Continuous Simulation

Tucker Morgan - tlm2152

2/16/2022

Generating 100 Samples from Our Population

```
source("./shared_code/data_gen_continuous.R")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: ggpp

##
## Attaching package: 'ggpp'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

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

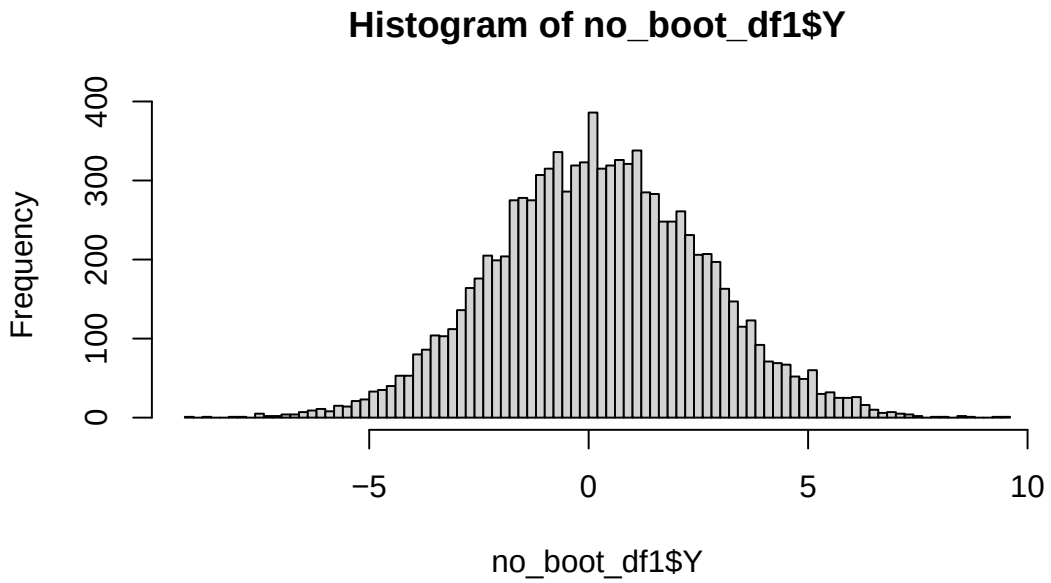
```
hist(map_dbl(1:length(no_boot_list), function(i) mean(no_boot_list[[i]]$A)  ),
     main = "Hist of Treatment Dist",
     xlab = "Treat Rate",
     ylab = "Count")
```


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

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

```
## [1] 0.2084
```

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



Implementing Nearest-Neighbor Matching

```
df <- no_boot_list

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

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

  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")
}
```

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

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

```
##        id             L1                  L2                L3
##  Min.   :   4   Min.   :-3.66694   Min.   :-3.7597   Min.   :-3.515842
##  1st Qu.:2562   1st Qu.:-0.64338   1st Qu.:-0.2258   1st Qu.:-0.687918
##  Median :5049   Median : 0.03557   Median : 0.3776   Median : 0.002274
##  Mean   :5078   Mean   : 0.04462   Mean   : 0.3922   Mean   :-0.006075
##  3rd Qu.:7600   3rd Qu.: 0.73305   3rd Qu.: 1.0144   3rd Qu.: 0.675937
##  Max.   :9999   Max.   : 4.01756   Max.   : 3.8289   Max.   : 3.436101
##
##        A              Y                  ps              weights     subclass
##  Min.   :0.0   Min.   :-9.1561   Min.   :0.03351   Min.   :1   1      :   2
##  1st Qu.:0.0   1st Qu.:-0.2429   1st Qu.:0.17833   1st Qu.:1   2      :   2
##  Median :0.5   Median : 1.2568   Median :0.22907   Median :1   3      :   2
##  Mean   :0.5   Mean   : 1.2713   Mean   :0.24118   Mean   :1   4      :   2
##  3rd Qu.:1.0   3rd Qu.: 2.7839   3rd Qu.:0.29305   3rd Qu.:1   5      :   2
##  Max.   :1.0   Max.   : 9.4518   Max.   :0.64207   Max.   :1   6      :   2
##                                                                (Other):4156
```

```
str(matched_df1)
```

```
## Classes 'matchdata', 'data.table' and 'data.frame':  4168 obs. of  9 variables:
##  $ id      : int  4 5 6 7 10 16 18 19 21 22 ...
##  $ L1      : num  0.346 0.686 1.217 1.715 1.271 ...
##  $ L2      : num  -0.259 -1.062 -0.799 1.285 0.796 ...
##  $ L3      : num  -0.411 0.573 1.84 0.597 0.714 ...
##  $ A       : int  0 0 1 0 1 0 0 1 1 1 ...
##  $ Y       : num  -0.591 -1.392 0.63 4.436 3.216 ...
##  $ ps      : num  0.178 0.122 0.133 0.318 0.265 ...
```

```
##  $ weights : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ subclass: Factor w/ 2084 levels "1","2","3","4",..: 350 1309 1142 1256 1 950 490 202 253 280 ...
##  - 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 = m_boot, 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)
}
```

```r
# 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 = n_sample * desired_prop))
    ) %>%
  mutate(coef = map(.x = boots, ~outcome_model_list(.x)))
```

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

Summary of 1000 Bootstraps in 100 Sub-Populations

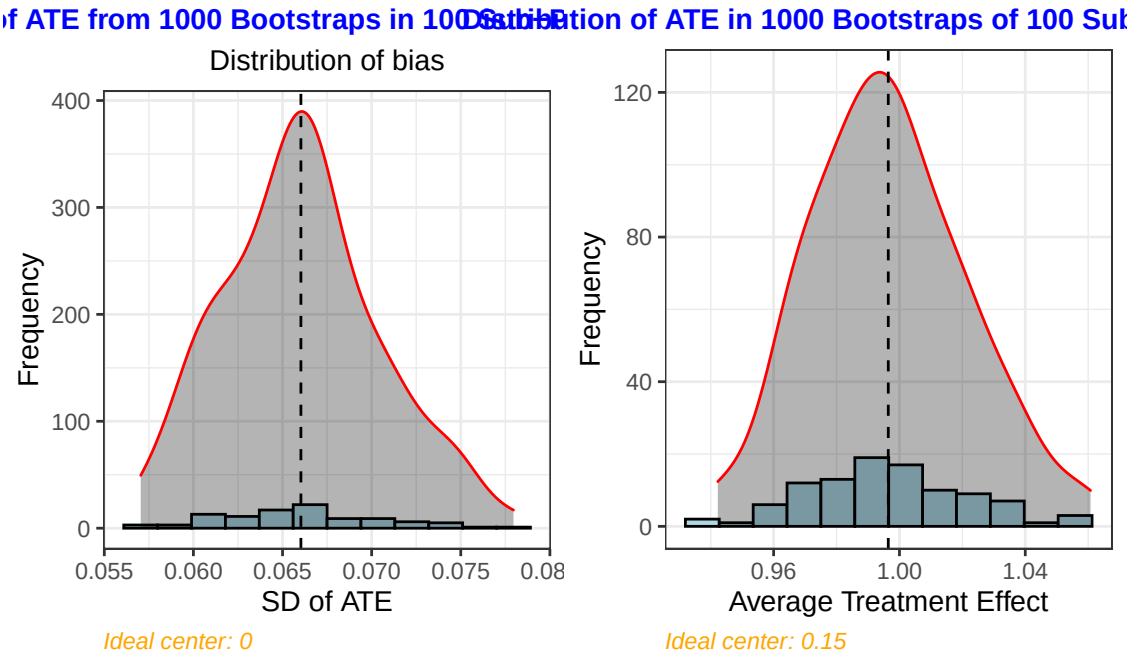
```r
boot_result <-
  boot_estimates %>%
  group_by(seq) %>%
  summarize(avg_trt_eff = mean(estimate), sd_ate = sd(estimate))

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..*4), 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..*8), 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 <= beta1 & ci_high >= beta1 ~ 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 <= beta1 & ci_high >= beta1 ~ 1,
                                          TRUE ~ 0
          ))
```

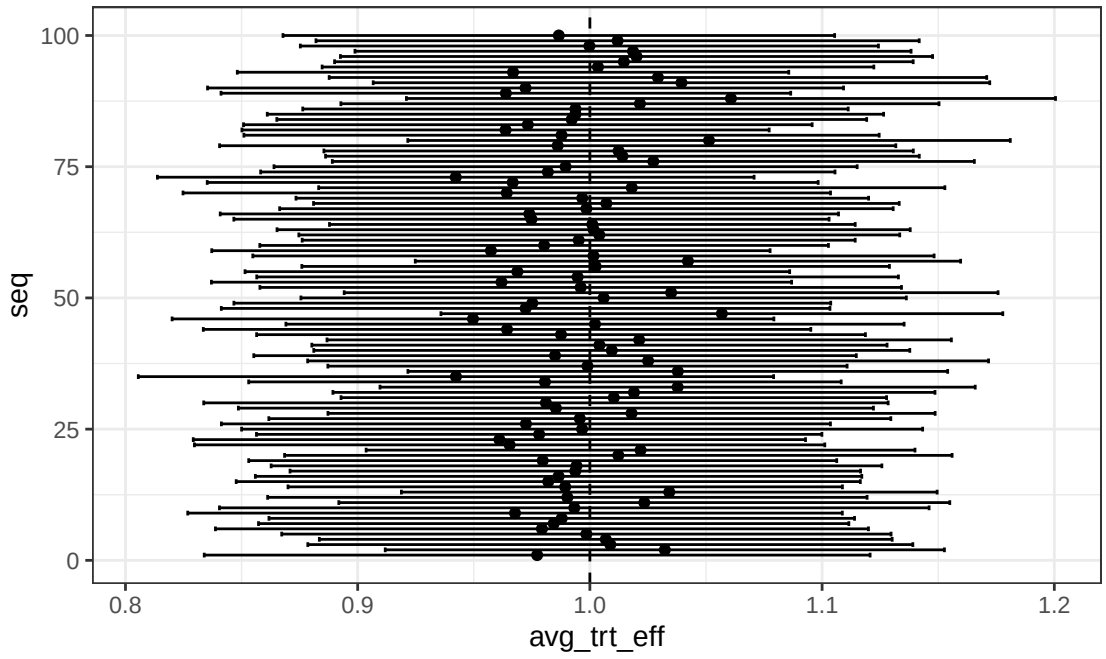
```
  plot = res %>%
    ggplot(aes(x = avg_trt_eff, y = seq)) +
    geom_point() +
    geom_errorbar(aes(xmin = ci_low, xmax = ci_high)) +
    geom_vline(xintercept = beta1, linetype = "dashed")

  return(plot)
}

cvg_rate(boot_result)
```

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

```
cvg_plot(boot_result)
```




```
save(boot_result, file = "./continuous_simulation_setting/continuous_big_med_pos.RData")
```