespace:base>
```

Much more than in our function! But now that you know how functions work, you can write your own functions for all sorts of things: automate recoding tasks, simulate data or simply play around!

3 Randomisation

In this question we will demonstrate that randomised experiments work by creating an imaginary experiment, using simulated data. We will simulate the potential outcome under control (Y_0) and the potential outcome under treatment (Y_1) for 100 units that form the sample for our experiment. Note that this is a *purely hypothetical scenario*. In reality, we never observe potential outcomes under both treatment and control for the same units: we only observe one of them (the fundamental problem of causal inference). By creating a randomised experiment with this dataset, we'll demonstrate that experiments overcome the fundamental problem.

Let's start by simulating data of our hypothetical experiment. For that, we will need to use the command `set.seed(1)` so that your results are the same as the solutions.

```r
# Set seed
set.seed(1)

# Simulate potential outcome under control (Y0) for 100 units
y0 <- round(rnorm(100, 50, 20), digits = 2)

# Set seed (again!)
set.seed(1)

# Simulate potential outcome under treatment (Y1) for 100 units
y1 <- y0 + rnorm(100, 20, 5)

# Tie everything into a dataset
a <- data.frame(cbind(y0, y1))
```

```r
# Get rid of unnecessary vectors
rm(y0, y1)
```

Task 1.1
Find the true Average Treatment Effect for all units, using Y_0 and Y_1.

```r
# YOUR ANSWER HERE
```

Task 1.2
Now, we'll randomly assign half of the units to treatment and half to control by creating a new variable indicating treatment status. We can do this using the following steps: i) First, input the command `set.seed(1)` again so that your results are the same as the solutions.

```r
# YOUR ANSWER HERE
```

ii) Assign each unit a random number from 1 to 100: create a new column in the dataset named `rand`, using the `sample()` command and a vector of the numbers 1 to 100.

```r
# YOUR ANSWER HERE
```

iii) Re-order the dataset from lowest to highest value of rand using the code `a <- a[order(a$rand),]`.

```r
# YOUR ANSWER HERE
```

iv) Create a treatment variable named `tr` that equals 1 for the first 50 units and 0 for the second 50 using the code `c(rep(1,50),rep(0,50))`.

```r
# YOUR ANSWER HERE
```

Task 1.3
Conduct a test to assess whether the treatment and control groups have the same average potential outcomes under control (Y_0). Has randomisation succeeded in creating treatment and control groups with equivalent potential outcomes under control?

```r
# YOUR ANSWER HERE
```

> Remember that randomisation isn't guaranteed to completely equalise potential outcomes in any one instance. Instead, it does so in expectation over many repeated randomisations, as we show below.

Task 1.4
Find the Average Treatment Effect from the experiment. How similar is it to the true Average Treatment Effect?

```
# YOUR ANSWER HERE
```

Task 1.5
Now, let's see how our experimental procedure performs over repeated randomisations, using a simulation:

Create a function that takes in the dataset `a`, carries out a randomised experiment, and plots the observed outcomes in treatment and control groups. You can do this by using code from task 1.2 and the plot function that we already provide below. Do we have to set a seed here?

> **Code Hint**: Remember that a function in R has the following format: `function.name <- function(input) { function commands }`.

```
# YOUR ANSWER HERE
```

Second, show the results of 5 randomised experiments by running the function 5 times.

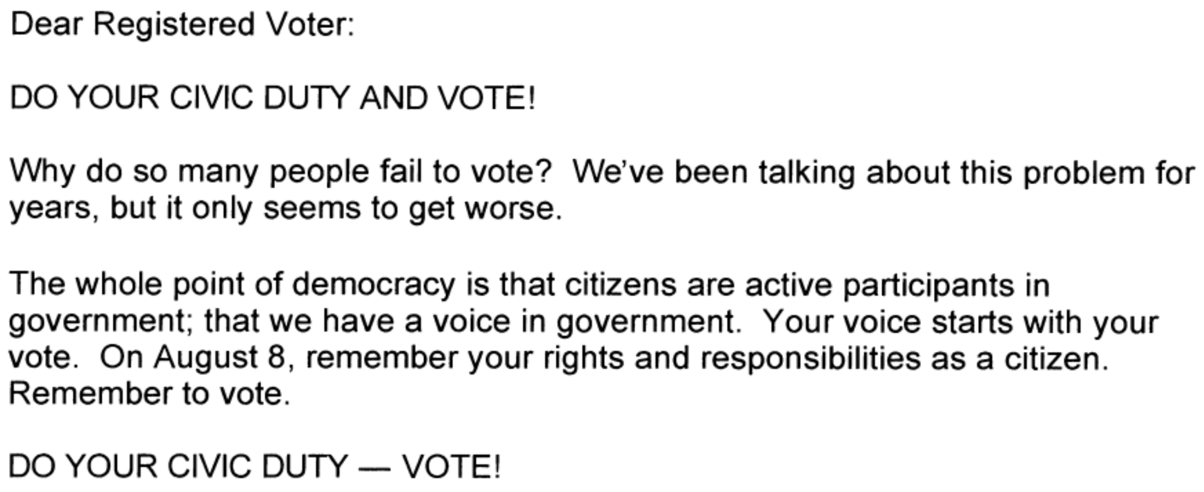
```
# YOUR ANSWER HERE
```

4 Social norms and turnout: A field experiment

4.1 Introduction

Why do people bother to vote? One hypothesis is adherence to social norms. Voting is widely regarded as a civic duty and people worry that others will think badly of them if they fail to participate. According to this theory, voters may receive two different types of utility from voting; (a) the intrinsic rewards from performing this duty and (b) the extrinsic rewards received when others observe them doing so. To gauge the effects of priming intrinsic motives and applying varying degrees of extrinsic pressure on voting behaviour, Gerber, Green, and Larimer conducted a famous field experiment in Michigan prior to the August 2006 primary election. The sample for the experiment was 344,084 voters. They were randomly assigned to either the control group or one of four treatment groups.
Gerber, A., Green, D., & Larimer, C. (2008). "Social Pressure and Voter Turnout: Evidence from a Large-Scale Field Experiment." *American Political Science Review*, 102 (1): 33-48.

Treatment: Civic duty
We'll practice analysing experiments by focusing on two of their treatments. The first treatment, "civic duty", involved sending a letter to the voter carrying the message "DO YOUR CIVIC DUTY - VOTE!".


Dear Registered Voter:

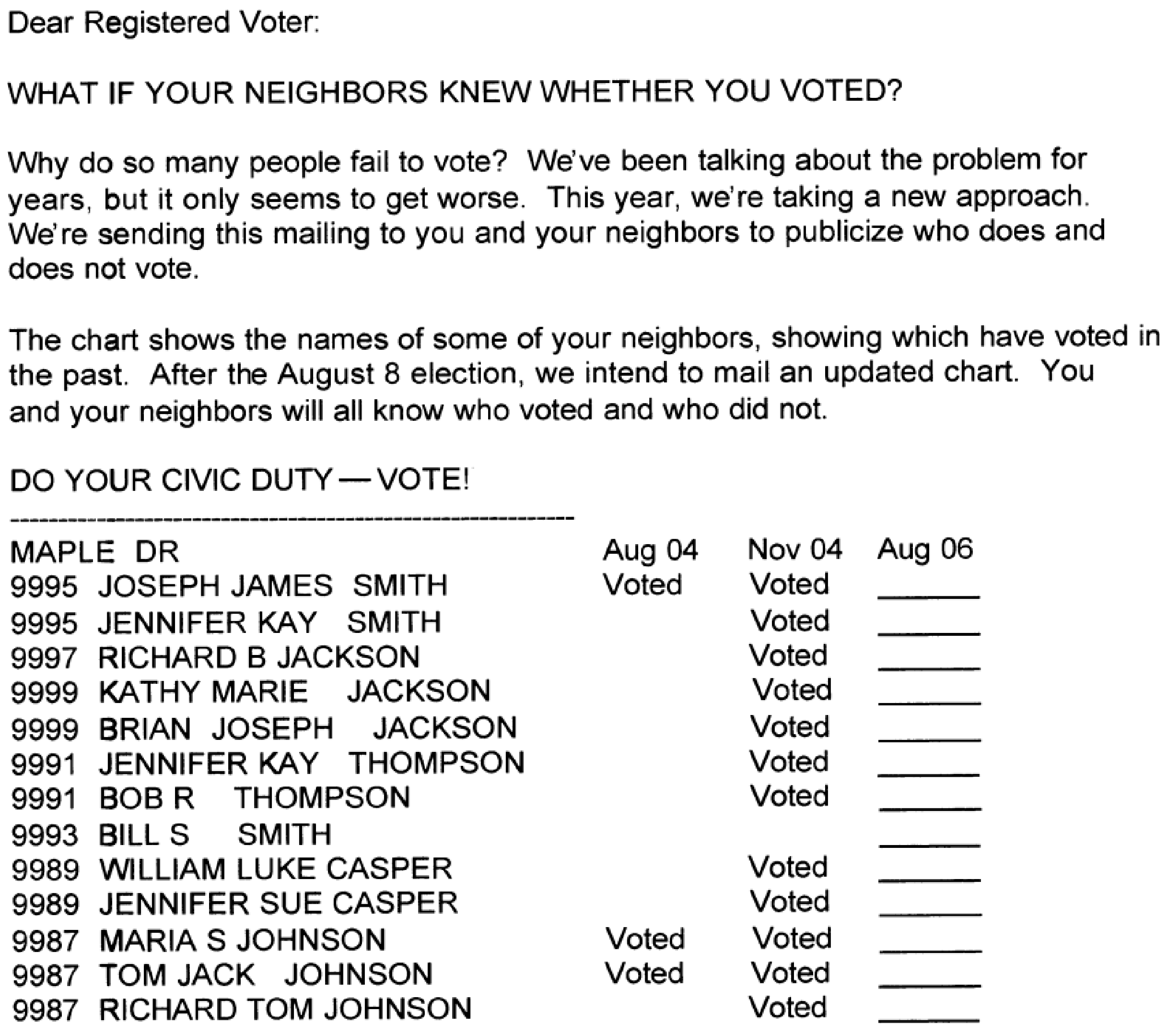
DO YOUR CIVIC DUTY AND VOTE!

Why do so many people fail to vote? We've been talking about this problem for years, but it only seems to get worse.

The whole point of democracy is that citizens are active participants in government; that we have a voice in government. Your voice starts with your vote. On August 8, remember your rights and responsibilities as a citizen. Remember to vote.

DO YOUR CIVIC DUTY — VOTE!

Treatment: Neighbours
The second treatment, “Neighbors” sent the same letter, but also informed the voter that who votes is public information (which is the case by law in the USA). It listed the recent voting record of each registered voter in the household and the voting records of those living nearby, and stated that a follow-up letter after the election would report back to the household and to their neighbours on who had voted and who had not.The idea was to see whether priming extrinsic motivations would encourage this treatment group to turn out more than the control group, who received no letter.



Dear Registered Voter:

WHAT IF YOUR NEIGHBORS KNEW WHETHER YOU VOTED?

Why do so many people fail to vote? We've been talking about the problem for years, but it only seems to get worse. This year, we're taking a new approach. We're sending this mailing to you and your neighbors to publicize who does and does not vote.

The chart shows the names of some of your neighbors, showing which have voted in the past. After the August 8 election, we intend to mail an updated chart. You and your neighbors will all know who voted and who did not.

DO YOUR CIVIC DUTY — VOTE!

MAPLE DR	Aug 04	Nov 04	Aug 06
9995 JOSEPH JAMES SMITH	Voted	Voted	______
9995 JENNIFER KAY SMITH		Voted	______
9997 RICHARD B JACKSON		Voted	______
9999 KATHY MARIE JACKSON		Voted	______
9999 BRIAN JOSEPH JACKSON		Voted	______
9991 JENNIFER KAY THOMPSON		Voted	______
9991 BOB R THOMPSON		Voted	______
9993 BILL S SMITH			______
9989 WILLIAM LUKE CASPER		Voted	______
9989 JENNIFER SUE CASPER		Voted	______
9987 MARIA S JOHNSON	Voted	Voted	______
9987 TOM JACK JOHNSON	Voted	Voted	______
9987 RICHARD TOM JOHNSON		Voted	______

Figure 1: *Letter sent to people in the neighbour group.*

For this question we'll use the original data of Gerber et al, contained in the file `gerber.Rda`.

Below is a list of the variable definitions:

- `sex` - gender (1 if female, 0 if male)
- `yob` - year of birth
- `p2004` - 1 if Respondent voted in the 2004 Primary Election, 0 otherwise
- `voting` - 1 if Respondent voted in the 2006 Primary Election, 0 otherwise (the outcome variable)
- `control` - 1 if Respondent is assigned to the control group, 0 otherwise
- `civicduty` - 1 if Respondent is assigned to the "Civic Duty" group, 0 otherwise
- `neighbors` - 1 if Respondent is assigned to the "Neighbors" group, 0 otherwise

Let's first load the `gerber.Rda` data into our environment!

```
# YOUR ANSWER HERE
```

Task 2.1
For both treatments, calculate the average treatment effect and test whether it is statistically significant. Interpret the results, giving a precise explanation of the magnitude of the treatment effects. What do they suggest about the motivations that people have for voting?

```
# YOUR ANSWER HERE
```

Task 2.2
For both treatment groups, compare the mean values of `yob`, `sex` and `p2004` to the control group. Do the results suggest that randomisation was successful? Is selection bias likely to be a problem in this experiment?

```
# YOUR ANSWER HERE
```

Task 2.3
Calculate the ATE for the the neighbors treatment using:

a) A regression containing only neighbors.
   Note that you will need to subset your data appropriately in order to obtain the correct control group.

```
# YOUR ANSWER HERE
```

b) A regression containing neighbors and the three background characteristics.
   Are there any big differences in the estimated ATE between the two specifications? Or between these two estimates and the result from task 2.1? Why or why not?

```
# YOUR ANSWER HERE
```