Continuous Simulation

Tucker Morgan - tlm2152

2/16/2022

Generating Covariates and Finding Coefficients in Propensity Model

```
set.seed(100)
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 Sub-Populations

```
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 = ".5 + 0.15*A + 2*L2 - 1*L3" ,
                    dist = "nonrandom")

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

ate_true = 0.15
no_boot_list <- generate_no_boot_data(100)
```

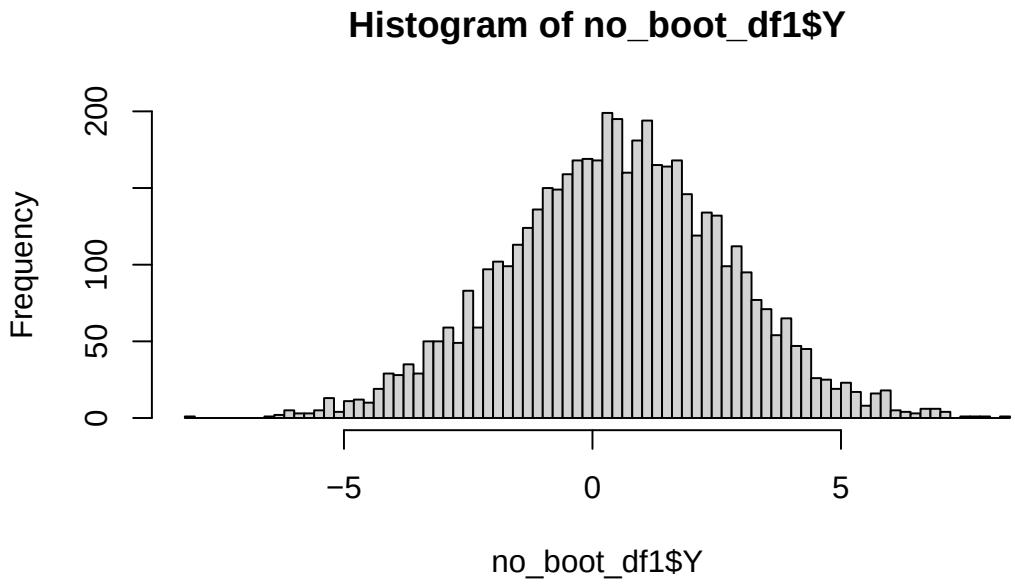
Let's take a look at one of our "no-boot" aka sub-population data sets.

```
no_boot_df1 <- no_boot_list[[1]]

sum(no_boot_df1$A) / nrow(no_boot_df1) # similar to 0.14
```

```
## [1] 0.1412
```

```
hist(no_boot_df1$Y, breaks = 100) # continuous distribution of outcome
```



Implementing Nearest-Neighbor Matching

```
df <- no_boot_list
```

```r
# 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")
}
```

Again, we'll look at one of our matched sub-pops.

```r
matched_df1 <- matched_df[[1]]
summary(matched_df1)
```

```
##        id             L1                  L2                 L3
##  Min.   :   1   Min.   :-3.573438   Min.   :-3.54470   Min.   :-3.72605
##  1st Qu.:1264   1st Qu.:-0.649672   1st Qu.:-0.69187   1st Qu.:-0.62017
##  Median :2443   Median :-0.007697   Median :-0.05612   Median : 0.07297
##  Mean   :2478   Mean   : 0.000126   Mean   :-0.05797   Mean   : 0.08463
##  3rd Qu.:3690   3rd Qu.: 0.683522   3rd Qu.: 0.62451   3rd Qu.: 0.77775
##  Max.   :5000   Max.   : 3.163505   Max.   : 3.04954   Max.   : 4.76319
##
##        A             Y                  ps            weights     subclass
##  Min.   :0.0   Min.   :-8.1366   Min.   :0.1028   Min.   :1   1      :   2
##  1st Qu.:0.0   1st Qu.:-1.0756   1st Qu.:0.1329   1st Qu.:1   2      :   2
##  Median :0.5   Median : 0.4035   Median :0.1421   Median :1   3      :   2
##  Mean   :0.5   Mean   : 0.3744   Mean   :0.1425   Mean   :1   4      :   2
##  3rd Qu.:1.0   3rd Qu.: 1.8915   3rd Qu.:0.1507   3rd Qu.:1   5      :   2
##  Max.   :1.0   Max.   : 7.7998   Max.   :0.2071   Max.   :1   6      :   2
##                                                               (Other):1400
```

```r
str(matched_df1)
```

```
## Classes 'matchdata', 'data.table' and 'data.frame':  1412 obs. of  9 variables:
##  $ id      : int  1 2 5 9 13 19 21 23 24 32 ...
##  $ L1      : num  0.594 -1.289 0.738 -1.947 -0.684 ...
##  $ L2      : num  -1.011 -0.162 -0.922 -1.158 0.465 ...
##  $ L3      : num  -0.212 -0.747 -1.866 -0.458 -0.358 ...
##  $ A       : int  1 1 1 1 1 1 1 0 1 0 ...
##  $ Y       : num  -1.16 1.073 0.673 -1.209 1.938 ...
##  $ ps      : num  0.148 0.131 0.13 0.14 0.132 ...
##  $ weights : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ subclass: Factor w/ 706 levels "1","2","3","4",..: 1 172 634 692 64 160 190 186 234 235 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "distance")= chr "ps"
##  - attr(*, "weights")= chr "weights"
##  - attr(*, "subclass")= chr "subclass"
```

The 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$tick()
  for (i in 1:length(list)) {
    mod <- glm(Y ~ A + ps,
               data = list[[i]],
               weights = weights) %>% summary()
    coefs <- mod$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 = 1000))
    ) %>%
  mutate(coef = map(.x = boots, ~outcome_model_list(.x)))
```

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

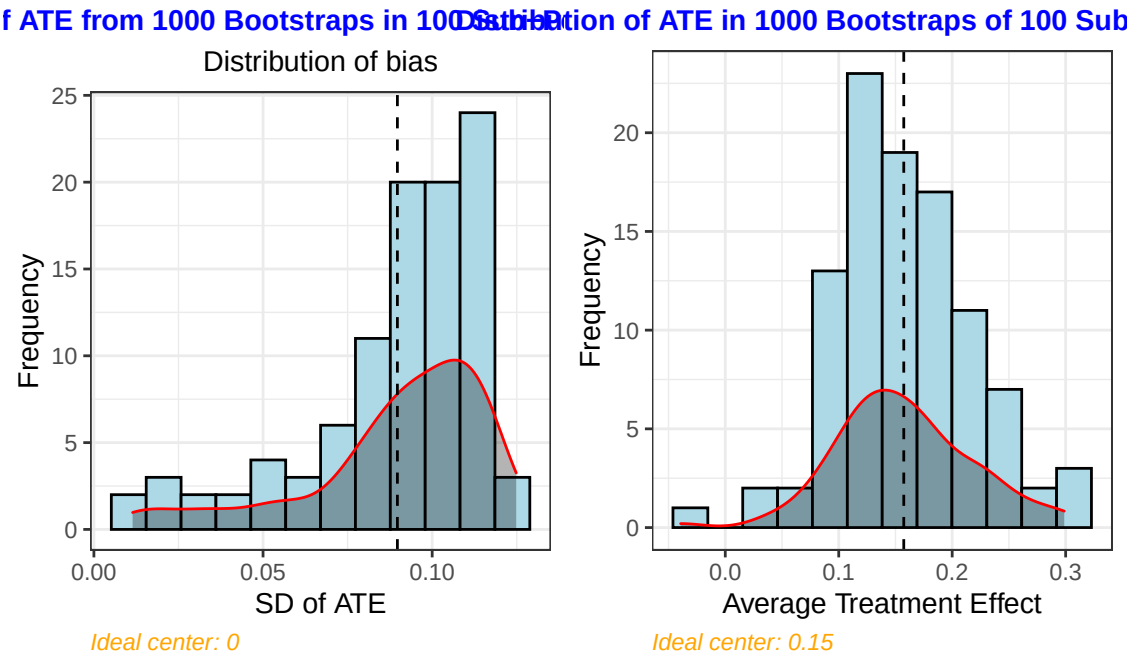
Summary of 1000 Bootstraps in 100 Sub-Populations

```
boot_result <-
  boot_estimates %>%
  group_by(seq) %>%
  summarize(avg_trt_eff = mean(estimate), sd_ate = sd(estimate))

fig1 <-
  boot_result %>%
  ggplot(aes(x = sd_ate, color = sd_ate)) +
  geom_histogram(fill = "light blue",  bins = 12,  color = "black") +
  geom_density(aes(y = ..density..*0.5), colour = "red",
               fill = "black", alpha = 0.3) +
  geom_vline(xintercept = mean(boot_result$sd_ate), linetype = "dashed") +
  labs(title = "SD of ATE from 1000 Bootstraps in 100 Sub-Populations",
       subtitle = "Distribution of bias",
  caption = "Ideal center: 0", x = "SD of ATE", y = "Frequency") +
  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 = avg_trt_eff)) +
  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$avg_trt_eff), linetype = "dashed") +
  labs(title = "Distribution of ATE in 1000 Bootstraps of 100 Sub-Populations",
       caption = "Ideal center: 0.15", x = "Average Treatment Effect", 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)
```



Confidence Intervals Coverage Rates

```
cvg_rate <- function(df){
  res = df %>%
    mutate(ci_low = avg_trt_eff - 1.96*sd_ate,
          ci_high = avg_trt_eff + 1.96*sd_ate,
          covered = case_when(
            ci_low <= ate_true & ci_high >= ate_true ~ 1,
                                                TRUE ~ 0
          ))

  return(sum(res$covered) / nrow(res))
}

cvg_plot <- function(df){
  res = df %>%
    mutate(ci_low = avg_trt_eff - 1.96*sd_ate,
          ci_high = avg_trt_eff + 1.96*sd_ate,
          covered = case_when(
            ci_low <= ate_true & ci_high >= ate_true ~ 1,
                                                TRUE ~ 0
          ))

  plot = res %>%
    ggplot(aes(x = avg_trt_eff, y = seq)) +
    geom_point() +
    geom_errorbar(aes(xmin = ci_low, xmax = ci_high)) +
    geom_vline(xintercept = ate_true, linetype = "dashed")

  return(plot)
```

```
}
```

```
cvg_rate(boot_result)
```

```
## [1] 0.99
```

```
cvg_plot(boot_result)
```