Continuous Simulation

Tucker Morgan - tlm2152

2/16/2022

Generating 100 Samples from Our Population

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

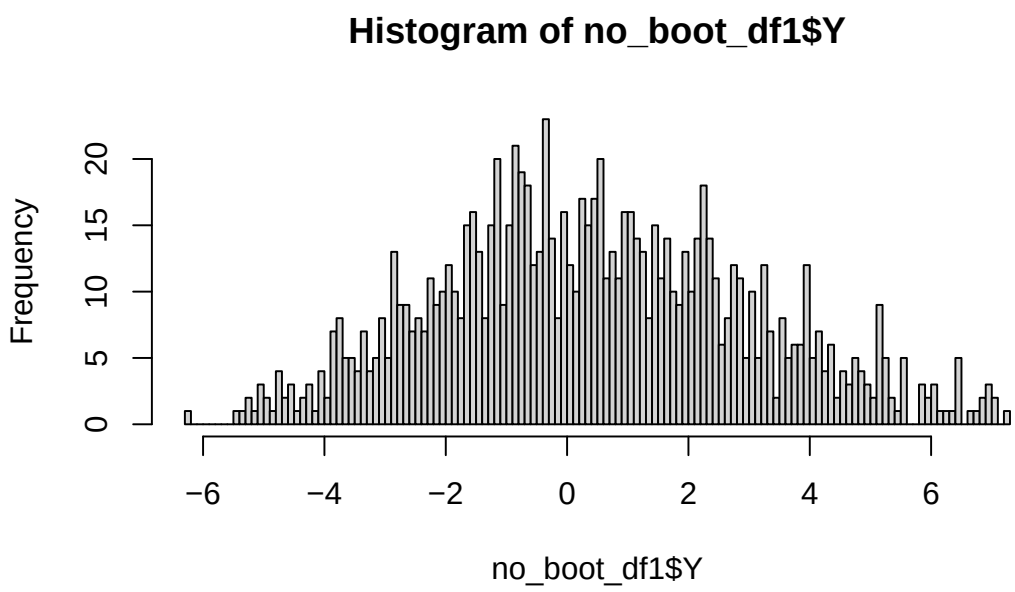
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## 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.

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

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

```
## [1] 0.326
```

```
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.   :  1.0   Min.   :-2.781322   Min.   :-1.1380   Min.   :-3.192380
##  1st Qu.:237.8   1st Qu.:-0.623489   1st Qu.: 0.0587   1st Qu.:-0.733689
##  Median :491.5   Median :-0.002109   Median : 0.5041   Median :-0.000879
##  Mean   :492.6   Mean   :-0.001345   Mean   : 0.5808   Mean   : 0.019118
```

```
##  3rd Qu.:743.5   3rd Qu.: 0.672748   3rd Qu.: 1.0346   3rd Qu.: 0.682791
##  Max.   :999.0   Max.   : 2.583806   Max.   : 3.0436   Max.   : 3.089810
##
##        A              Y                   ps             weights     subclass
##  Min.   :0.0   Min.   :-3.6396   Min.   :0.02825   Min.   :1   1      :  2
##  1st Qu.:0.0   1st Qu.: 0.1241   1st Qu.:0.23631   1st Qu.:1   2      :  2
##  Median :0.5   Median : 1.5501   Median :0.42910   Median :1   3      :  2
##  Mean   :0.5   Mean   : 1.6550   Mean   :0.46593   Mean   :1   4      :  2
##  3rd Qu.:1.0   3rd Qu.: 3.0258   3rd Qu.:0.68347   3rd Qu.:1   5      :  2
##  Max.   :1.0   Max.   : 7.2944   Max.   :0.99148   Max.   :1   6      :  2
##                                                                (Other):640
```

```
str(matched_df1)
```

```
## Classes 'matchdata', 'data.table' and 'data.frame':  652 obs. of  9 variables:
##  $ id      : int  1 2 3 4 7 11 12 13 14 15 ...
##  $ L1      : num  -1.403 0.604 1.966 0.346 1.715 ...
##  $ L2      : num  0.169 0.444 1.305 -0.304 0.936 ...
##  $ L3      : num  0.28 0.817 0.426 0.758 0.226 ...
##  $ A       : int  0 1 1 1 1 0 0 1 1 1 ...
##  $ Y       : num  0.982 3.024 3.356 1.549 4.159 ...
##  $ ps      : num  0.279 0.402 0.787 0.132 0.64 ...
##  $ weights : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ subclass: Factor w/ 326 levels "1","2","3","4",..: 212 35 65 103 222 233 217 7 9 15 ...
##  - 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)
}
```

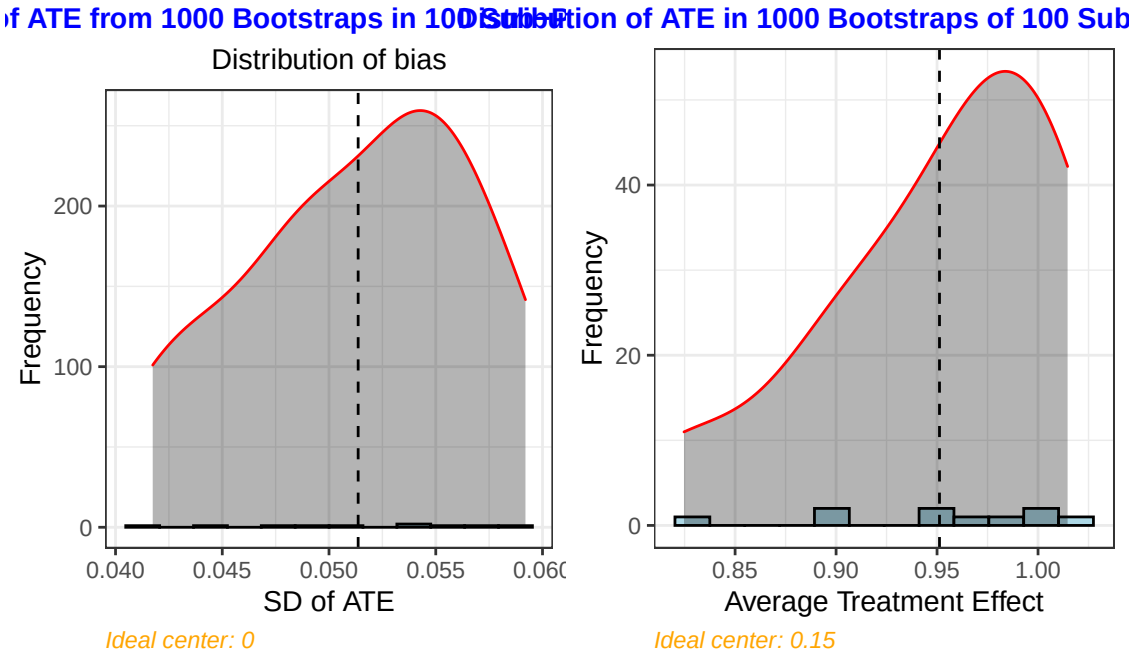
```r
# ### input matched dataframe, output however many bootstrapped samples you want ###
# first, set seed vector for reproducibility


# now, define function

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

simple_boot <- function(df, n, size = 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)
}


# 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)
```

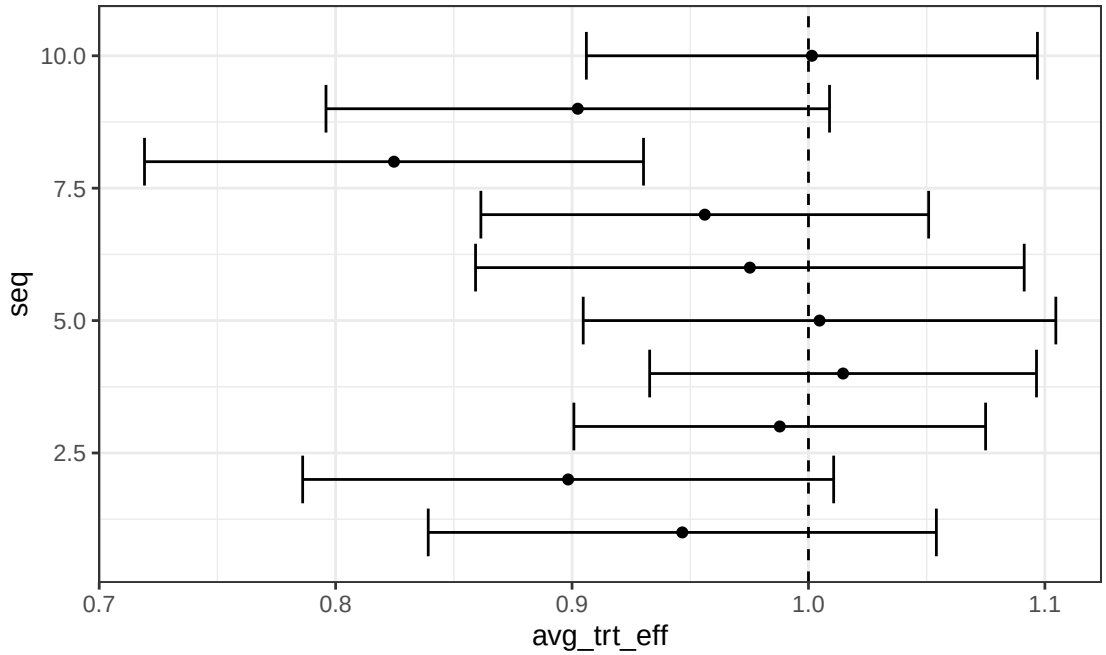


Confidence Intervals Coverage Rates

```
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] 0.9
```

```
cvg_plot(boot_result)
```




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