Randomization Inference

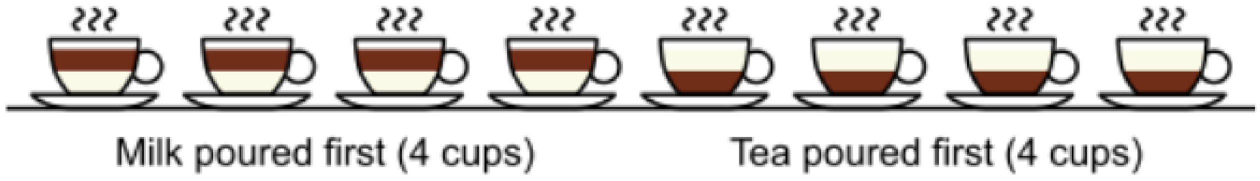
Annie Chen
1/22/2020

Problem Set 1

- Common issues
- OHs: Tuesdays 2:45pm - 3:45pm if you have a specific question about your assignment

Randomization Inference

> " A lady declares that by tasting a cup of tea made with milk she can discriminate whether the milk or the tea infusion was first added to the cup. We will consider the problem of designing an experiment by means of which this assertion can be tested. […] [It] consists in mixing eight cups of tea, four in one way and four in the other, and presenting them to the subject for judgment in a random order. The subject has been told in advance of that the test will consist, namely, that she will be asked to taste eight cups, that these shall be four of each kind […]. " — Fisher, 1935.



Hypothesis Testing in Randomization Inference

- Another way of assessing the uncertainty around the $\widehat{ATE}$ of our sample?
- Define the sharp null hypothesis as:

$$H_0 : Y_i(1) - Y_i(0) = 0$$

  for all units i.
- Note that this is "stronger" than:

$$\mathbb{E}[Y_i(1) - Y_i(0)] = 0$$

- What's the difference? Why is the first a stronger statement than the second?

Hypothesis testing in Randomization Inference

- Under the null ($H_0 : Y_i(1) = Y_i(0)$), we can construct an *exact* [1] sampling distribution for the sample ATE.
- How? Assuming $Y_i(1) = Y_i(0)$ means that we observe *both* potential outcomes for unit i! This allows us to simulate all possible randomizations with the full set of potential outcomes.
- Then, we ask: "If this null were true, how likely am I to get the estimate that I actually obtained?"

```
browseURL("https://www.jwilber.me/permutationtest")
```

1. *Exact if all possible random assignments are simulated. The number of permutations can blow up quickly, in which case, it is not practical to simulate them all. See additional notes.*

A simple example with some code

Suppose we have a vector of observed outcomes from an experiment with 9 observations where 5 the units were treated.

```r
# Treatment assignment vector
d <- c(0,1,1,1,0,1,0,0,1)

# Observed outcomes
Y <- c(2,1,3,4,2,3,0,4,6)

# compute the sample ATE
obs.sate <- mean(Y[which(d==1)]) - mean(Y[which(d==0)])
obs.sate
```

```
## [1] 1.4
```

- Exercise: generate all possible random assignments (permutations) for the 9 units, and choose only the arrangements where 5 units are treated.

Create matrix of all possible treatment assignments for all 9 units.

```
# number of units
n <- 9

# units can either be given treatment or not
D <- c(0,1)

# permute treatment assignments
all_permute_treatment <- expand.grid(rep(list(D), n))

head(all_permute_treatment)
```

```
##   Var1 Var2 Var3 Var4 Var5 Var6 Var7 Var8 Var9
## 1    0    0    0    0    0    0    0    0    0
## 2    1    0    0    0    0    0    0    0    0
## 3    0    1    0    0    0    0    0    0    0
## 4    1    1    0    0    0    0    0    0    0
## 5    0    0    1    0    0    0    0    0    0
## 6    1    0    1    0    0    0    0    0    0
```

```r
# selecting only the rows where the number of treated is equal to 5
all_permute_treatment <- all_permute_treatment[rowSums(all_permute_treatment) == 5, ]

head(all_permute_treatment, n = 10)
```

```
##    Var1 Var2 Var3 Var4 Var5 Var6 Var7 Var8 Var9
## 32    1    1    1    1    1    0    0    0    0
## 48    1    1    1    1    0    1    0    0    0
## 56    1    1    1    0    1    1    0    0    0
## 60    1    1    0    1    1    1    0    0    0
## 62    1    0    1    1    1    1    0    0    0
## 63    0    1    1    1    1    1    0    0    0
## 80    1    1    1    1    0    0    1    0    0
## 88    1    1    1    0    1    0    1    0    0
## 92    1    1    0    1    1    0    1    0    0
## 94    1    0    1    1    1    0    1    0    0
```

```
...
```

Number of Combinations

- For N observations, where n_t denotes the number of observations in the treated group, n_c is number of observations in the control group, and $n_t + n_c = N$, the permutation of assignments is:

$$\binom{N}{n_t} = \frac{N!}{n_t!(N - n_t)!} = \frac{N!}{n_t!n_c!}$$

Number of Combinations

- Our example: N = 9, treated = 5, control = 4

```
factorial(9)/(factorial(5)*factorial(4))
```

```
## [1] 126
```

```
# same as nrow(all_permute_treatment)
```

- What if N = 50? In the case where the number of possible randomizations is too large and we cannot obtain an exact distribution, we can rely on Monte Carlo approximation.

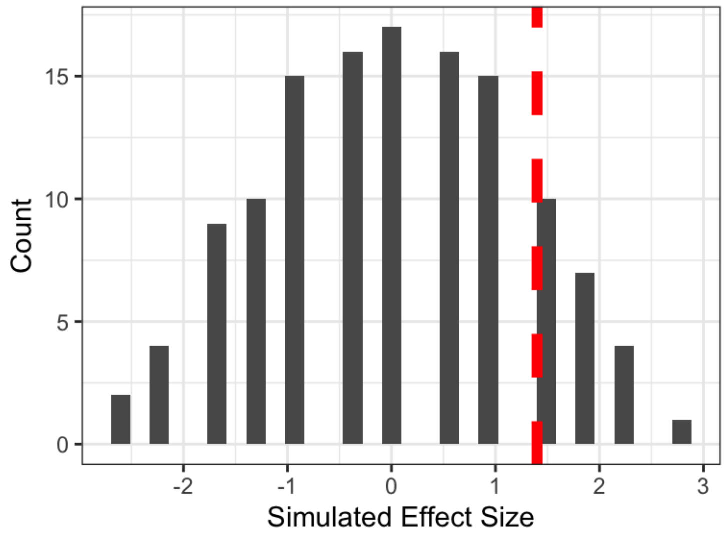
- Now, we want to simulate a distribution of a test statistic under the sharp null.
- Exercise: For each row, calculate the difference in means and plot the estimates.

- For each permutation, estimate the ATE.[2]

```r
permute.sate <- c(length(all_permute_treatment))

for(i in 1:nrow(all_permute_treatment)){
  D_star <- unlist(all_permute_treatment[i,])
  permute.sate[i] <- mean(Y[which(D_star==1)]) - mean(Y[which(D_star==0)])
}
```

2. We can always replace this with another quantity of interest (i.e. rank sum statistic). This is part of the appeal of RI!

```
# quantile(permute.sate, c(0.025, 0.975))


# this is our sampling distribution!
library(ggplot2)
qplot(permute.sate) +
  labs(x = "Simulated Effect Size", y = "Count") +
  geom_vline(xintercept = obs.sate, lty =2, col = "red", lwd = 2) +
  theme_bw()
```



- Let's count how many of the simulated values are as large (in magnitude) as the observed $\widehat{ATE}$.

```
# This is a two sided test
t_star <- length(permute.sate[which(abs(permute.sate) >= abs(obs.sate))])
```

- Then, divide by the number of combinations to get the [???] . What did we just compute? [3]

```
t_star/nrow(all_permute_treatment)
```

```
## [1] 0.2936508
```

- How do we interpret this value?
- How does this differ from a t-test?

3. It's just a counting problem:

$$p = \frac{1}{K}\sum_{k=1}^{K} \mathcal{I}(|\hat{T}_k| \geq |T|)$$

where K is the total number of combinations, and T is the test statistic. $\mathcal{I}(\cdot)$ is the indicator function.

Summary of RI Procedure

1. Define the sharp null and choose a test statistic.
2. Calculate observed test statistic.
3. Permute vectors of different possible randomization assignments.
4. Compute the test statistic for these simulations.
5. Find the p-value.

All this can be done easily with the `ri2` package!

- Key functions: `declare_ra()` and `conduct_ri()`
- Can also specify different designs (clustered, blocking).

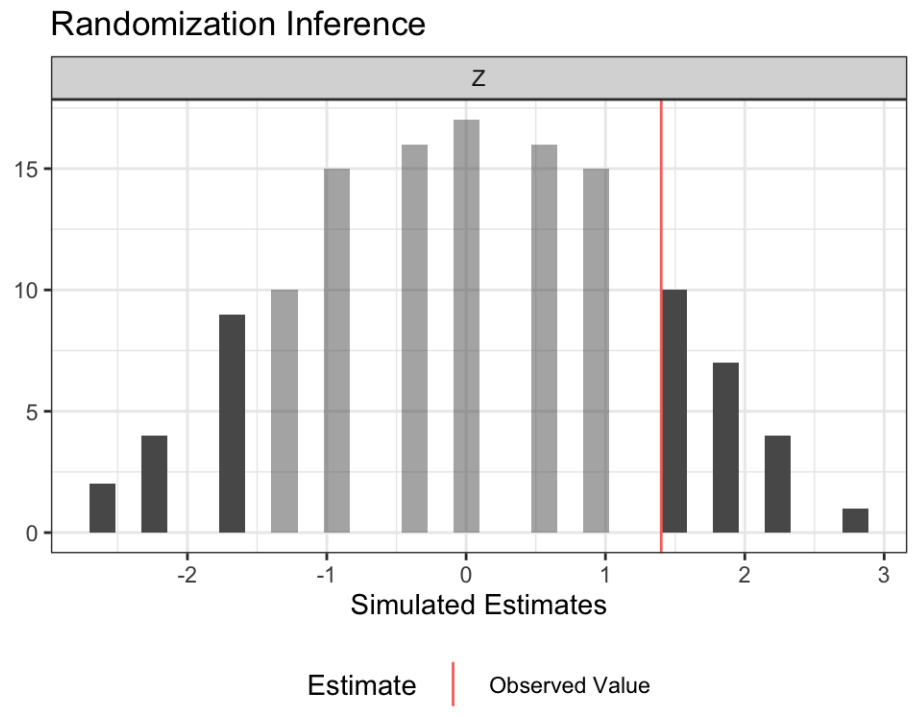
```
library(ri2)

# using the same toy example...
dat_table <- data.frame(Z = d, Y = Y)

# "declare" your randomization procedure: we have 9 observations, 5 of which are treated
declaration <- declare_ra(N = 9, m = 5)

# Conduct Randomization Inference
ri <- conduct_ri(
  formula = Y ~ Z,
  declaration = declaration,
  sharp_hypothesis = 0,
  data = dat_table
  # sims = 100
)
```

Look familiar?

```
summary(ri)
```

```
##   term estimate two_tailed_p_value
## 1    Z      1.4          0.2936508
```

```
plot(ri)
```



Summary of RI Procedure

- ADVANTAGES
- No distributional assumptions. Under what conditions would this be beneficial?
- Works for any test-statistic
- DISADVANTAGES
- The sharp null is often uninteresting (how often is a causal effect exactly zero for every single unit?)
- For large N, still relying on asymptotics.

Clustered Data and Cluster Randomization

- It's not always possible to assign treatment at the unit level is impossible. Instead, we assign treatment to clusters of units.
- Suppose we have units i where $i \in \{1, \ldots, N\}$, which are grouped in clusters j for $j \in \{1, \ldots, M\}$.
- Each cluster is equally sized, such that there are $\frac{N}{M}$ units in each.
- Here: $N = 1000$ units, $M = 50$ clusters, and cluster size is *1000/50 = 20*

Simulating Clustered Data

- Randomly assign half the clusters ($\frac{M}{2}$) to treatment.
- Induce intracluster correlation by creating a common mean for each cluster (`mu_j0`), so that the POs for each unit is correlated with those of other units in the same cluster:

$$Y_{ij}(0) \sim \mathcal{N}(\mu_{j0}, 1)$$

$$Y_{ij}(1) \sim \mathcal{N}(\mu_{j1}, 1)$$

where

$$\mu_{j0} \sim \mathcal{U}(0, 10)$$

and

$$\mu_{j1} \sim \mathcal{U}(6, 16)$$

Simulating Clustered Data

```r
N <- 1000
M <- 50

# Begin by creating a cluster indicator
J <- rep(c(1:M), each = 20)

# Draw group-level means for each cluster
mu_j0 <- runif(M, 0, 10)
mu_j1 <- runif(M, 6, 16)

# Calculate new values (potential outcomes under control and treatment) with clustered data
Y0 <- rnorm(N, mean = rep(mu_j0, each = 20), sd = 1)
Y1 <- rnorm(N, mean = rep(mu_j1, each = 20), sd = 1)
```

Simulating Clustered Data

- TRY TO DO THE FOLLOWING:

1. randomly select half of the clusters into treatment,
2. create a vector of 0s and 1s representing treatment assignment,
3. calculate the realized outcome What's the formula again?
4. get the ATE, $\mathbb{E}[Y_{ij}|D_{ij} = 1] - \mathbb{E}[Y_{ij}|D_{ij} = 0]$, using the difference in means estimator

Clustered Data and Cluster Randomization

1. randomly select half of the clusters into treatment

```
# randomly pick 25 of the 50 clusters to be assigned to treatment
D <- sample(unique(J), length(unique(J))/2)
D_j <- rep(0, length(unique(J)))
```

1. create a vector of 0s and 1s representing treatment assignment

```
# find the randomly chosen clusters and assign them to treatment (D = 1)
D_j[which(unique(J) %in% D)] <- 1
D_ij <- rep(D_j, each = length(J)/length(unique(J)))
```

Clustered Data and Cluster Randomization

1. compute the realized outcome for each unit i in cluster j : $Y = [Y_1 \times D] + [Y_0 \times (1 - D)]$

```
# calculate the realized outcome Y
Y_ij <- Y1*D_ij + Y0*(1-D_ij)

clustered_data <- data.frame(outcome = Y_ij, treated = D_ij, clusterid = J)
head(clustered_data)
```

```
##    outcome treated clusterid
## 1 4.003776       0         1
## 2 2.607063       0         1
## 3 5.292124       0         1
## 4 3.802461       0         1
## 5 3.369920       0         1
## 6 3.664236       0         1
```

Clustered Data and Cluster Randomization

- Calculate the difference in means

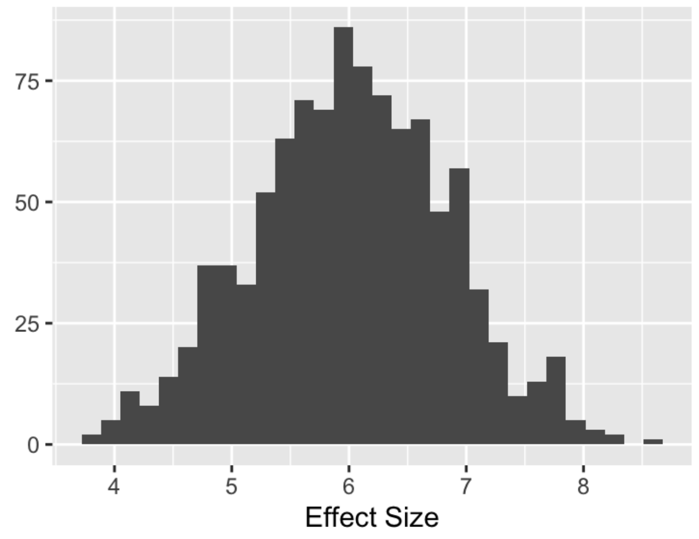
```
mean(clustered_data$outcome[clustered_data$treated == 1]) - mean(clustered_data$outcome[clustered_data$treated == 0]
```

```
## [1] 5.576706
```

```
# or t.test, or bivariate OLS, whatever floats your boat
#t.test(clustered_data$outcome[clustered_data$treated == 1], clustered_data$outcome[clustered_data$treated == 0])
```

Clustered Data and Cluster Randomization

- If we repeat the process 1,000 times (use `replicate()`), we get a distribution of simulated treatment effects.



- remember to `set.seed` for replicability!
- Experiment with different cluster sizes, sample sizes, and number of clusters. How do these factors change the variability of the simulated distribution?

Clustered Data and Cluster Randomization

- if `var_within` is small (implying similarity between units in a cluster), then ρ (ICC) [4] is large ($\rho \to 1$) and the design effect (ratio between variance under clustered vs. complete randomization)[5] is also large.
- The variance under cluster randomization is much greater than that under complete (individual) random assignment. Put differently, the efficiency penalty increases as unit outcomes within a cluster become more similar.
- Intuitively, when random assignment is done at the cluster level, greater similarity between observations within clusters means that we have "less information" (i.e. loss of statistical power) and the effective sample size diminishes.
- see Angrist and Phischke (p. 323) for derivation of Moulton Factor/Design Effect from a model-based approach (as opposed to experimental design, but intuition is similar).

4. $ICC = \rho = \frac{\sigma_B^2}{\sigma_W^2+\sigma_B^2}$ where σ_B^2 is between-cluster variance and the denominator is the total variance.

5. More precisely, is the variance of the estimator when the effect of clustering is taken into account over the variance of the estimator under the hypothesis of a simple random sample:

$$\frac{\mathbb{V}_{clu}(\beta)}{\mathbb{V}_{ols}(\beta)} = 1 + (N-1)\rho$$