PS Bootstrap Binary Standardization

Hun & Amy

2/9/2022

The goal of this document is to combine a lot of the functions that Hun has made and made them into functions that can easily to called for the main loop of the simulations function.

Function to find the intercept to attain the desired treatment rate

```r
treatment_rate_coef <- function(desired_prop, seq_try, df, logit_formula ) {
  # setting up data setting variables
  seq_select = tibble(seq_try, mean_A = rep(0, length(seq_try)))

  # for loop that goes through all intercept options.
  for(i in c(1:length(seq_try))) {
    beta_0_treat = seq_try[i]

    # this will be where the beta is updated. Must have beta_0_treat in formula
    A_logit <- eval(parse(text=logit_formula ))
    p_A     <- exp(A_logit)/(1+ exp(A_logit))
    A       <- rbinom(N, 1, p_A) # this decided treat and and non treated

    # getting the mean treated
    seq_select$mean_A[i] = mean(A)
  }

  # # plotting visualizing check
  # seq_select %>%
  #   ggplot(aes(x=seq_try, y=mean_A))+
  #   geom_point()

  # selecting coeff that gives desired treated untreated proportion
  if( sum(seq_select$mean_A == desired_prop) > 0) {
    beta_0_treat = seq_select %>%  filter(mean_A == desired_prop ) %>% pull(seq_try)
  } else {
    beta_0_treat = seq_select %>%
      filter(mean_A < desired_prop + 0.01) %>%
      filter(mean_A > desired_prop - 0.01) %>%
      summarise(
        avg = mean(seq_try)
      ) %>% pull(avg)
  }

  # returning the beta value
  return(mean(beta_0_treat))
}
```

Finding the right beta_effect estimate for binary data

```
binary_outcome_rate <- function(desired_diff, seq_try, df, logit_formula ) {
  seq_select = tibble(seq_try, prop_save = rep(0, length(seq_try)))
  for(i in c(1:length(seq_try))) {
    beta_effect = seq_try[i]

    # binary outcome
    y_logit   <-eval(parse(text=logit_formula ))
    p_outcome <- exp(y_logit)/(1+ exp(y_logit))
    y_binary  <- rbinom(N, 1, p_outcome)

    # finding difference between groups
    prop = tibble(y_binary, A = df$A) %>%
      group_by(A) %>%
      summarise( mean_out = mean(y_binary)) %>%
      pivot_wider(
        names_from = A,
        values_from = mean_out
      ) %>%
      mutate (diff = `1`-`0`) %>%
      pull(diff)

    # having the difference
    seq_select$prop_save[i] = prop
  }

  # graphing
  # seq_select %>%
  #   ggplot(aes(x=seq_try, y=prop_save))+
  #   geom_point()

 # saving the best beta.
  beta_effect = seq_select %>%
    filter(prop_save < desired_diff + 0.01) %>%
    filter(prop_save > desired_diff - 0.01) %>%
    summarise(
      avg = mean(seq_try)
    ) %>% pull(avg)
 return( beta_effect)
}

# logit_formula <- "beta_effect*df$A + 0.5*df$L2 + 0.3*df$L3"
# seq_try = seq(-4,4, 0.01)
# desired_diff = 0.4
# beta_effect = binary_outcome_rate(desired_diff, seq_try, df, logit_formula)
```

Generating data

```r
data_gen <- function(N, desired_prop) {

  # Setting the Observed Variables
  L1 <- rnorm(N, 0, 1)
  L2 <- rnorm(N, 0, 1)
  L3 <- rnorm(N, 0, 1)

  # adding the covaraites to dataset
  df <- tibble(L1, L2, L3)

  desired_prop = 0.2
  seq_try      = seq(-2,2, 0.01)
  logit_formula <- "beta_0_treat+ 0.5*df$L1 + 0.4*df$L2"

  # get the desited treatment rate
  beta_0_treat = treatment_rate_coef(desired_prop, seq_try, df, logit_formula)

  # Creating the treatment assigment
  A_logit <- beta_0_treat+ 0.5*df$L1 + 0.4*df$L2
  p_A     <- exp(A_logit)/(1+ exp(A_logit))
  A       <- rbinom(N, 1, p_A)

  # adding the treated untreated data to the dataset
  df$A = A

  # continuous outcome
  y_continous <- 1*df$A +  0.5*df$L2 + 0.3*df$L3 +  rnorm(N, 0,1)

  df$y_continous = y_continous


  # binary outcome
  logit_formula <- "beta_effect*df$A + 0.5*df$L2 + 0.3*df$L3"
  seq_try = seq(-4,4, 0.01)
  desired_diff = 0.4
  beta_effect = binary_outcome_rate(desired_diff, seq_try, df, logit_formula)

  y_logit   <-eval(parse(text=logit_formula ))
  p_outcome <- exp(y_logit)/(1+ exp(y_logit))
  y_binary  <- rbinom(N, 1, p_outcome)

  df$y_binary = y_binary

  return(df)
}
N = 100
desired_prop = 0.2
data_gen(N, desired_prop)
```

```
## # A tibble: 100 x 6
##        L1     L2      L3     A y_continous y_binary
##     <dbl>  <dbl>   <dbl> <int>       <dbl>    <int>
##  1 -0.969 -1.92  -0.0297     0      -1.20         0
##  2 -1.54   0.375 -0.300      0      -0.896        0
##  3  0.362  0.240  0.699      0       1.13         1
##  4  1.25  -0.781 -1.06       0       0.282        0
##  5 -0.266 -0.517  0.103      0      -0.990        0
##  6 -0.132 -1.09  -1.87       0      -1.53         0
##  7  0.882  0.733 -1.06       0      -0.175        1
##  8 -0.154 -0.802 -1.73       0       0.454        0
##  9  0.821 -0.792  0.836      1       2.60         1
## 10  0.374 -0.698  0.594      0       0.168        1
## # ... with 90 more rows
```

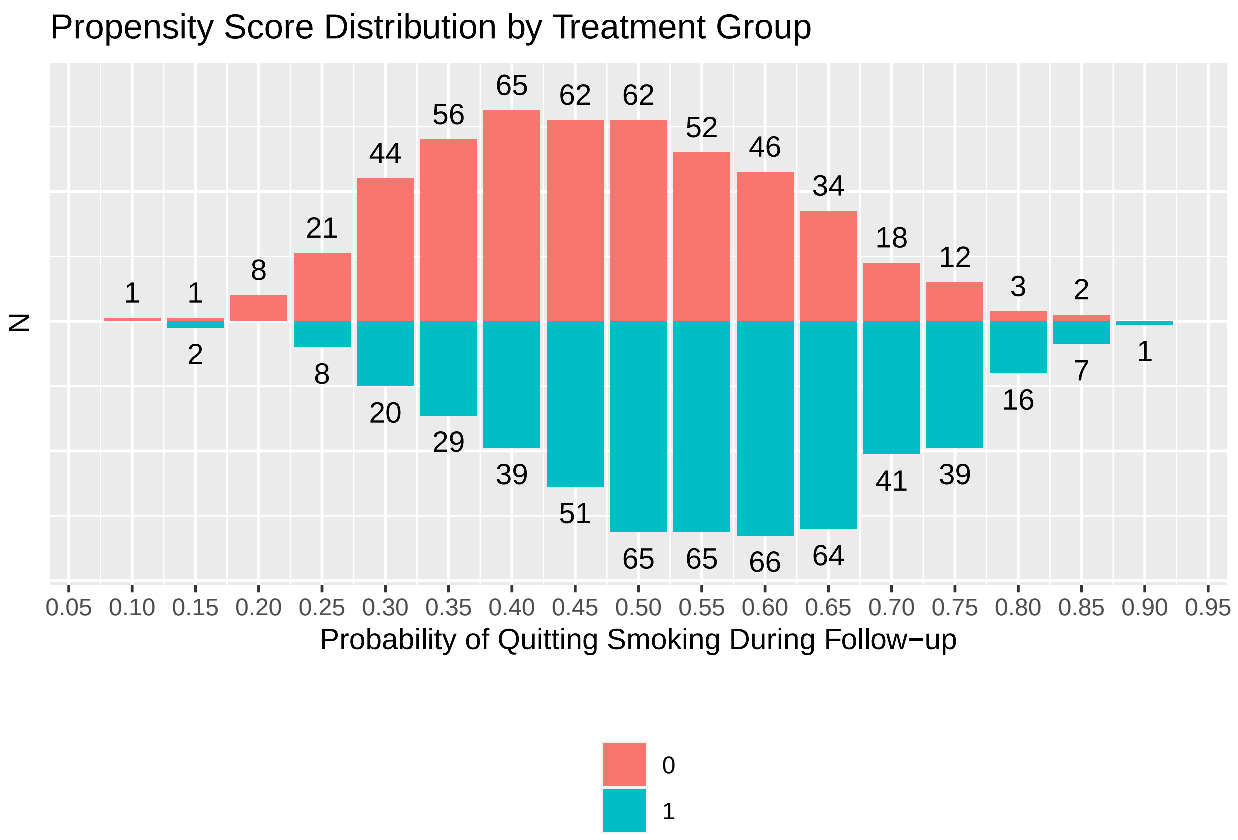
```r
pre_data <- defData(varname = "L1", formula = "0", variance = 1, dist = "normal")
pre_data <- defData(pre_data, varname = "L2", formula = "0", variance = 1,
                    dist = "normal")
pre_data <- defData(pre_data, varname = "L3", formula = "0", variance = 1,
                    dist = "normal")
pre_data <- defData(pre_data, varname = "A", formula = " 0.5*L1 + 0.27*L2 -0.17*L3",
                    dist = "binary", link = "logit")
pre_data <- defData(pre_data, varname = "Y", formula = "0.5*A + 0.8*L2 + -0.1*L3",
                    dist = "binary", link = "logit")
set.seed(7777)
df <- genData(1000, pre_data)
expit <- function(beta) {
return(exp(beta)/(1 + exp(beta)))
}
ATE <- expit(sum(0.5 + 0.8*df$L2 - 0.1*df$L3)) - expit(sum(0.8*df$L2 - 0.1*df$L3))
# this is not true ATE
# True log odds ratio: 0.5
```

Propensity Score Model

500 pairs Propensity Score distribution

```r
df %>%
  mutate(ps.grp = round(ps/0.05) * 0.05) %>%
  group_by(A, ps.grp) %>%
  summarize(n = n()) %>%
  ungroup() %>%
  mutate(n2 = ifelse(A == 0, yes = n, no =  -1*n)) %>%
  ggplot(aes(x = ps.grp, y = n2, fill = as.factor(A))) +
  geom_bar(stat = 'identity', position = 'identity') +
  geom_text(aes(label = n, x = ps.grp, y = n2 + ifelse(A == 0, 8, -8))) +
  xlab('Probability of Quitting Smoking During Follow-up') +
  ylab('N') +
  ggtitle('Propensity Score Distribution by Treatment Group') +
  scale_fill_discrete('') +
  scale_x_continuous(breaks = seq(0, 1, 0.05)) +
  theme(legend.position = 'bottom', legend.direction = 'vertical',
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank())
```

```
## `summarise()` regrouping output by 'A' (override with `.groups` argument)
```



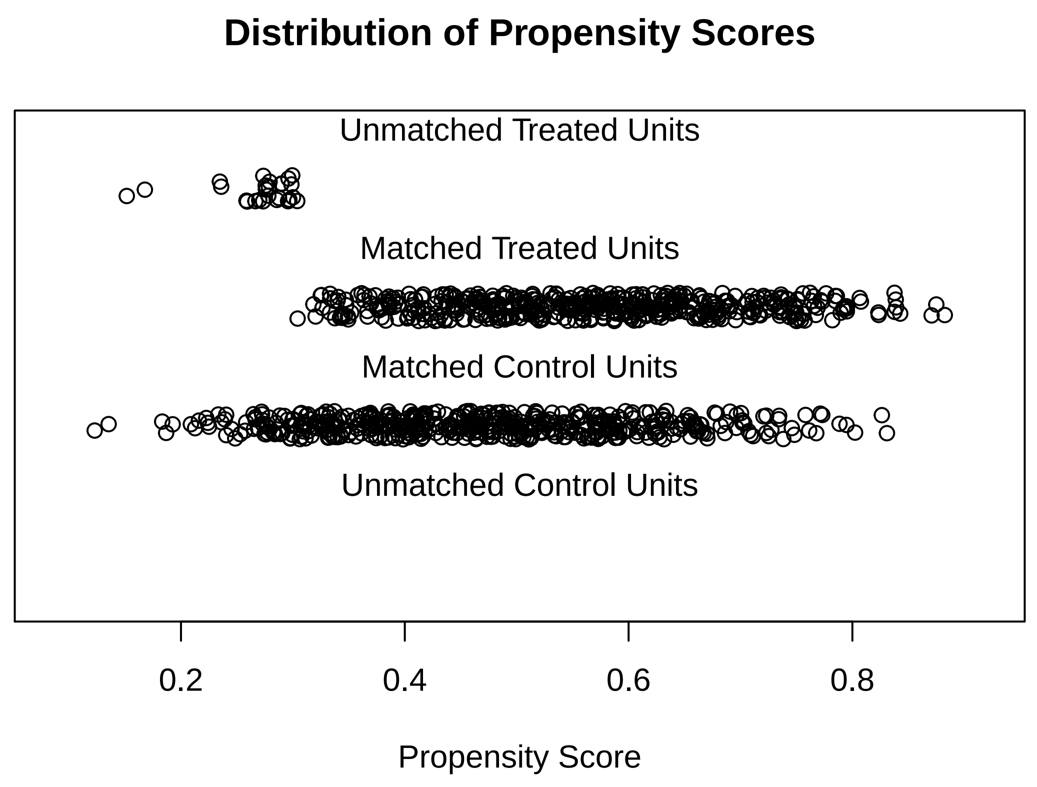
Nearest neighbor propensity score matching

```
matched <- matchit(A ~ L1 + L2 + L3, data = df,
                   distance = "glm", link = "logit",
                   method = "nearest", ratio = 1)
```

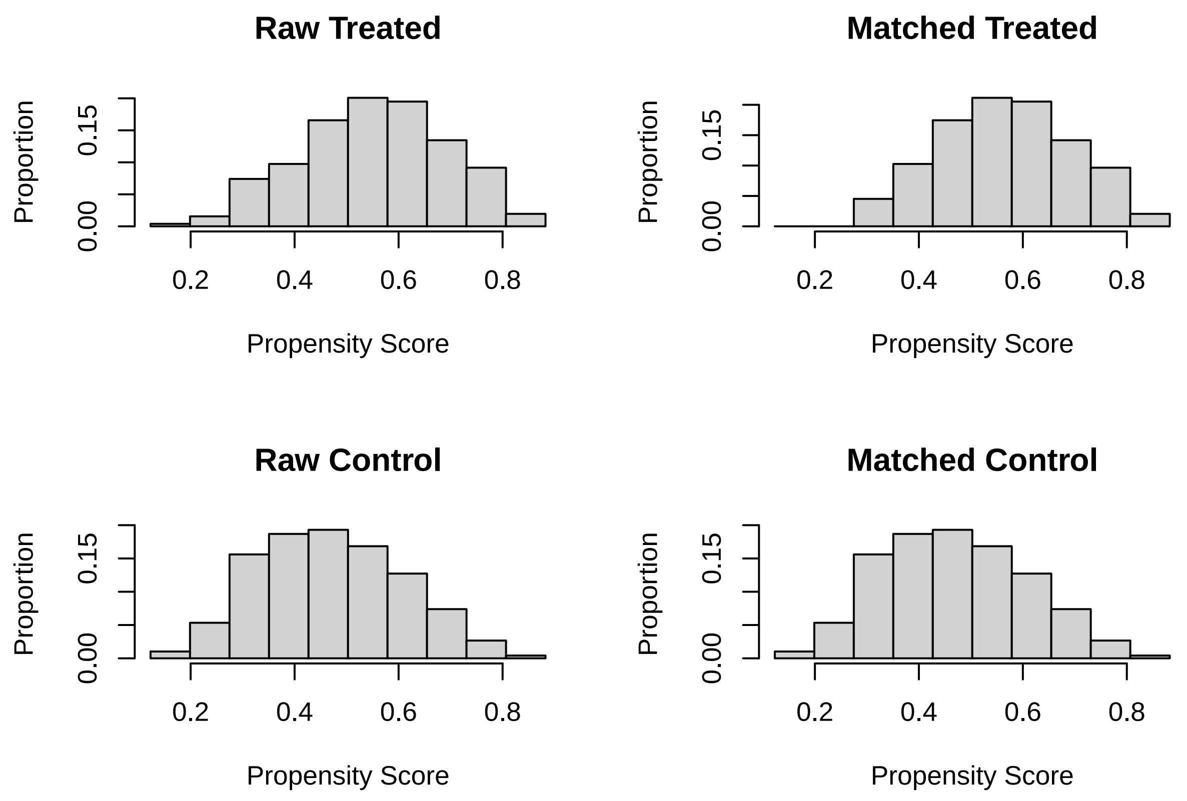
```
summary(matched)[2]
```

```
## $nn
##                 Control Treated
## All (ESS)           487     513
## All                 487     513
## Matched (ESS)       487     487
## Matched             487     487
## Unmatched             0      26
## Discarded             0       0
```

```
plot(matched, type = "jitter", interactive = FALSE)
```



```r
plot(matched, type = "histogram")
```

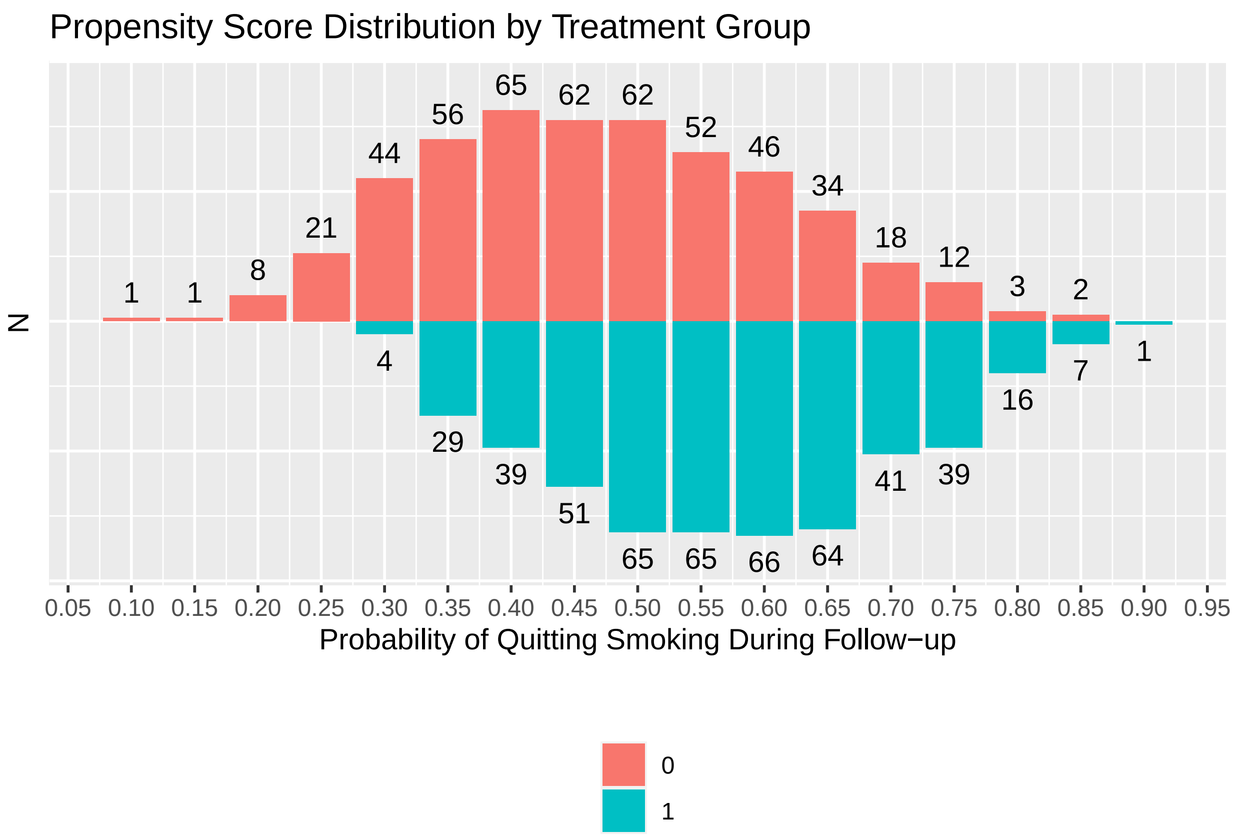


```r
matched_df <-
  match.data(matched)
```

495 pairs propensity score distribution

```
matched_df %>%
  mutate(ps.grp = round(ps/0.05) * 0.05) %>%
  group_by(A, ps.grp) %>%
  summarize(n = n()) %>%
  ungroup() %>%
  mutate(n2 = ifelse(A == 0, yes = n, no =  -1*n)) %>%
  ggplot(aes(x = ps.grp, y = n2, fill = as.factor(A))) +
  geom_bar(stat = 'identity', position = 'identity') +
  geom_text(aes(label = n, x = ps.grp, y = n2 + ifelse(A == 0, 8, -8))) +
  xlab('Probability of Quitting Smoking During Follow-up') +
  ylab('N') +
  ggtitle('Propensity Score Distribution by Treatment Group') +
  scale_fill_discrete('') +
  scale_x_continuous(breaks = seq(0, 1, 0.05)) +
  theme(legend.position = 'bottom', legend.direction = 'vertical',
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank())
```

```
## `summarise()` regrouping output by 'A' (override with `.groups` argument)
```



simple bootstrap

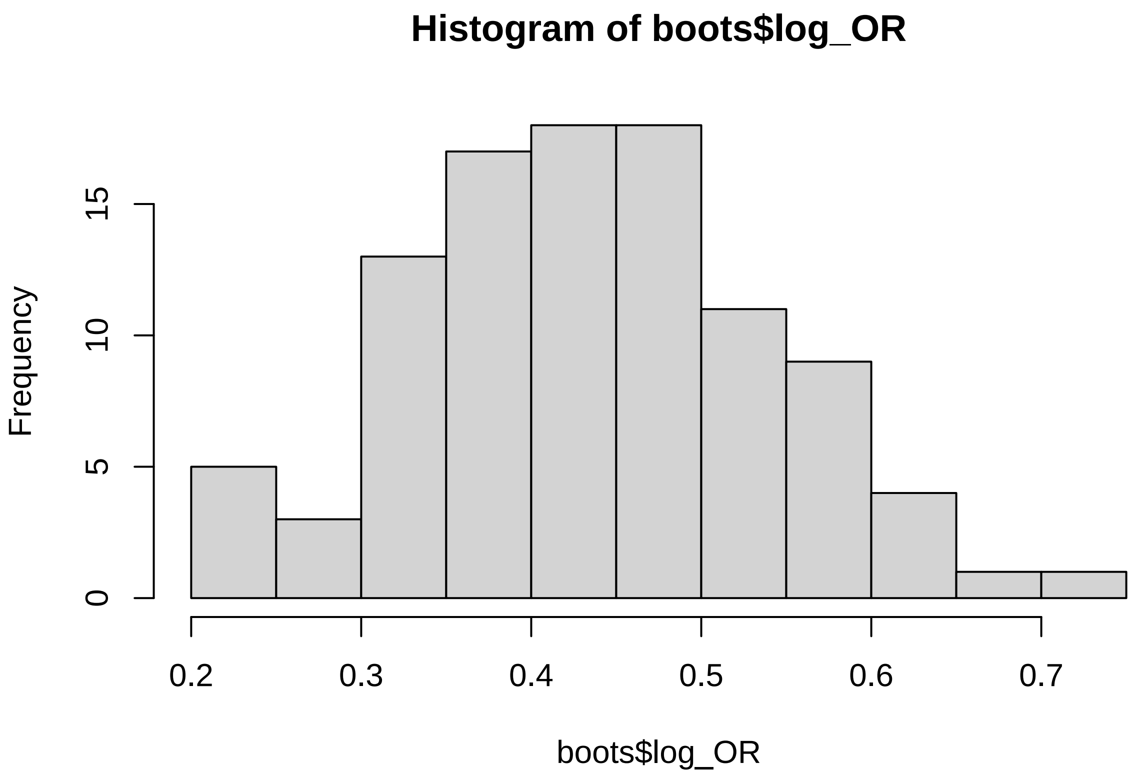
```
nboot <- 100
# set up a matrix to store results
```

```r
boots <- data.frame(i = 1:nboot,
                    se_ATE = NA,
                    se_OR = NA,
                    log_OR = NA,
                    mean1 = NA,
                    mean0 = NA,
                    difference = NA
                    )
# loop to perform the bootstrapping

for (i in 1:nboot) {
  # sample with replacement
  sampl <- matched_df %>% filter(subclass %in% sample(levels(subclass),500, replace =  TRUE))

  bootmod <- glm(Y ~ A + ps, data = sampl,
                 weights = weights, family = binomial)

  # create new data sets
  sampl.treated <- sampl %>%
    mutate(A = 1)

  sampl.untreated <- sampl %>%
    mutate(A = 0)

  # predict values
  sampl.treated$pred.y <-
    predict(bootmod, sampl.treated, type = "response")

  sampl.untreated$pred.y <-
    predict(bootmod, sampl.untreated, type = "response")


   # output results

  boots[i, "log_OR"] <- summary(bootmod)$coeff[2,1]

  boots[i, "se_OR"] <- summary(bootmod)$coeff[2,2]

  boots[i, "se_ATE"] <-
    sqrt((summary(bootmod)$coeff[2,2]*mean(sampl.treated$pred.y) *
       (1 - mean(sampl.treated$pred.y)))^2 +
    (summary(bootmod)$coeff[2,2]*mean(sampl.untreated$pred.y) *
       (1 - mean(sampl.untreated$pred.y)))^2)

  boots[i, "mean1"] <- mean(sampl.treated$pred.y)
  boots[i, "mean0"] <- mean(sampl.untreated$pred.y)
  boots[i, "difference"] <- boots[i, "mean1"] - boots[i, "mean0"]

  mean_log_OR <- mean(boots$log_OR)

  Empirical_se_ATE <- sd(boots$difference)

  mean_se_ATE <- mean(boots$se_ATE)
  Empirical_se_log_OR <- sd(boots$log_OR)

  mean_se_log_OR <- mean(boots$se_OR)

  ATE <- mean(boots$difference)

  # once loop is done, print the results
  if (i == nboot) {
    cat("ATE:")
    cat(ATE)
    cat("\n")
    cat("\n")
    cat("Empirical_se_ATE:")
    cat(Empirical_se_ATE)
    cat("\n")
    cat("\n")
    cat("mean_se_ATE:")
    cat(mean_se_ATE)
    cat("\n")
    cat("\n")
    cat("95% CI for ATE:")
    cat(ATE - 1.96*Empirical_se_ATE,
        ",",
        ATE + 1.96*Empirical_se_ATE)
    cat("\n")
    cat("\n")
    cat("mean_log_OR:")
    cat(mean_log_OR)
    cat("\n")
    cat("\n")
    cat("Empirical_se_log_OR:")
    cat(Empirical_se_log_OR)
    cat("\n")
    cat("\n")
    cat("mean_se_log_OR:")
    cat(mean_se_log_OR)
    cat("\n")
    cat("\n")
    cat("95% CI for log odds ratio:")
    cat(mean_log_OR - 1.96*mean_se_log_OR,
        ",",
        mean_log_OR + 1.96*mean_se_log_OR)
  }
}
```

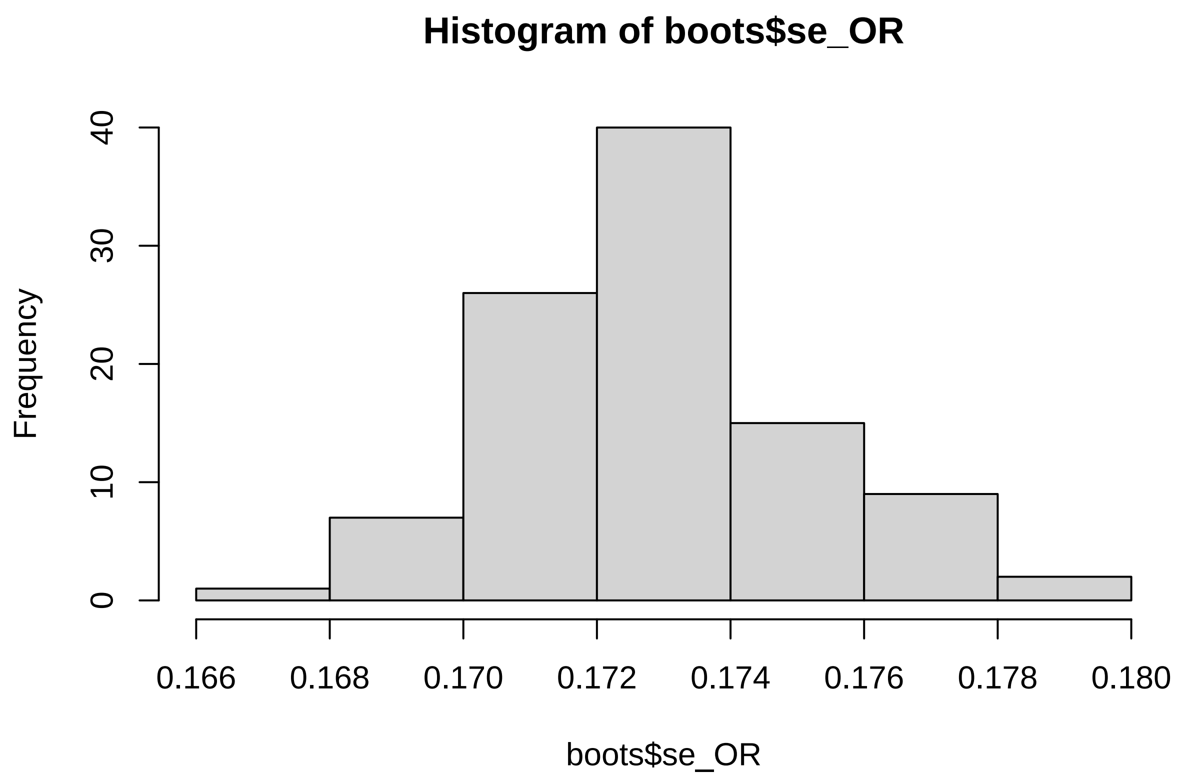
```
## ATE:0.1068614
##
## Empirical_se_ATE:0.025629
##
## mean_se_ATE:0.0601082
##
## 95% CI for ATE:0.05662853 , 0.1570942
##
```

```
## mean_log_OR:0.4353995
##
## Empirical_se_log_OR:0.1046255
##
## mean_se_log_OR:0.172854
##
## 95% CI for log odds ratio:0.09660568 , 0.7741934
```

```
hist(boots$log_OR)
```



```
hist(boots$se_OR)
```



```r
a <- glm(Y ~ A + ps, data = sampl,
                 weights = weights, family = binomial)


summary(a)
```

```
##
## Call:
## glm(formula = Y ~ A + ps, family = binomial, data = sampl, weights = weights)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5387  -1.1270   0.8968   1.0713   1.4361
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.8868     0.3264  -2.717  0.00660 **
## A             0.5997     0.1728   3.470  0.00052 ***
## ps            1.3180     0.6373   2.068  0.03863 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 868.96  on 627  degrees of freedom
## Residual deviance: 844.52  on 625  degrees of freedom
## AIC: 850.52
##
## Number of Fisher Scoring iterations: 4
```