Simulation Study: Example 2

Ashley I Naimi

Spring 2024

Contents

1 Adjusted versus Unadjusted ITT Estimates

Let's take a look at another question we can answer with simulation: comparing adjusted to unadjusted estimates of the ITT effect in a randomized trial. Again, it would help us to get more specific with our intentions. For instance, we might ask whether we adjsuting for known causes of the outcome yields an improvement in the bias and efficiency of the ITT estimator.

2 Aims

To start, let's specify some aims of our simulation study. These will help guide how we write the code we need to execute the tasks to fulfill our aims:

- Compare the bias and efficiency (i.e., standard error performance) of an ITT estimator when we don't adjust, and adjust for a set of variables known to cause the outcome

3 Data Generating Mechanisms

In terms of generating our data, we will rely on a simple triangle DAG, with a set of confounding variables, a single exposure, and a single outcome:

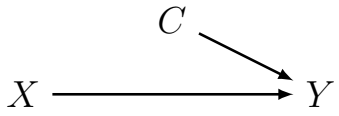


Figure 1: Simple causal diagram representing a randomized trial, with randomized treatmetn assignment A, outcome cause C, and outcome Y.

This gives us a relatively straightforward causal ordering, as follows:

- The variables C and A are equally exogenous
- The variable Y is the most endogenous

4 Estimands

We will focus our attention on a setting with a continuous outcome. Thus, we can compare the unadjusted and adjusted mean difference between the treated

and untreated:

$$E(Y \mid A = 1) - E(Y \mid A = 0)$$

Note that, in a well-conducted randomized trial, this is equivalent to the average treatment effect on the mean difference scale:

$$E(Y^{a=1} - Y^{a=0})$$

5 Methods

5.1 Unadjusted and Adjusted Linear Regression

With a continuous outcome and a binary randomized treatment variable, we can fit simple adjusted and unadjusted linear regression models to compute the ITT effect:

$$E(Y \mid A) = \alpha_0 + \alpha_1 A$$

$$E(Y \mid A, C) = \beta_0 + \beta_1 A + \beta_2 C$$

In these cases, provided the models are correctly specified, both α_1 and β_1 can be interpreted as the intention to treat effect.

6 Performance Measures

Once we obtain our simulation data, we will use the following performance measures to compare unadjusted and adjusted ITT effects:

- bias of the point estimate
- bias of the standard error

7 True Estimand Value

In this case, because we are interested in unadjusted and adjusted effects in a linear model, we do not need to rely on Monte Carlo integration to compute the true value. In fact, the true value will be whatever coefficient we choose to parameterize the treatment-outcome relationship in the linear model used to

simulate the outcome. Because the model is linear, and there is no confounding bias to adjust for, we can expect that the unadjusted and adjusted effects are equivalent.

Note that, in nonlinear models (such as logistic regression), this equivalence will not hold, in which case we'd need to use Monte Carlo integration to compute the true value.

8 Simulation Code

Once the true values are obtained, we can then run our simulation. Let's look through the code we'll use to do this:

```
library(parallel)
library(lmtest)
library(sandwich)

expit<-function(a){1/(1+exp(-a))}

set.seed(123)

simulation_function <- function(index,
                                sample_size = 500,
                                true_value = 2,
                                c_number = 10,
                                cov_mat = 0,
                                diag_cov = 1){

  # printing to console won't work with parallel processing
  print(index)

  # DATA GENERATION
  n <- sample_size
  print(n)

  # how many confounders to simulate?
```

```r
p <- c_number
print(p)

## confounder matrix
sigma <- matrix(cov_mat, nrow=p, ncol=p)
diag(sigma) <- diag_cov
c <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=sigma)

# DESIGN MATRIX FOR THE PROPENSITY SCORE MODEL
piMat <- model.matrix(
  as.formula(
    paste("~(",
          paste("c[,",1:ncol(c),"]", collapse="+"),
          ")"
    )
  )
)[,-1]

# simulate the treatment via 50:50 randomization
x <- rbinom(n, size = 1, p = .5)

# parameters for the covariate outcome relation
parmsC_mu <- rep(1.25, c_number)

# simulate the outcome
y <- rnorm(n, mean = 100 + true_value*x + piMat%*%parmsC_mu, sd = 2)

# construct dataset
analysis_data <- data.frame(y, x, c)

# ANALYSIS

mod1 <- lm(y ~ x, data = analysis_data)
unadjusted_effect <- summary(mod1)$coefficients[2,1:2]
```

```
  mod2 <- lm(y ~ ., data = analysis_data)
  adjusted_effect <- summary(mod2)$coefficients[2,1:2]

  # SIMULATION FUNCTION OUTPUT

  res <- data.frame(sample_size = sample_size,
                    c_number = c_number,
                    cov_mat  = cov_mat,
                    diag_cov = diag_cov,
                    true_value = true_value,
                    unadjusted_effect,
                    adjusted_effect)

  return(res)
}

simulation_results <- mclapply(1:5000,
                               function(x) simulation_function(index = x,
                                                               sample_size = 500,
                                                               true_value = 2,
                                                               c_number = 10,
                                                               cov_mat = 0,
                                                               diag_cov = 1),
                               mc.cores = detectCores() - 2)

sim_res <- do.call(rbind, simulation_results)

# # save the data to file!

write_csv(sim_res, here("lectures/06_Section6_SimulationInPractice", "simulation_results_rct.csv"))
```

9 Performance Evaluation

This simulation takes some time to run, so let's be sure we save our results to a file so that we can re-import them without having to run the simulation again to rebuild the entire dataset. We can import our saved data back into R using standard code, such as the `read` functions in the tidyverse:

```
a <- read_csv(here("lectures/06_Section6_SimulationInPractice", "simulation_results_rct.csv"))

head(a)
```

```
## # A tibble: 6 x 9
##   sample_size c_number cov_mat diag_cov true_value unadjusted_estimate
##         <dbl>    <dbl>   <dbl>    <dbl>      <dbl>               <dbl>
## 1         250        5       0        1          2                1.48
## 2         250        5       0        1          2                2.28
## 3         250        5       0        1          2                1.23
## 4         250        5       0        1          2                2.88
## 5         250        5       0        1          2                2.31
## 6         250        5       0        1          2                1.39
## # i 3 more variables: unadjusted_estimate_se <dbl>, adjusted_estimate <dbl>,
## #   adjusted_estimate_se <dbl>
```

Let's start with some basic analyses. We'll look at the mean of the unadjusted and adjusted mean differences, which should be 2 across all parameter specifications:

```
a %>%
  group_by(sample_size, c_number, cov_mat) %>%
  summarize(meanAdj = mean(adjusted_estimate),
            meanUnAdj = mean(unadjusted_estimate))
```

```
## # A tibble: 18 x 5
## # Groups:   sample_size, c_number [9]
##    sample_size c_number cov_mat meanAdj meanUnAdj
##          <dbl>    <dbl>   <dbl>   <dbl>     <dbl>
```

```
##  1         250        5     0      2.00      2.00
##  2         250        5     0.5    2.00      1.99
##  3         250       10     0      2.00      2.00
##  4         250       10     0.5    2.00      1.99
##  5         250       15     0      2.00      2.01
##  6         250       15     0.5    2.00      1.99
##  7         500        5     0      2.00      2.00
##  8         500        5     0.5    2.00      2.00
##  9         500       10     0      2.00      2.00
## 10         500       10     0.5    2.00      2.01
## 11         500       15     0      2.00      2.00
## 12         500       15     0.5    2.00      1.97
## 13        1000        5     0      2.00      2.00
## 14        1000        5     0.5    2.00      2.00
## 15        1000       10     0      2.00      2.00
## 16        1000       10     0.5    2.00      2.00
## 17        1000       15     0      2.00      1.99
## 18        1000       15     0.5    2.00      2.00
```

These results show what we'd expect to see: that, on average, the point estimates are centered on the true value. What about the standard errors?:

```
a %>%
  group_by(sample_size, c_number, cov_mat) %>%
  summarize(meanAdjSE = mean(adjusted_estimate_se),
            meanUnAdjSE = mean(unadjusted_estimate_se))
```

```
## # A tibble: 18 x 5
## # Groups:   sample_size, c_number [9]
##    sample_size c_number cov_mat meanAdjSE meanUnAdjSE
##          <dbl>    <dbl>   <dbl>     <dbl>       <dbl>
##  1         250        5     0       0.256       0.435
##  2         250        5     0.5     0.256       0.663
##  3         250       10     0       0.259       0.561
##  4         250       10     0.5     0.258       1.20
##  5         250       15     0       0.261       0.663
```

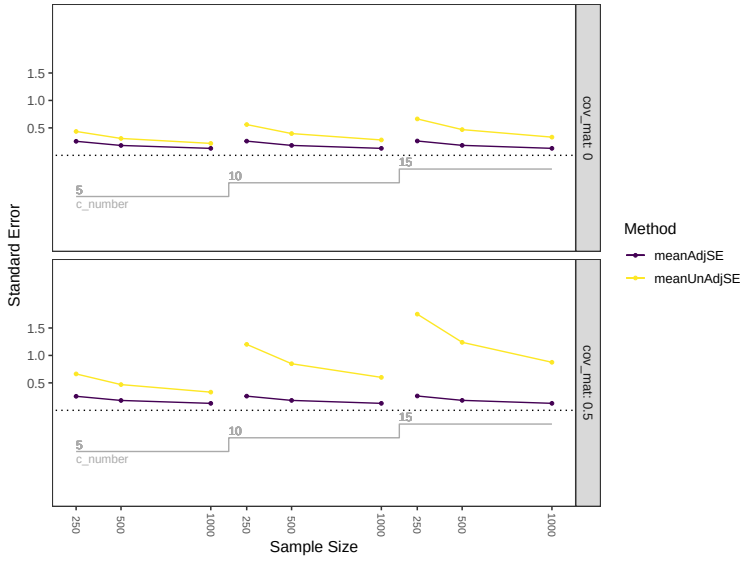
```
##  6         250       15     0.5     0.261       1.75
##  7         500        5     0       0.180       0.308
##  8         500        5     0.5     0.180       0.469
##  9         500       10     0       0.181       0.396
## 10         500       10     0.5     0.181       0.849
## 11         500       15     0       0.182       0.469
## 12         500       15     0.5     0.182       1.24
## 13        1000        5     0       0.127       0.217
## 14        1000        5     0.5     0.127       0.331
## 15        1000       10     0       0.127       0.280
## 16        1000       10     0.5     0.127       0.600
## 17        1000       15     0       0.127       0.331
## 18        1000       15     0.5     0.127       0.875
```

This is difficult to interpret specifically, but clearly the standard error estimator for the unadjusted ITT estimator is much larger than the adjusted ITT estimator. Let's visualize these results using a nested loop plot:

```
plot_res <- a %>%
  group_by(sample_size, c_number, cov_mat) %>%
  summarize(meanAdjSE = mean(adjusted_estimate_se),
            meanUnAdjSE = mean(unadjusted_estimate_se))

pacman::p_load(looplot)

p = nested_loop_plot(resdf = plot_res,
                     x = "sample_size", steps = "c_number", grid_rows = "cov_mat",
                     steps_y_base = -.25, steps_y_height = .25, steps_y_shift = .1,
                     x_name = "Sample Size", y_name = "Standard Error",
                     spu_x_shift = 200,
                     steps_values_annotate = TRUE, steps_annotation_size = 3,
                     hline_intercept = 0,
                     y_expand_add = c(1, NULL),
                     post_processing = list(
                       add_custom_theme = list(
                         axis.text.x = element_text(angle = -90,
```

```
                                                    vjust = 0.5,
                                                    size = 8)
                        ))
)

ggsave(here("_images", "nested_loop_plot_rct.pdf"), width = 8, height = 6)
```

Here, we can more clearly see the differences between adjusted and unadjusted ITT estimators when different numbers of covariates are adjusted for under different covariate correlation structures across our three sample sizes:



Finally, it's important to note three subtleties here. **First,** the standard errors for the adjusted an unadjusted estimators are performing as expected. They are accurately capturing the variability in the point estimates.

We can confirm this by comparing the standard deviation of the point estimates from the adjusted and unadjusted approaches to the means of their respective standard error estimates:

```
a %>%
  group_by(sample_size, c_number, cov_mat) %>%
  summarize(meanAdjSE = mean(adjusted_estimate_se),
            sdAdj = sd(adjusted_estimate)) %>%
  mutate(se_bias = sdAdj - meanAdjSE)
```

```
## # A tibble: 18 x 6
## # Groups:   sample_size, c_number [9]
##    sample_size c_number cov_mat meanAdjSE sdAdj    se_bias
##          <dbl>    <dbl>   <dbl>     <dbl> <dbl>      <dbl>
##  1         250        5     0       0.256 0.258  0.00239
##  2         250        5     0.5     0.256 0.254 -0.00148
##  3         250       10     0       0.259 0.261  0.00251
##  4         250       10     0.5     0.258 0.260  0.00144
##  5         250       15     0       0.261 0.263  0.00217
##  6         250       15     0.5     0.261 0.261  0.000223
##  7         500        5     0       0.180 0.180  0.0000391
##  8         500        5     0.5     0.180 0.182  0.00242
##  9         500       10     0       0.181 0.181  0.0000503
## 10         500       10     0.5     0.181 0.181  0.000524
## 11         500       15     0       0.182 0.181 -0.000726
## 12         500       15     0.5     0.182 0.181 -0.000655
## 13        1000        5     0       0.127 0.128  0.000762
## 14        1000        5     0.5     0.127 0.127 -0.000357
## 15        1000       10     0       0.127 0.129  0.00207
## 16        1000       10     0.5     0.127 0.127  0.000120
## 17        1000       15     0       0.127 0.128  0.000211
## 18        1000       15     0.5     0.127 0.128  0.000130
```

```
a %>%
  group_by(sample_size, c_number, cov_mat) %>%
  summarize(meanUnadjSE = mean(unadjusted_estimate_se),
            sdUnadj = sd(unadjusted_estimate)) %>%
  mutate(se_bias = sdUnadj - meanUnadjSE)
```

```
## # A tibble: 18 x 6
## # Groups:   sample_size, c_number [9]
##    sample_size c_number cov_mat meanUnadjSE sdUnadj   se_bias
##          <dbl>    <dbl>   <dbl>       <dbl>   <dbl>     <dbl>
```

```
##  1          250         5      0         0.435   0.440  0.00497
##  2          250         5      0.5       0.663   0.661 -0.00220
##  3          250        10      0         0.561   0.563  0.00259
##  4          250        10      0.5       1.20    1.21   0.00812
##  5          250        15      0         0.663   0.664  0.000805
##  6          250        15      0.5       1.75    1.74  -0.00757
##  7          500         5      0         0.308   0.304 -0.00334
##  8          500         5      0.5       0.469   0.468 -0.00103
##  9          500        10      0         0.396   0.398  0.00148
## 10          500        10      0.5       0.849   0.851  0.00239
## 11          500        15      0         0.469   0.471  0.00200
## 12          500        15      0.5       1.24    1.22  -0.0161
## 13         1000         5      0         0.217   0.218  0.000975
## 14         1000         5      0.5       0.331   0.334  0.00243
## 15         1000        10      0         0.280   0.282  0.00212
## 16         1000        10      0.5       0.600   0.606  0.00545
## 17         1000        15      0         0.331   0.330 -0.00142
## 18         1000        15      0.5       0.875   0.887  0.0120
```

Fundamentally, what is happening here is that the adjusted estimator is simply more efficient than the unadjusted estimator. However, both are correct.

Second, our illustration is limited in that our outcome is simulated from a linear model. Because of this, we can place the adjusted and unadjusted point estimates on equal footing as they are both targeting the same estimand. That is, because of the combination of randomization and linearity of the outcome model, the unadjusted and adjusted estimator are mathematically equivalent.

If our outcome was simulated from a nonlinear model, such as logistic regression, we'd have to take into consideration noncollapsibility here, which would add a considerable degree of complexity to this our simulation study.

Third and final, we did not explore the consequences of misspecification of the adjusted model in our simulations. To illustrate this point, suppose there were several interactions between the randomized treatment and the covariates in the data generating mechanism that we **did not know to acccount for** in the analytic model used to adjust our ITT estimator. Alternatively, suppose that a subset of the covariates were related to the outcome in complex (e.g., curvilinear) ways, and we failed to account for this in our analytic model because we

did not know these relationships were present.

In this case, it is possible that using a misspecified model to obtain an adjusted estimator results in bias of the ITT effect, which obviates the whole purpose of deploying a randomized trial. This is one reason why unadjusted ITT effects are most commonly reported in the literature, even though adjusted estimators are more efficient. In effect, efficiency is a second order property, while bias is a first order property.