Simple Binary Bootstrap Full Simulation (Clean)

Waveley

2022-02-14

Generating Data

First, let's generate some data.

```
set.seed(20220214)
pre_data <-
    defData(
      varname = "L1",
      formula = "0",
      variance = 1,
      dist = "normal"
      )

pre_data <-
    defData(
      pre_data,
      varname = "L2",
      formula = "0",
      variance = 1,
      dist = "normal"
      )

pre_data <-
  defData(
    pre_data,
    varname = "L3",
    formula = "0",
    variance = 1,
    dist = "normal"
    )

pre_data <-
  defData(
    pre_data,
    varname = "A",
    formula = "0.5*L1 + 0.27*L2 - 0.17*L3",
    dist = "binary",
    link = "logit"
    )

pre_data <-
  defData(
    pre_data,
    varname = "Y",
    formula = "0.5*A + 0.8*L2 -0.1*L3",
    dist = "binary",
    link = "logit")

# establish seed_vec for 1000 runs of noboot data

seed_vec <- runif(10000, min = 20220214, max = 202202140) %>% round(0) %>% unique()

# function to generate however many noboot datasets at whatever size
generate_noboot_data <- function(n, size = 5000, seeds = seed_vec, defData = pre_data) {
  df <- list()

  # adding progress bar for sanity
  pb <- progress_bar$new(format = "generating data... [:bar]", total = n)

  for (i in 1:n) {
    pb$tick()
    set.seed(seeds[i])
    df[[i]] <- genData(size, defData)
  }

  return(df)
}

df <- generate_noboot_data(100)
```

Propensity Score Modeling and NNM

Now, we will perform propensity score modeling and nearest neighbor matching.

```
# could possible turn this into a function later.
matched_df <- list()

for (i in 1:length(df)) {

  # not sure if exposuremodel is needed
  exposureModel <- glm(A ~ L1 + L2 + L3, data = df[[i]], family = "binomial")

  # getting estimated propensity score, not sure if this is needed
  df[[i]]$ps <- predict(exposureModel, df[[i]], type = "response")

  matched <- matchit(A ~ L1 + L2 + L3,
                          data = df[[i]],
                          distance = "glm",
                          link = "logit",
                          method = "nearest",
                          ratio = 1) # perform NNM
  matched_df[[i]] <- match.data(matched)
}
```

Empirical Treatment Effect SE (and Mean)

Taken as standard deviation of estimated treatment effect (β_1) from rlength(df)‘ generated “noboot” datasets.

```
## # A tibble: 1 x 2
##    mean simulated_se
##   <dbl>        <dbl>
## 1 0.501       0.0610
```

Simple Bootstrap

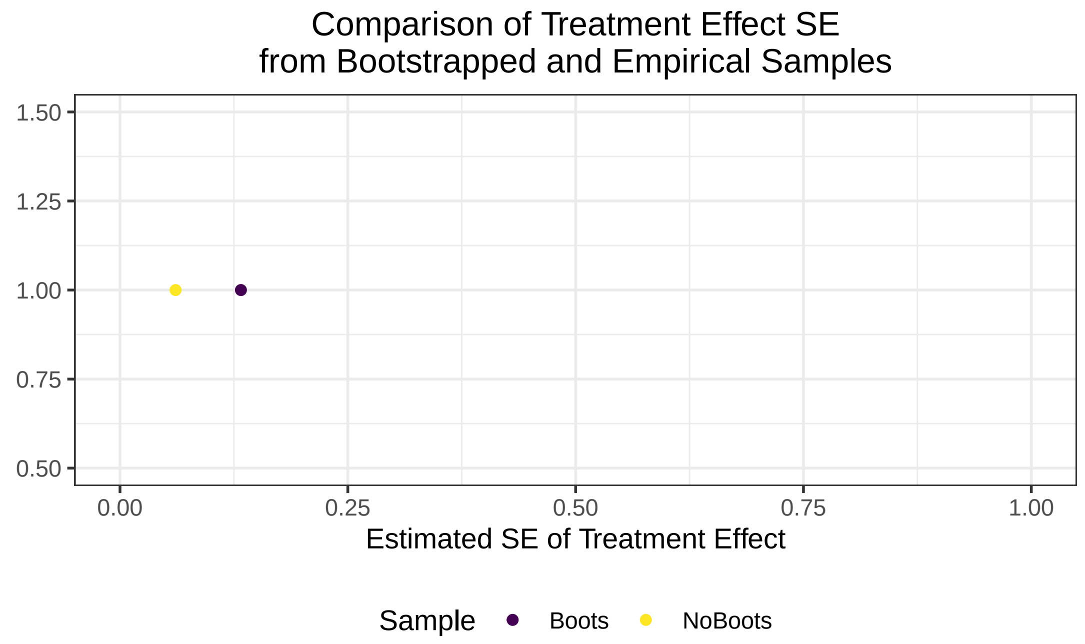
```
# ### function to iterate glm over a list, to be used in purr:map ###
# returns tibble of parameter estimates and standard errors.

outcome_model_list <- function(list) {
  tib_coef <- tibble()
  pb3$tick()
  for (i in 1:length(list)) {
    mod <- glm(Y ~ A + L2 + L3,
               data = list[[i]],
               weights = weights,
               family = "binomial") %>%
      summary()
    coefs <- mod$coefficients[2,1:2]
    tib_coef <- bind_rows(tib_coef, tibble(estimate = coefs[1], se = coefs[2]))
  }
    return(tib_coef)
}

# ### input matched dataframe, output however many bootstrapped samples you want ###
# first, set seed vector for reproducibility

seed_vec_2 <- runif(10000, min = 20220215, max = 202202150) %>% round(0) %>% unique()

# now, define function

simple_boot <- function(df, n, size = 500, seeds = seed_vec_2){
  boots <- list()
  pb2$tick()
  for (i in 1:n) {
  set.seed(seeds[i])
  boots[[i]] <-
    df %>%
    filter(subclass %in% sample(levels(subclass),
                                size,
                                replace =  TRUE))
  }
  return(boots)
}

# adding progress bars for sanity
pb2 <- progress_bar$new(format = "bootstrapping... [:bar]", total = nrow(df_tib))
pb3 <- progress_bar$new(format = "performing glm... [:bar]", total = nrow(df_tib))
```

```r
# creating booted tibbles, applying functions through purr:map.

boot_tib <-
  df_tib %>%
  mutate(
    boots = map(.x = data, ~simple_boot(.x, n=10)),
    ) %>%
  mutate(coef = map(.x = boots, ~outcome_model_list(.x)))

# extracting glm parameter estimates
boot_estimates <-
  boot_tib %>%
  mutate(seq = seq(1:nrow(boot_tib))) %>%
  select(coef, seq) %>% unnest(coef)

# calculating mean and standard deviation of estimates to estimate standard error
boot_out <-
  boot_estimates %>%
  group_by(seq) %>%
  summarize(simulated_se = sd(estimate),
            mean = mean(estimate))

# preparing the data to be plotted
boot_mean_se <-
  boot_out %>%
  summarize(fin_sim_se = mean(simulated_se)) %>%
  mutate(x = fin_sim_se, y = 1, label = "Boots")

boot_out %>% select(mean, simulated_se)
```

```
## # A tibble: 100 x 2
##     mean simulated_se
##    <dbl>        <dbl>
##  1 0.453       0.0843
##  2 0.541       0.119
##  3 0.641       0.116
##  4 0.439       0.0898
##  5 0.467       0.134
##  6 0.413       0.160
##  7 0.430       0.124
##  8 0.427       0.121
##  9 0.438       0.142
## 10 0.513       0.148
## # ... with 90 more rows
```

Maybe a Plot?

```r
to_gg <- bind_rows(boot_mean_se, empirical_out)

to_gg %>% ggplot(aes(x = x, y = y, col = label)) +
  geom_point() +
  xlim(0, 1) +
  ylim(0.5, 1.5) +
  labs(title = "Comparison of Treatment Effect SE\nfrom Bootstrapped and Empirical Samples",
       x = "Estimated SE of Treatment Effect", y = '', col = 'Sample')
```


Comparison of Treatment Effect SE
from Bootstrapped and Empirical Samples
Estimated SE of Treatment Effect
Sample
Boots
NoBoots