Binary_Simulation_v2

Hun

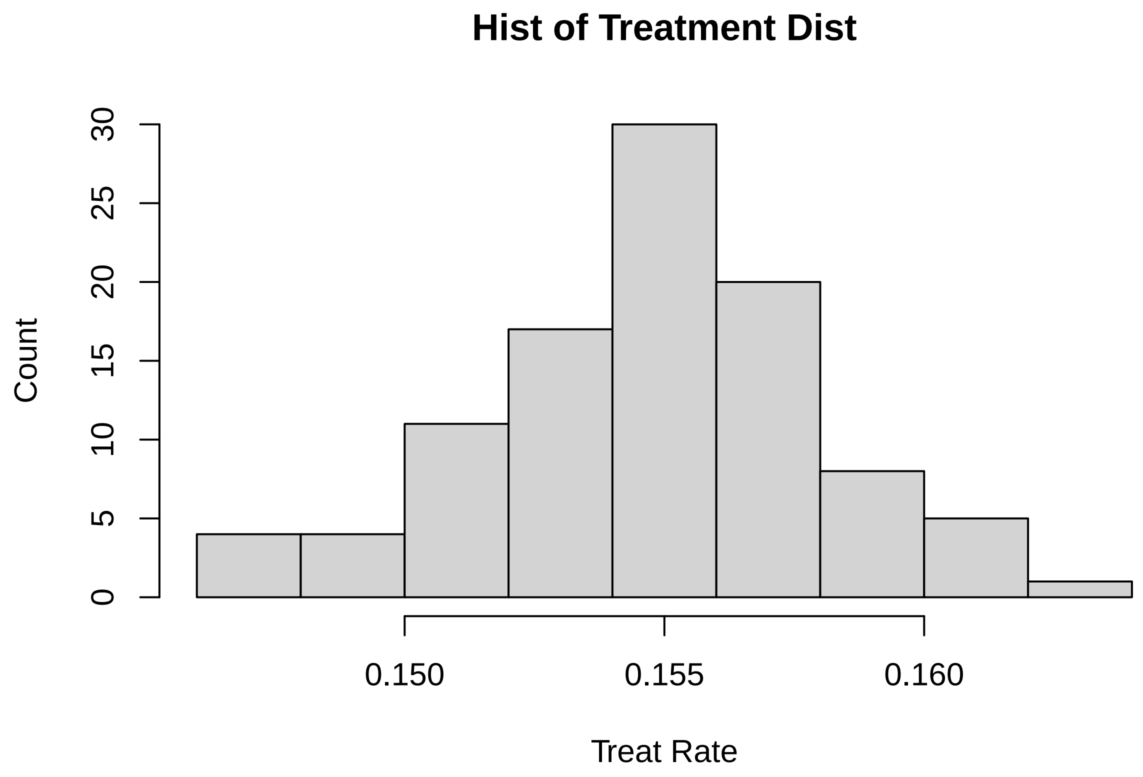
2/15/2022

```r
source('./shared_code/data_gen_final_pretty.R')
```

Function for finding the optimal combintion of coefficits for covariates to have desired proportion of the treated.

checking dist of treatment rate.

```r
hist(map_dbl(1:length(no_boot_list), function(i) mean(no_boot_list[[i]]$A)),
     main = "Hist of Treatment Dist",
     xlab = "Treat Rate",
     ylab = "Count")
```

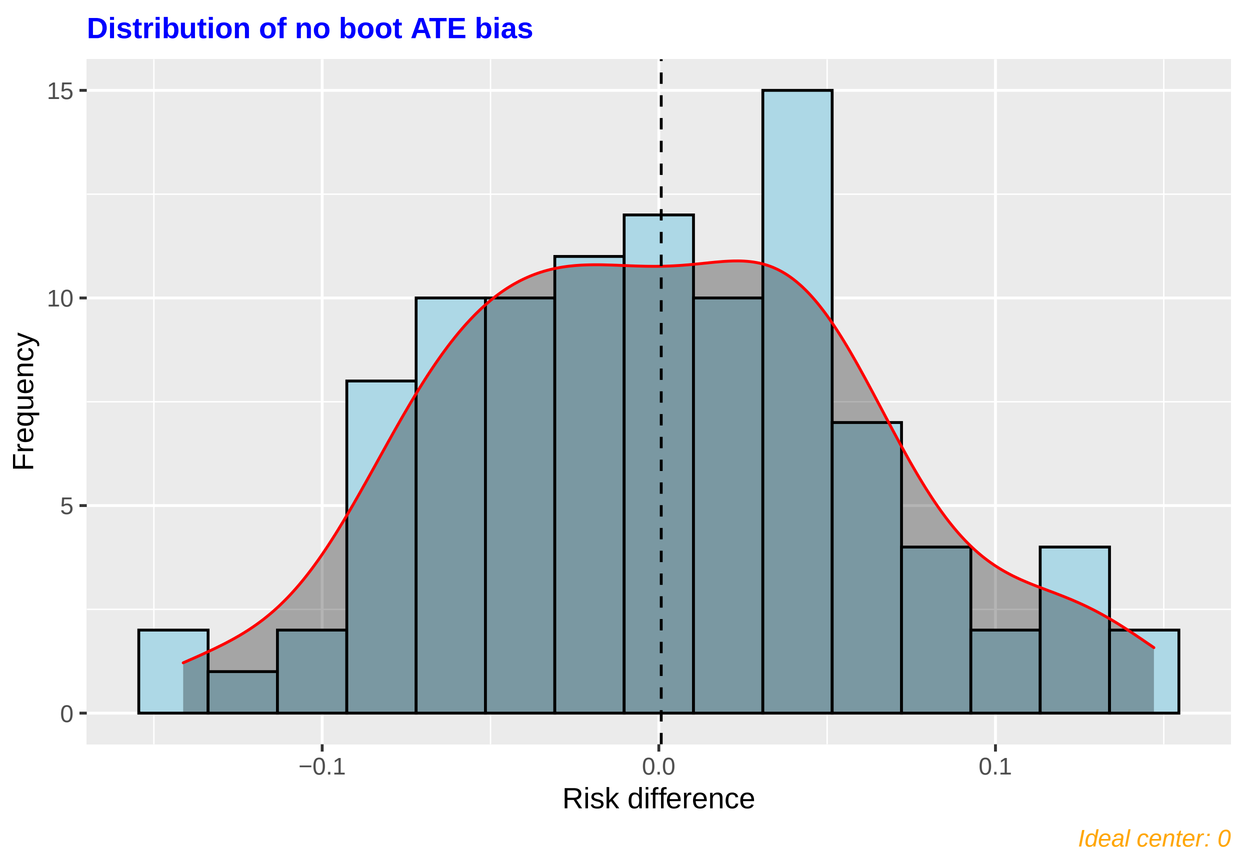


Function for no boot bias distribution

```r
no_boot_result <-
  function(n)  {
  b <- vector()
    for (i in 1:n) {
  a <- glm(Y ~ A + L2 + L3, data = no_boot_list[[i]], family = "binomial")
  b[i] <- summary(a)$coef[2,1]
}
return(b)
  }

no_boot_OR <- no_boot_result(100)
```

No boot Bias plot

```r
no_boot_bias <-
  tibble(no_boot_OR = no_boot_OR) %>%
  summarise(bias = (no_boot_OR - 0.767))

no_boot_bias %>%
  ggplot(aes(x = bias)) +
  geom_histogram(fill = "light blue", bins = 15, color = "black") +
  geom_density(aes(y = ..density..*2), colour = "red",
               fill = "black", alpha = 0.3) +
  geom_vline(xintercept = mean(no_boot_bias$bias), linetype = "dashed") +
  labs(title = "Distribution of no boot ATE bias",
       caption = "Ideal center: 0", x = "Risk difference", y = "Frequency") +
  theme(
  plot.title = element_text(color = "blue", size = 11, face = "bold"),
  plot.caption = element_text(color = "orange", face = "italic")
  )
```



Implementing nearest-neighbor matching (NNM)

```r
df <- no_boot_list
pb4 <- progress_bar$new(format = "bootstrapping... [:bar] :percent eta: :eta",
                        total = length(df))

# could possible turn this into a function later.
matched_df <- list()
# mean_treat_save <- rep(0, length(df))
# match_save <- rep(0, length(df))
for (i in 1:length(df)) {
  pb4$tick()
  matched <- matchit(A ~  L2 + L3,
                        data = df[[i]],
                        distance = "glm",
                        link = "logit",
                        method = "nearest",
                        ratio = 1) # perform NNM

  matched_df[[i]] <- match.data(matched, distance = "ps")
  # mean_treat_save[i] <- (5000*mean(df[[i]]$A)*2)
  # match_save[i] <- (match.data(matched, distance = "ps")  %>%
  #                       select(subclass) %>% unique() %>% summarize(n=n()) %>% pull(n))*2
}
```

simple Bootstrap

```r
# creating the tibble to apply map function
matched_tib <-
  tibble(data = matched_df)

# ### 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()
  boots <- tibble(mean1 = NA,
                  mean0 = NA,
                  difference = NA)
  pb3$tick()
  for (i in 1:length(list)) {
    mod <- glm(Y ~ A + ps,
               data = list[[i]],
               weights = weights,
               family = "binomial")

    sampl_all_treated <- list[[i]] %>%
    mutate(A = 1)

    sampl_all_untreated <- list[[i]] %>%
    mutate(A = 0)

    sampl_all_treated$pred.y <-
    predict(mod, sampl_all_treated, type = "response")

    sampl_all_untreated$pred.y <-
    predict(mod, sampl_all_untreated, type = "response")

    boots[i, "mean1"] <- mean(sampl_all_treated$pred.y)
    boots[i, "mean0"] <- mean(sampl_all_untreated$pred.y)
    boots[i, "difference"] <- boots[i, "mean1"] - boots[i, "mean0"]
    #coefs <- mod$coefficients[2,1:2]
    #tib_coef <- bind_rows(tib_coef, tibble(estimate = coefs[1]))
  }
    return(boots)
}
```

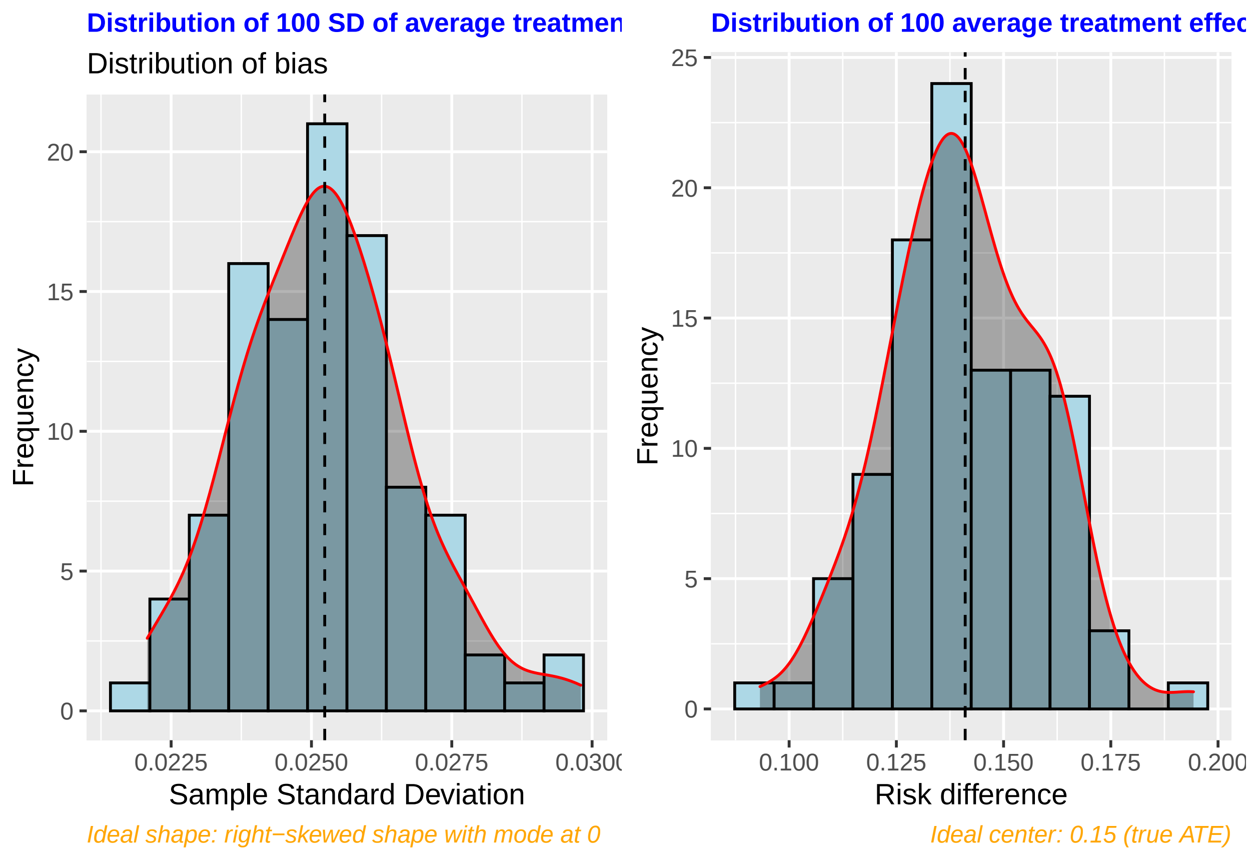
```r
# ### input matched dataframe, output however many bootstrapped samples you want ###
# first, set seed vector for reproducibility


# now, define function

seed_vec_2 <- rnorm(100000, mean = 0, sd = 10000) %>% round(0) %>% unique()

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$new(format = "bootstrapping... [:bar] :percent eta: :eta",
                        total = length(df))
pb3 <- progress_bar$new(format = "bootstrapping... [:bar] :percent eta: :eta",
                        total = length(df))

# creating booted tibbles, applying functions through purr:map.
boot_tib <-
  matched_tib %>%
  mutate(
    boots = map(.x = data, ~simple_boot(.x, n = 100))
    ) %>%
  mutate(ATE = map(.x = boots, ~outcome_model_list(.x)))
```

Unnesting bootstrap coefficients

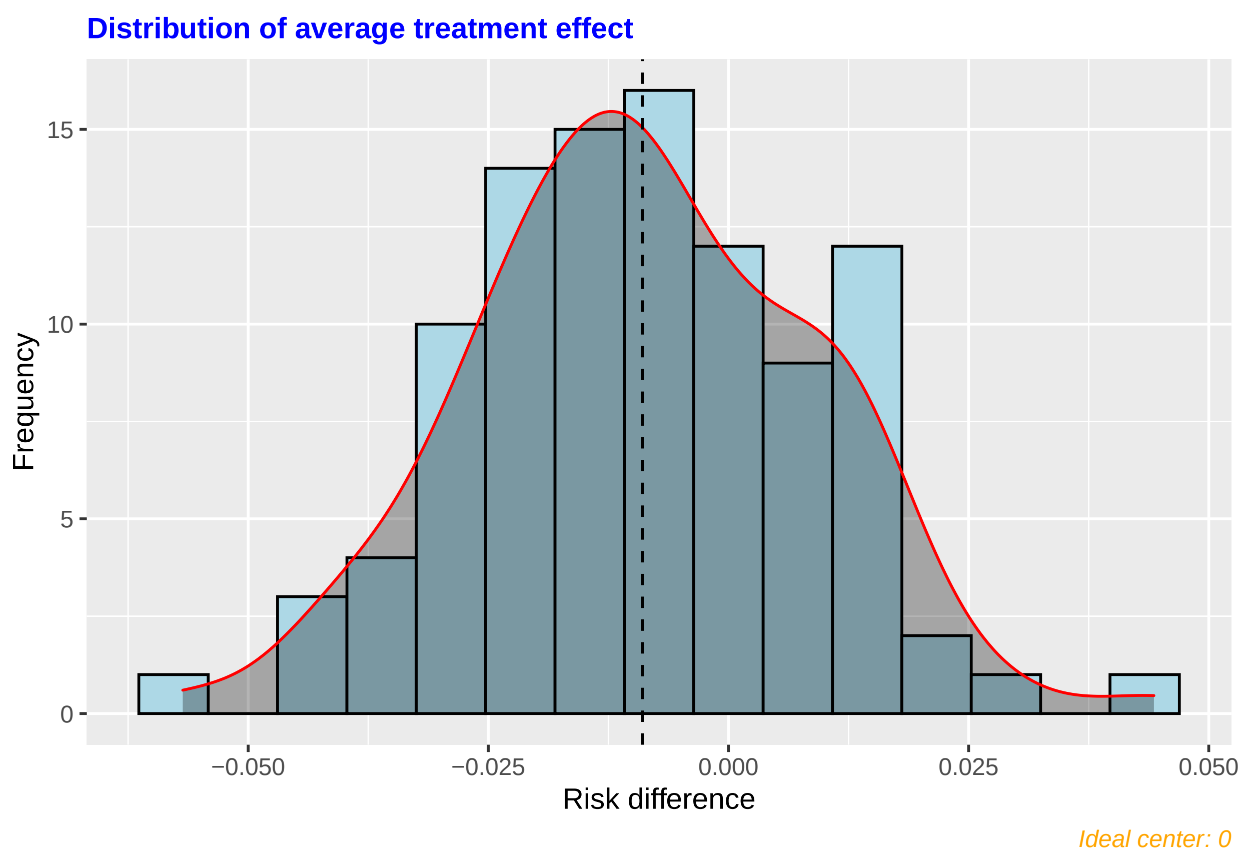
```r
boot_estimates <-
  boot_tib %>%
  mutate(seq = seq(1:nrow(boot_tib))) %>%
  select(ATE, seq) %>% unnest(ATE)
```

Summary of 1000 bootstrap of 100 no boot

```r
boot_result <-
  boot_estimates %>%
  group_by(seq) %>%
  summarize(mean_ATE = mean(difference),
            sd_mean_ATE = sd(difference),
            bias = mean_ATE - 0.15)

fig1 <-
  boot_result %>%
  ggplot(aes(x = sd_mean_ATE)) +
  geom_histogram(fill = "light blue",  bins = 12,  color = "black") +
  geom_density(aes(y = ..density..*0.07), colour = "red",
               fill = "black", alpha = 0.3) +
  geom_vline(xintercept = mean(boot_result$sd_mean_ATE), linetype = "dashed") +
  labs(title = "Distribution of 100 SD of average treatment effect",
       subtitle = "Distribution of bias",
       x = "Sample Standard Deviation", y = "Frequency",
       caption = "Ideal shape: right-skewed shape with mode at 0 ") +
  theme(
  plot.title = element_text(color = "blue", size = 10, face = "bold"),
  plot.subtitle = element_text(color = "black"),
  plot.caption = element_text(color = "orange", face = "italic")
  )


fig2 <-
  boot_result %>%
  ggplot(aes(x = mean_ATE)) +
  geom_histogram(fill = "light blue", bins = 12, color = "black") +
  geom_density(aes(y = ..density..*1), colour = "red",
               fill = "black", alpha = 0.3) +
  geom_vline(xintercept = mean(boot_result$mean_ATE), linetype = "dashed") +
  labs(title = "Distribution of 100 average treatment effect",
       caption = "Ideal center: 0.15 (true ATE)",
       x = "Risk difference", y = "Frequency") +
  theme(
  plot.title = element_text(color = "blue", size = 10, face = "bold"),
  plot.caption = element_text(color = "orange", face = "italic")
  )



plot_grid(fig1, fig2)
```



```r
fig3 <-
  boot_result %>%
  ggplot(aes(x = bias)) +
  geom_histogram(fill = "light blue", bins = 15, color = "black") +
  geom_density(aes(y = ..density..*0.7), colour = "red",
               fill = "black", alpha = 0.3) +
  geom_vline(xintercept = mean(boot_result$bias), linetype = "dashed") +
  labs(title = "Distribution of average treatment effect",
       caption = "Ideal center: 0", x = "Risk difference", y = "Frequency") +
  theme(
  plot.title = element_text(color = "blue", size = 11, face = "bold"),
  plot.caption = element_text(color = "orange", face = "italic")
  )

fig3
```



100 Confidence Intervals Coverage Rate using central limit theorem

```
CI_coverage_rate <- function(nboot){
  boot_CI_ATE <- list()
  count_true <- list()

  for (i in 1:nboot) {
     boot_CI_ATE[[i]] <-
       c(boot_result$mean_ATE[i] + qnorm(0.025)*sd(boot_result$mean_ATE),
         boot_result$mean_ATE[i] + qnorm(0.975)*sd(boot_result$mean_ATE))

      count_true[i] <-
         between(0.15, range(boot_CI_ATE[[i]])[1], range(boot_CI_ATE[[i]])[2])
  }

  result =
    as.vector(unlist(count_true)) %>%
    sum()
  result = paste0("CI coverage rate:", result, "%")

  return(result)
}

CI_coverage_rate(100)
```

```
## [1] "CI coverage rate:92%"
```

Sample mean central limit theorem to build confidence interval

```
(qnorm(0.025) * sd(boot_result$mean_ATE)) + 0.15 #Lower CI range
```

```
## [1] 0.1153242
```

```
(qnorm(0.975) * sd(boot_result$mean_ATE)) + 0.15 #Upper CI range
```

```
## [1] 0.1846758
```