Section: Module 2

Permutation Test

Kentaro Nakamura

GOV 2002

September 19th, 2025

Logistics

- Section Location: CGIS South S001 Kin-Chung Lam Room
- If you have any question,
  - visit office hours
  - ask question on **Ed**
- General advice:
  - Pay attention to the **details** of the slides
  - Work on review questions every week
  - Work on problem set (even if you do not submit)
  - Read Imbens and Rubin (2015, Chapter 5 for permutation test)
- Midterm is typically harder than problem set, so be prepared!

Problem Set 2: Structure

- Question 2: Permutation Test
  - 2(a): Wilcoxon Rank Sum Test
  - 2(b): Permutation test with pre-treatment covariates
  - 2(c): Test Inversion and Hodges-Lehmann estimate
  - 2(d): Test about Maximal Effect
- Question 3: Conditional Independence Test

Permutation Test: Review

- Finite population framework
  - Only source of randomness is **treatment assignment**
  - We fix the potential outcome $\mathcal{O}_n = \{Y_i(1), Y_i(0)\}_{i=1}^n$
- Assume **Unconfoundedness**
  - $\{Y_i(1), Y_i(0)\} \perp\!\!\!\perp T_i$
  - Guaranteed by design (RCT) $\rightarrow$ we can permute the treatment
- Overview of Hypothesis Testing Procedure
  - Assume the null hypothesis
  - Permute treatment assignment under the null and calculate the test statistic
  - If we see that the observed test statistic is significantly different from the ones calculated under the null, then it indicates the null hypothesis is wrong

Permutation Test: Example

ID	T_i	Y_i	$Y_i(1)$	$Y_i(0)$
1	1	3	?	?
2	1	5	?	?
3	0	2	?	?
4	0	1	?	?

- Calculate the observed test statistic

$$
\begin{aligned}
S &= \sum_{i=1}^{4} T_i R_i(Y_1, Y_2, Y_3, Y_4) \\
&= 3 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 0 \\
&= 7
\end{aligned}
$$

Permutation Test: Example

- By using $Y_i = T_i Y_i(1) + (1 - T_i) Y_i(0)$ (consistency)

ID	T_i	Y_i	$Y_i(1)$	$Y_i(0)$
1	1	3	3	?
2	1	5	5	?
3	0	2	?	2
4	0	1	?	1

Permutation Test: Example

- Now, let's assume sharp null $H_0 : Y_i(1) = Y_i(0)$
- Under the sharp null, we can fill the missing values.

ID	T_i	Y_i	$Y_i(1)$	$Y_i(0)$
1	1	3	3	3
2	1	5	5	5
3	0	2	2	2
4	0	1	1	1

Permutation Test: Example

- Under the sharp null, let's permute treatment assignment
- There are six combinations of treatment assignment
  - (1,1,0,0)
  - (1,0,1,0)
  - (1,0,0,1)
  - (0,1,1,0)
  - (0,1,0,1)
  - (0,0,1,1)

Permutation Test: Example

ID	T_i	Y_i	$Y_i(1)$	$Y_i(0)$	$R_i(\mathbf{Y}(1))$	$R_i(\mathbf{Y}(0))$
1	1	3	3	3	3	3
2	1	5	5	5	4	4
3	0	2	2	2	2	2
4	0	1	1	1	1	1

- Compute the test statistic under the permuted assignment
  - (1,1,0,0) $\rightarrow 3+4=7$
  - (1,0,1,0) $\rightarrow 3+2=5$
  - (1,0,0,1) $\rightarrow 3+1=4$
  - (0,1,1,0) $\rightarrow 4+2=6$
  - (0,1,0,1) $\rightarrow 4+1=5$
  - (0,0,1,1) $\rightarrow 2+1=3$

Permutation Test: Coding Exercise

- Let's try writing the code of Wilcoxon's rank sum test together!
- *Procedure*
  - STEP 1: Write a function to calculate test statistic (hint: use `rank` function)
  - STEP 2: Calculate test statistic with observed data
  - STEP 3: Permute the treatment and calculate test statistic under the permuted treatment assignment (use `sample` function)
  - STEP 4: Calculate p-value
- Note: `wilcox.test()` gives you **Mann-Whitney U test statistic**

$$U = \sum_{i=1}^{n} T_i R_i(Y_1, \cdots, Y_n) - \underbrace{\frac{n_1(n_1+1)}{2}}_{\text{Mean}}$$

so without second term, the function gives the different value of test statistic

Permutation Test: Example Code

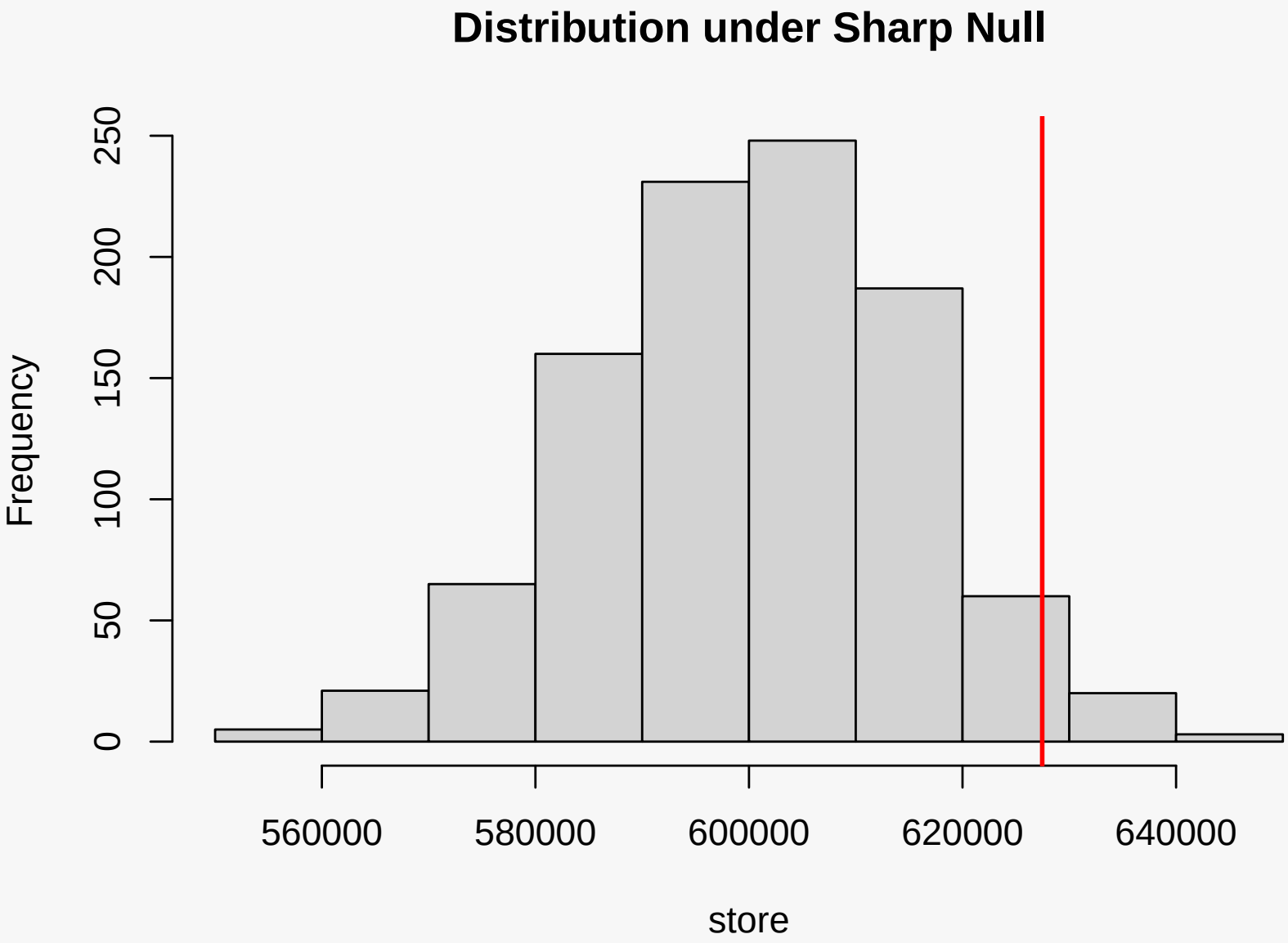
```r
# function to calculate test statistics
calc_test_statistics <- function(treat, outcome) {
  n1 <- sum(treat)
  W  <- sum(treat * rank(as.numeric(outcome)))
  U  <- W - n1*(n1+1)/2
  return(U)
}
obs <- calc_test_statistics(treat = treat, outcome = Y)
```

Permutation Test: Example Code

```r
nsim <- 1000 # nsim: Number of Monte Carlo Approximation
store <- rep(NA, nsim) #storage for each result

## Monte Carlo Approximation
for (b in 1:nsim){
  # permute treatment
  treat_sim <- sample(treat)
  # calculate test statistic
  store[b] <- calc_test_statistics(treat = treat_sim, outcome =  Y)
}
```

Permutation Test: Reference Distribution



Caution: with and without replacement (added)

- You need to use **without replacement** (i.e., `replace = FALSE`)
  - With replacement: After you draw an item, you put it back into the population before drawing again.
  - Without replacement: After you draw an item, you do not put it back.
- This is because you need to fix the number of treated observation
  - If you put it back, then it is not guaranteed that the sum of treated people stays the same.
- Let's check how the code works.
  - With replacement, you might get more or less treated people.

```
# with replacement
set.seed(123)
sample(c(1,1,1,0,0,0), replace = TRUE) # 4 observations are treated, not 3!
```

```
## [1] 1 0 1 1 1 0
```

More general sharp null (1)

- Consider more general sharp null $H_0 : Y_i(1) = Y_i(0) + \tau_0$
  - Let's say, $\tau_0 = 5$
- Let's think about the same toy example

ID	T_i	Y_i	$Y_i(1)$	$Y_i(0)$
1	1	3	?	?
2	1	5	?	?
3	0	2	?	?
4	0	1	?	?

More general sharp null (1)

ID	T_i	Y_i	$Y_i(1)$	$Y_i(0)$
1	1	3	?	?
2	1	5	?	?
3	0	2	?	?
4	0	1	?	?

- In observed data, treatment effect adjusted Wilcoxon's rank sum test statistic is given by

$$\begin{aligned} S &= \sum_{i=1}^{4} T_i R_i(Y_1 - T_1\tau_0, Y_2 - T_2\tau_0, Y_3 - T_3\tau_0, Y_4 - T_4\tau_0) \\ &= 1 \times 1 + 2 \times 1 + 4 \times 0 + 3 \times 0 \\ &= 3 \end{aligned}$$

More general sharp null (2)

- Let's fill the table. With consistency, we can impute $Y_i(T_i)$

ID	T_i	Y_i	$Y_i(1)$	$Y_i(0)$
1	1	3	3	?
2	1	5	5	?
3	0	2	?	2
4	0	1	?	1

More general sharp null (3)

- Then, let's use sharp null $H_0 : Y_i(1) = Y_i(0) + \tau_0$ with $\tau_0 = 5$

ID	T_i	Y_i	$Y_i(1)$	$Y_i(0)$
1	1	3	3	-2
2	1	5	5	0
3	0	2	7	2
4	0	1	6	1

More general sharp null (4)

- Under the sharp null, let's permute treatment assignment and calculate the test statistic
- Notice that, under the sharp null, the test statistic is

$$S = \sum_{i=1}^{4} T_i R_i(Y_1 - T_1\tau_0, Y_2 - T_2\tau_0, Y_3 - T_3\tau_0, Y_4 - T_4\tau_0)$$
$$= \sum_{i=1}^{4} T_i \underbrace{R_i(Y_1(0), Y_2(0), Y_3(0), Y_4(0))}_{\text{Does not depend on } T_i}$$

- Since we fill $Y_i(0)$ under the sharp null, we just need to permute the treatment assignment T_i

Test Inversion: Procedure

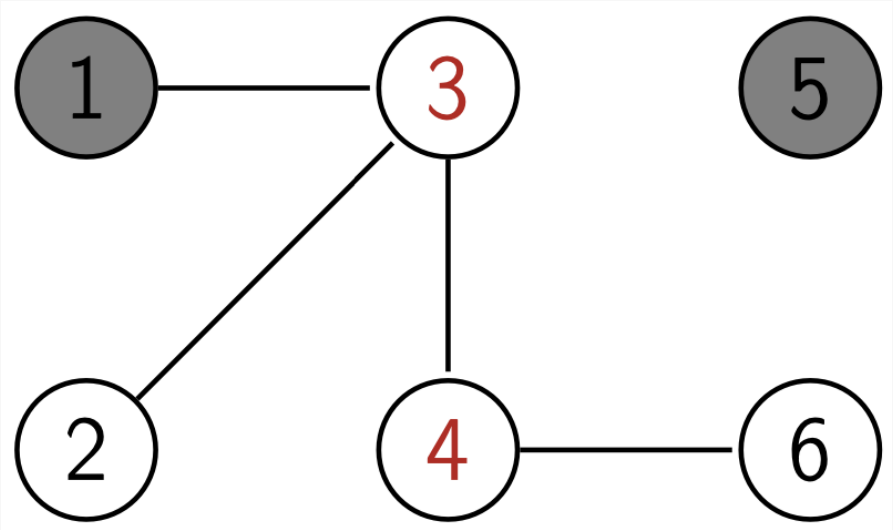
- **Procedure**
  - STEP 1: Define the range of treatment effect values
  - STEP 2: For each value in the range, calculate the test statistic and compute the p-value
  - STEP 3: Collect the range of p-values that is larger than α
- **Note**: Since the reference distribution does not depend on τ, you can simplify the procedure above using `wilcox.test()` on the treatment effect adjusted outcome

Beyond Rank Sum Test

- Procedure of permutation test does not depend on specific test statistic
  - STEP 1: Calculate test statistic with observed data
  - STEP 2: Permute the treatment assignment and calculate test statistic with permuted treatment assignment under the null
  - STEP 3: Compare the distribution of test statistic in STEP 2 (reference distribution) with the test statistic based on the observed data
- In Q2(b) and Q3, you might want to use other test statistic
- Read Chapter 5 of Imbens and Rubin (2015) for different test statistics

Recap: Conditional Randomization Test

- Let's understand the example from the class
  - Gray units: treated / White units: untreated
  - Focal units: unit 3 and 4



- Question: Are units 3 and 4 affected by the treatment of their friends?

Recap: Testing Spillover Effects

- Null Hypothesis: $Y_3, Y_4 \perp\!\!\!\perp T_1, T_2, T_5, T_6 \mid T_3, T_4$
  - Under the null, given the fixed value of T_3 and T_4, the permutation of T_1, T_2, T_5, T_6 should not affect Y_3, Y_4
- Notice the difference / connection from permutation test
  - Permutation Test: Assume sharp null. Randomization is guaranteed by design
  - CRT: Randomization is justifiable under the null
- **Procedure**
  - STEP 1: Calculate test statistic $\mathrm{Corr}(Y_{\text{focal}}, \bar{T}_{\text{friend}})$ on the observed data
  - STEP 2: Permute the treatment assignment T_1, T_2, T_5, T_6 given the original value of T_3, T_4
  - STEP 3: For each permutation, calculate the test statistic and create the reference distribution
  - STEP 4: Compare the observed test statistic with the reference distribution

Question 3: General Tips

- Understand the setup
- Consider the test statistic
- After finding test statistic, start writing the code!
    - STEP 1: Write the function to calculate the observed test statistic
    - STEP 2: Write the function to permute the treatment (but be aware how to permute!)
    - STEP 3: Calculate the reference distribution and compare it with the observed data test statistic