

Binary_Simulation_v2

Hun

2/15/2022

Function for finding the optimal combination of coefficients for covariates to have desired proportion of the treated.

```
covariate_coef <- function(desired_prop, cov_df) {  
  
  alpha_0 = log(desired_prop/(1 - desired_prop))  
  
  coef_L1 <- sample(cov_df$L1, 10000, replace = TRUE)  
  coef_L2 <- sample(cov_df$L2, 10000, replace = TRUE)  
  coef_L3 <- sample(cov_df$L3, 10000, replace = TRUE)  
  
  A_logit <- vector(mode = "list", length = length(coef_L2))  
  p_A <- vector(mode = "numeric", length = length(coef_L2))  
  
  u <- alpha_0 + coef_L1[1]*cov_df$L1 + coef_L2[1]*cov_df$L2 - coef_L3[1]*cov_df$L3  
  
  p_A[1] <- mean(exp(u)/(1 + exp(u)))  
  
  tol <- 0.001  
  
  i = 1  
  
  while (abs(p_A[i] - 0.14) > tol) {  
  
    i = i + 1  
  
    A_logit[[i]] <-  
      alpha_0 + coef_L1[i]*cov_df$L1 + coef_L2[i]*cov_df$L2 - coef_L3[i]*cov_df$L3  
  
    p_A[i] <- mean(exp(A_logit[[i]])/(1 + exp(A_logit[[i]])))  
  
    if (abs(p_A[i] - 0.14) < tol) {  
      mean_treated_proportion <- p_A[i]  
      desired_coef_L1 <- coef_L1[i]  
      desired_coef_L2 <- coef_L2[i]  
      desired_coef_L3 <- coef_L3[i]  
    }  
  }  
}
```

```

    if (i > length(coef_L2)) {
      stop("You need better coverage, mate.")
    }
  }

  return(tibble(alpha_0, mean_treated_proportion,
                desired_coef_L1, desired_coef_L2, desired_coef_L3))
}

```

Generating 100 no boot samples

```

seed_vec <- rnorm(100000, mean = 0, sd = 100) %>% round(0) %>% unique()

generate_no_boot_data <- function(n, size = 5000, seeds = seed_vec) {

  df <- list()

  cov_df <- list()

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

  for (i in 1:n) {
    pb$tick()
    set.seed(seeds[i])

    set.seed(seeds[i])
    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")

    cov_df[[i]] <- genData(5000, pre_data)

    cov_coef_df <- covariate_coef(0.14, cov_df[[i]])

    L1_coef <- cov_coef_df$desired_coef_L1
    L2_coef <- cov_coef_df$desired_coef_L2
    L3_coef <- cov_coef_df$desired_coef_L3

    pre_data <- defData(pre_data, varname = "L1_coef", formula = L1_coef)
    pre_data <- defData(pre_data, varname = "L2_coef", formula = L2_coef)
    pre_data <- defData(pre_data, varname = "L3_coef", formula = L3_coef)

    pre_data <- defData(pre_data, varname = "A",
                        formula = "-1.815 + L1_coef*L1 + L2_coef*L2 + L3_coef*L3",
                        dist = "binary", link = "logit")
  }
}

```

```

pre_data <- defData(pre_data, varname = "Y",
                    formula = "0.437 + 0.767*A + 1.39*L2 -0.7*L3" ,
                    dist = "binary", link = "logit")

df[[i]] <- genData(size, pre_data)
df[[i]] <- df[[i]] %>% select(-L1_coef, -L2_coef, -L3_coef)
}
return(df)
}

no_boot_list <- generate_no_boot_data(100)

```

Function for no boot bias distribution

```

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)

```

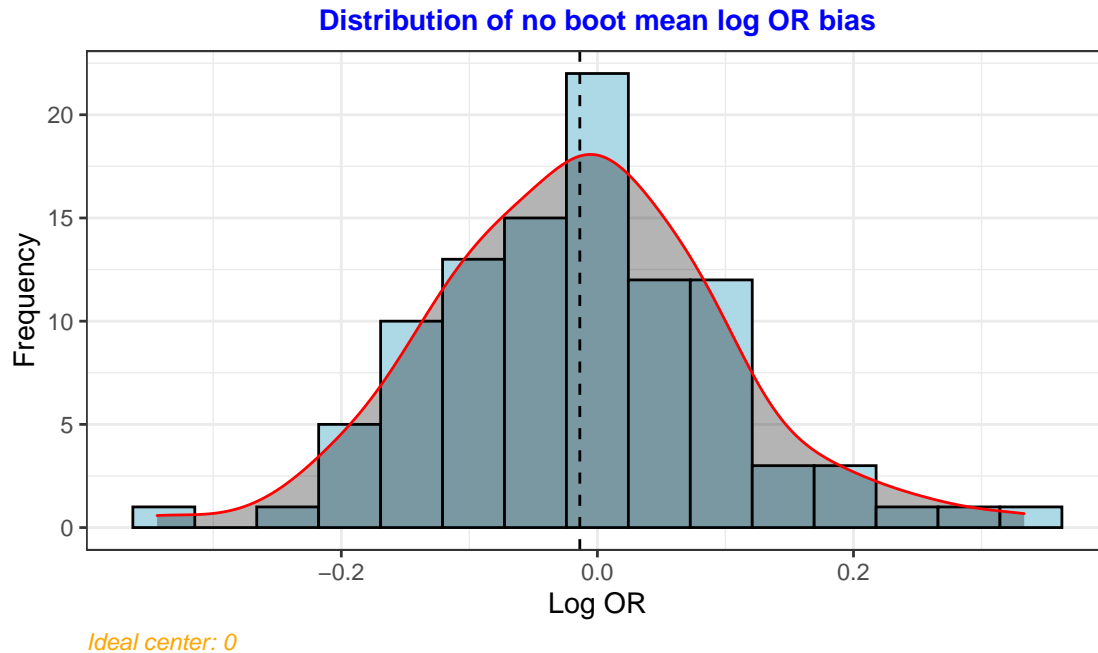
No boot Bias plot

```

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..*5), colour = "red",
               fill = "black", alpha = 0.3) +
  geom_vline(xintercept = mean(no_boot_bias$bias), linetype = "dashed") +
  labs(title = "Distribution of no boot mean log OR bias",
       caption = "Ideal center: 0", x = "Log OR", 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)

```
df <- no_boot_list

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

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

  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, distance = "ps")
}
```

simple Bootstrap

```
# 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()
  pb3$tick()
  for (i in 1:length(list)) {
    mod <- glm(Y ~ A + ps,
              data = list[[i]],
              weights = weights,
              family = "binomial") %>% summary()
    coefs <- mod$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

# 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$new(format = "bootstrapping... [:bar]", total = nrow(matched_tib))
pb3 <- progress_bar$new(format = "performing glm... [:bar]", total = nrow(matched_tib))

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

```

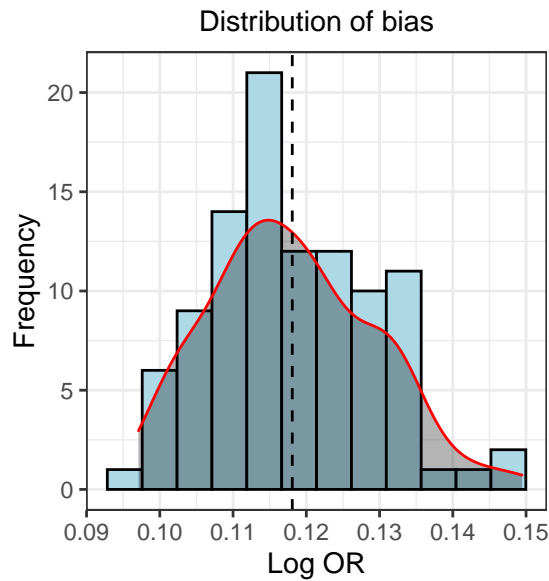
Unnesting bootstrap coefficients

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

Summary of 1000 bootstrap of 100 no boot

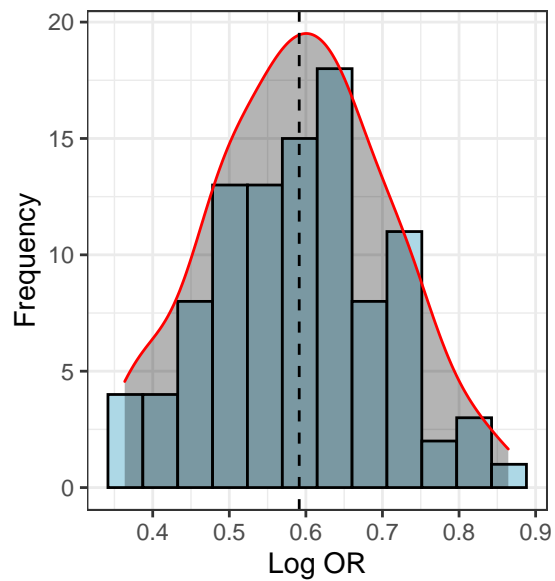
```
boot_result <-  
  boot_estimates %>%  
  group_by(seq) %>%  
  summarize(mean_log_OR = mean(estimate), sd_log_OR = sd(estimate),  
            bias = (mean_log_OR - 0.767))  
fig1 <-  
  boot_result %>%  
  ggplot(aes(x = sd_log_OR, color = sd_log_OR)) +  
  geom_histogram(fill = "light blue", bins = 12, color = "black") +  
  geom_density(aes(y = ..density..*0.4), colour = "red",  
              fill = "black", alpha = 0.3) +  
  geom_vline(xintercept = mean(boot_result$sd_log_OR), linetype = "dashed") +  
  labs(title = "Distribution of 100 SD of mean log OR",  
       subtitle = "Distribution of bias", x = "Log OR", y = "Frequency",  
       caption = "Ideal shape: right-skewed shape with mode at 0 ") +  
  theme(  
    plot.title = element_text(color = "blue", size = 11, face = "bold"),  
    plot.subtitle = element_text(color = "black"),  
    plot.caption = element_text(color = "orange", face = "italic")  
  )  
  
fig2 <-  
  boot_result %>%  
  ggplot(aes(x = mean_log_OR)) +  
  geom_histogram(fill = "light blue", bins = 12, color = "black") +  
  geom_density(aes(y = ..density..*6), colour = "red",  
              fill = "black", alpha = 0.3) +  
  geom_vline(xintercept = mean(boot_result$mean_log_OR), linetype = "dashed") +  
  labs(title = "Distribution of 100 mean log OR",  
       caption = "Ideal center: 0.767 (true log OR)", x = "Log OR", y = "Frequency") +  
  theme(  
    plot.title = element_text(color = "blue", size = 11, face = "bold"),  
    plot.caption = element_text(color = "orange", face = "italic")  
  )  
  
plot_grid(fig1, fig2)
```

Distribution of 100 SD of mean log OI



Ideal shape: right-skewed shape with mode a

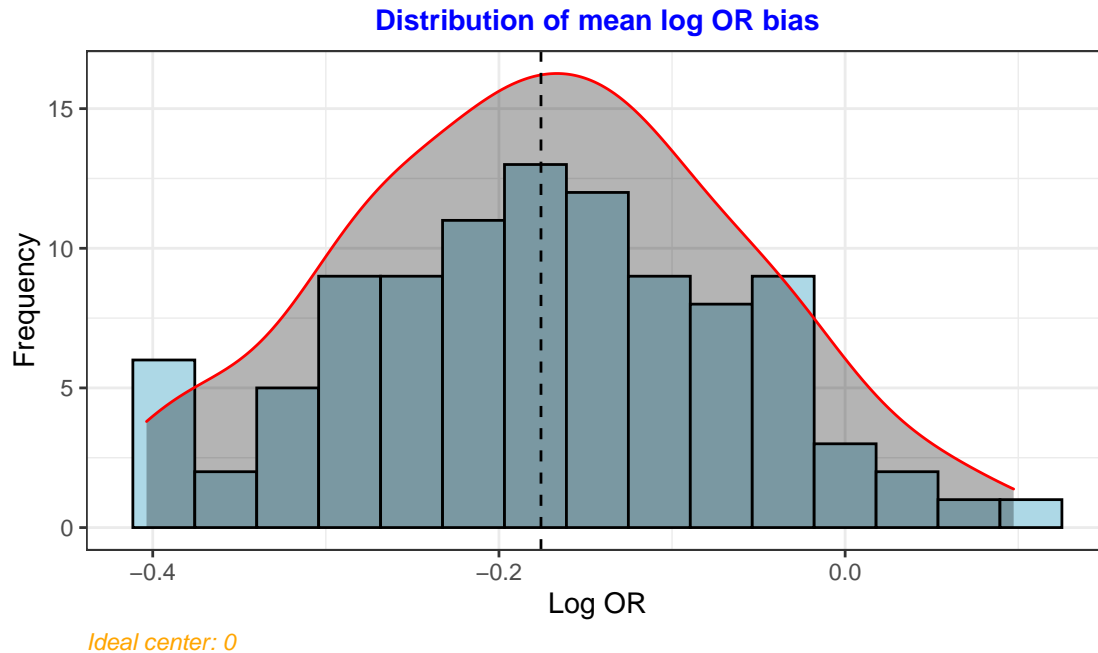
Distribution of 100 mean log OR



Ideal center: 0.767 (true log OR)

```
fig3 <-
  boot_result %>%
  ggplot(aes(x = bias)) +
  geom_histogram(fill = "light blue", bins = 15, color = "black") +
  geom_density(aes(y = ..density..*5), colour = "red",
               fill = "black", alpha = 0.3) +
  geom_vline(xintercept = mean(boot_result$bias), linetype = "dashed") +
  labs(title = "Distribution of mean log OR bias",
       caption = "Ideal center: 0", x = "Log OR", 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

```
CI_coverate_rate <- function(nboot){
  boot_CI_log_OR <- list()
  count_true <- list()

  for (i in 1:nboot) {
    boot_CI_log_OR[[i]] <-
      c(boot_result$mean_log_OR[i] - 1.96*boot_result$sd_log_OR,
        boot_result$mean_log_OR[i] + 1.96*boot_result$sd_log_OR)

    count_true[i] <-
      between(0.767, range(boot_CI_log_OR[[i]])[1], range(boot_CI_log_OR[[i]])[2])
  }

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

  return(result)
}

CI_coverate_rate(100)
```

```
## [1] "CI coverage rate:84%"
```