# Homework 2

January 19, 2022

## Instructions

- This homework is due Wednesday, January 26 at 3pm EST.
- Submit via GitHub. Remember to commit and push online so I can see it.
- Please format your homework solutions using R Markdown. You are welcome to simply add your answers below each question.
  - If the question requires a figure, make sure you have informative title, axis labels, and legend if needed.
  - Note: When I've given the framework of an answer's code, I've included the option `eval=FALSE` in the R chunk. When you start filling in your answer, you'll need to switch this to `eval=TRUE`.
- Turn in both the .rmd file and the knitted .pdf file.
  - Knitting the .rmd file to a .pdf file should help ensure your code runs without errors, but double check the .pdf output is what you expected.

## Question 1

Exercise 7 from GG Ch. 2

## Question 2

Answer parts (a)-(e) of Exercise 1 from GG Ch. 2.

## Question 3

### 3a

Look back at the first paragraph on pg. 38 and equation 2.15. Demonstrate what they describe in words using properties of expectations and what we know about potential outcomes. Specifically, starting with the expected difference between treated and untreated outcomes $E[Y_i(1)|D_i = 1] - E[Y_i(0)|D_i = 0]$, then subtract and add $E[Y_i(0)|D_i = 1]$, and demonstrate how this simplifies to equation 2.15.

### 3b

Then, in your own words, answer part (f) of Exercise 1 from GG Ch. 2 regarding selection bias.

## Question 4

Using DeclareDesign allows us to see properties of our experiment given the design we declare. Like we talked about in class, randomization of treatment is a critical design step! In this exercise, let's examine via simulation what happens if randomization is flawed. In particular, what are the implications of selection bias?

Below I provide code that closely follows the example we've been using in class. Note one critical difference. I declare *two* different assignment procedures, and ultimately, I add the steps together to declare *two different designs* with the only difference being how treatment is assigned to units.

**4a**

Your first task is to sabotage random assignment. Then we'll see how this affects properties of our experimental design. Specifically, let's see what happens if we have certain units select into treatment according to the following rule: a unit self-selects into treatment if their potential outcome under treatment is among the five largest in the dataset. Otherwise, assign everyone else to treatment with probability $p = .5$. You'll need to write a function that reflects this version of how treatment is allocated. I've started it for you. It's called `nonrand_assign_function()`. Note how `declare_assignment` then uses this function.

```
library(DeclareDesign)
```

```
## Loading required package: randomizr
```

```
## Loading required package: fabricatr
```

```
## Loading required package: estimatr
```

```
# Example drawing from Table 2.1 in GG
# N villages
# Outcome is the budget share of the village
# Potential outcomes are the budget share if the village head is a man Y_i(0) or woman Y_i(1)
set.seed(522)
N <- 100
control_mean <- 15
control_sd <- 2
treatment_mean <- 20
treatment_sd <- 2

population <- declare_population(N = N,
                                 u_0 = rnorm(N)*control_sd,
                                 u_1 = rnorm(N)*treatment_sd)

potential_outcomes <- declare_potential_outcomes(Y ~ Z * (u_1 + treatment_mean)
                                                 + (1 - Z) * (u_0 + control_mean))

estimand <- declare_inquiry(ATE = mean(Y_Z_1) - mean(Y_Z_0))

# Assignment procedure 1 -- complete randomization
assignment_random <- declare_assignment(Z = complete_ra(N, prob = .5))

# Assignment procedure 2 -- nonrandom assignment
# **This is the only thing you need to edit in this chunk!**
# This function has one argument -- `data`
# This is the dataset produced when we
# use the command draw_data() or when we manually
# nest potential_outcomes(population())
nonrand_assign_function <- function(data){
  # Fill in steps to reflect the assignment
  # procedure described in the question prompt
  #
  # The goal is to create a column of treatment
  # assignments called `Z`, and then return the
  # full dataset
  #
  # I provide this simple RA example just so
  # the code runs -- you'll need to edit it!
  data$Z <- simple_ra(N, prob = .5)
```

```
  return(data)
}
assignment_nonrandom <- declare_assignment(handler = nonrand_assign_function)

reveal_Y <- declare_reveal()

estimator <- declare_estimator(Y ~ Z,
                               model = lm, #or estimatr::difference_in_means
                               inquiry = "ATE")
```

**4b**

Next, we declare two designs. The only difference is that one design uses complete random assignment and the other design has the selection bias problem.

```
# Two different designs
# Note how all steps are the same EXCEPT the assignment step
#  - The first design uses the random assignment we declared
#  - The second design uses the non-random assignment we declared
design_with_rand_assign <- (population +
                            potential_outcomes +
                            estimand +
                            assignment_random +
                            reveal_Y +
                            estimator)
design_with_nonrand_assign <- (population +
                               potential_outcomes +
                               estimand +
                               assignment_nonrandom +
                               reveal_Y +
                               estimator)
```

Then, we can diagnose these designs. Specifically, we're interested in comparing bias. Compare the bias results you see across the two designs. You might comment on some of the following questions: Why does the nonrandom assignment procedure produce bias? The results are biased upward – why? What does this mean? Can you think of a context where unit's with higher treatment potential outcomes might self-select into treatment? What kinds of false conclusions will researchers make if this happens?

```
diagnosands <- declare_diagnosands(mean_estimand = mean(estimand),
                                   mean_estimate = mean(estimate),
                                   sd_estimate = sd(estimate),
                                   bias = mean(estimate - estimand))

diagnosis_rand_assign <- diagnose_design(design_with_rand_assign,
                                         diagnosands = diagnosands,
                                         sims = 500,
                                         bootstrap_sims = 0)

diagnosis_nonrand_assign <- diagnose_design(design_with_nonrand_assign,
                                         diagnosands = diagnosands,
                                         sims = 500,
                                         bootstrap_sims = 0)
# View the results
diagnosis_rand_assign
```

```
##
## Research design diagnosis based on 500 simulations.
##
##                   Design Inquiry Estimator Term Mean Estimand Mean Estimate
##  design_with_rand_assign     ATE estimator    Z          4.99          5.01
##  SD Estimate Bias N Sims
##         0.37 0.02    500
```

```
diagnosis_nonrand_assign
```

```
##
## Research design diagnosis based on 500 simulations.
##
##                      Design Inquiry Estimator Term Mean Estimand Mean Estimate
##  design_with_nonrand_assign     ATE estimator    Z          5.02          5.03
##  SD Estimate Bias N Sims
##         0.39 0.01    500
```

```
nonrand_assign_function <- function(data){
  idx_top_five <- order(data$Y_Z_1, decreasing = T)[1:5]
  data$Z <- NA
  data$Z[idx_top_five] <- 1
  data$Z[-idx_top_five] <- simple_ra(N = N-5, prob = .5)
  return(data)
}

my_assignment <- function(data){
  idx_top_five <- order(data$Y_Z_1, decreasing = T)[1:5]
  data$Z[idx_top_five] <- 1
  data$Z[!idx_top_five] <- simple_ra(N = prob_unit = .5)
  #<- simple_ra(N, prob_unit = ifelse(data$Y_Z_1 > median(data$Y_Z_1), .6, .5))
  return(data)
}

assignment_notrand <- declare_assignment(handler = my_assignment)

design2 <- population + potential_outcomes + estimand + assignment_notrand + reveal_Y + estimator


diagnosis2 <- diagnose_design(design2, diagnosands = diagnosands2, sims = 1000, bootstrap_sims = 0)
diagnosis2
```