Sim_setting_binary

Hun

2/11/2022

For dichotomous outcomes, we use a logit link i.e., $log\frac{E[Y|X]}{1-E[Y|X]} = \sum_{i=0}^{p} \theta_i X_i$ and the true average treatment effect is $E[Y^1 = 1] - E[Y^0 = 1]$.

In this project, we want to simulate the binary outcome so that the treatment (i.e. smoking status) causes an increases in the probability of the occurrence of the outcome (i.e. lung cancer) by 15 percent. For our population without treatment (smoking) history, we assume that the probability of getting lung cancer is 0.2. Namely, $E[Y^0 = 1] = 0.2$. From this fact, we know that our true log odds ratio (β_1) for our binary logistic regression is approximately 0.767.

We set our desired proportion of treatment is 0.14 given that 14% of U.S. adults were current smokers in 2019.

Based on this simulation design, we can obtain our α_0 coefficient for our propensity score model and β_0 and β_1 for our logistic model with some algebra computation.

With logit model and standardization method (creating pseudo population),

$E[Y^1 = 1] - E[Y^0 = 1]$

$\Leftrightarrow \; E[Y^1 = 1|A = 1] - E[Y^0 = 1|A = 0]$ (by exchangeabiltiy)

$\Leftrightarrow \; E[Y = 1|A = 1] - E[Y = 1|A = 0]$

$\Leftrightarrow \; \frac{e^{\beta_0+\beta_1 A+\beta_2 L_1+\beta_3 L_2}}{1+e^{\beta_0+\beta_1 A+\beta_2 L_1+\beta_3 L_2}} - 0.2 = 0.15$

$\Leftrightarrow \; e^{\beta_0+\beta_1 A+\beta_2 L_1+\beta_3 L_2} = 0.35 + 0.35e^{\beta_0+\beta_1 A+\beta_2 L_1+\beta_3 L_2}$

$\Leftrightarrow \; \beta_0 + \beta_1 A + \beta_2 L_1 + \beta_3 L_2 = log\frac{0.35}{0.65}$

Now we use,

$E[Y^0 = 1] = e^{\beta_0+\beta_2 L_1+\beta_3 L_2} = 0.2$

$\Leftrightarrow \; \beta_0 + \beta_2 L_1 + \beta_3 L_2 = log\frac{0.2}{0.8}$

$\Leftrightarrow \; \beta_2 L_1 + \beta_3 L_2 = log\frac{0.2}{0.8} + \beta_0$

Using this equation, we get the following:

$\Rightarrow \; \beta_0 + \beta_1 A + log\frac{0.2}{0.8} + \beta_0 = log\frac{0.35}{0.65}$

$\Leftrightarrow \; \beta_0 \approx (-0.619 + 1.386 + 0.767A)/2$

$\Rightarrow \; E[\hat{\beta_0}] \approx E[-0.619 + 1.386 + 0.767A]/2$

$\Rightarrow \; \hat{\beta_0} \approx (-0.619 + 1.386 + 0.767E[A])/2$

$\Rightarrow \; \hat{\beta_0} \approx (-0.619 + 1.386 + 0.767 * 0.14)/2$

$\Rightarrow \; \hat{\beta_0} \approx 0.437$

From this computation, we can see that no matter what other covariates are in the model, we can obtain $\hat{\beta_0}$.

Finding α_0 and coefficients for L1, L2 ,L3 in propensity score model

This process allows us to find the values of coefficients for our propensity score model in order to achieve the expected proportion of the treated in our data simulation.

```
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")

df <- genData(5000, pre_data)


data_gen <- function(desired_prop) {

   alpha_0 = log(desired_prop/(1 - desired_prop))

   coef_L1 <- sample(df$L1, 1000, replace = TRUE)

   coef_L2 <- sample(df$L2, 1000, replace = TRUE)

   coef_L3 <- sample(df$L3, 1000, 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]*df$L1 + coef_L2[1]*df$L2 - coef_L3[1]*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]*df$L1 + coef_L2[i]*df$L2 - coef_L3[i]*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))
}


covariate_df <- data_gen(0.14)
covariate_df
```

```
## # A tibble: 1 x 5
##   alpha_0 mean_treated_proporti~ desired_coef_L1 desired_coef_L2 desired_coef_L3
##     <dbl>                  <dbl>           <dbl>           <dbl>           <dbl>
## 1   -1.82                  0.141         -0.0225          0.0525          -0.120
```

Generating the treatment and outcome data

Here, our intention is to introduce confouders in the data generation process in order to see how using propensity matching and standardization can tackle the problems caused by confounders in predicting the average treatment effect.

```
L1_coef <- covariate_df$desired_coef_L1
L2_coef <- covariate_df$desired_coef_L2
L3_coef <- covariate_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 <- genData(5000, pre_data)

mean(df$A) #proportion of getting treated (true proportion: 0.14)
```

```
## [1] 0.1496
```

Nearest neighbor propensity score matching

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

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

```
## $nn
##                Control Treated
## All (ESS)         4252     748
## All               4252     748
## Matched (ESS)      748     748
## Matched            748     748
## Unmatched         3504       0
## Discarded            0       0
```

```r
matched_df <-
  match.data(matched, distance = "ps")
```

Generating large data close to true model and get its variance covariance matrix.

```r
True <- defData(varname = "A",
                    formula = 0.14,
                dist = "binary")

True <- defData(True, varname = "Y",
                    formula = "0.437 + 0.767*A" ,
                dist = "binary", link = "logit")

set.seed(7)
close_true_df <- genData(500000, True)

model <- glm(Y~A, data = close_true_df, family = "binomial")
model
```

```
##
## Call:  glm(formula = Y ~ A, family = "binomial", data = close_true_df)
##
## Coefficients:
## (Intercept)            A
##      0.4351       0.7640
##
## Degrees of Freedom: 499999 Total (i.e. Null);  499998 Residual
## Null Deviance:       659100
## Residual Deviance: 652000    AIC: 652000
```

```r
sqrt(vcov(model)[2,2]) ## standard error of Beta_1
```

```
## [1] 0.009489783
```

PS matched model varianec-covariance matrix

```r
matched_ps_model <- glm(Y ~ A + ps, data = matched_df, family = "binomial")
```

```r
sqrt(vcov(matched_ps_model)[2,2]) ## standard of Beta_1
```

```
## [1] 0.1221168
```

simple bootstrap

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

se_log_OR <- sqrt(sum((boots$log_OR - mean(boots$log_OR))^2) / (nrow(boots - 1)))

Empirical_se_ATE <- sd(boots$difference)

mean_se_ATE <- mean(boots$se_ATE)

Empirical_se_log_OR <- sd(boots$log_OR) # = se_log_OR (manually calculated)

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*Empirical_se_log_OR,
        ",",
        mean_log_OR + 1.96*Empirical_se_log_OR)
  }
}
```

```
## ATE:0.1205792
##
## Empirical_se_ATE:0.02286006
##
## mean_se_ATE:0.0542074
##
## 95% CI for ATE:0.07577353 , 0.165385
##
## mean_log_OR:0.654242
##
## Empirical_se_log_OR:0.1221457
##
## mean_se_log_OR:0.1749445
##
## 95% CI for log odds ratio:0.4148363 , 0.8936476
```

True ATE: 0.15

True log OR: 0.767