HUDM6026 Homework_04

Seng Lei and Chenguang Pan

Feb 17, 2023

PART 01: Homework 03's Solution

Step 1, input the given variance-covariance matrix; Note, since these ten variables follow the standard normal distribution, the values in correlation matrix is same to the value in var-cov martix.

```
> library(mvtnorm)
> cov = matrix(c(1,0,0,0,0.2,0,0,0,0,0,
+                0,1,0,0,0,0.9,0,0,0,0,
+                0,0,1,0,0,0,0,0.2,0,0,
+                0,0,0,1,0,0,0,0,0.9,0,
+                0.2,0,0,0,1,0,0,0,0,0,
+                0,0.9,0,0,0,1,0,0,0,0,
+                0,0,0,0,0,0,1,0,0,0,
+                0,0,0.2,0,0,0,0,1,0,0,
+                0,0,0,0.9,0,0,0,0,1,0,
+                0,0,0,0,0,0,0,0,0,1),10,10)
> # randomly generate one row of data to test this matrix
> # X <- rmvnorm(1,sigma = cov) # looking good
```

Step 2, write a function to generate 500 observations

```
> dat_gen <- function(n, cov){
+   return(rmvnorm(n, mean=rep(0,10), sigma=cov))}
> s1 <- dat_gen(500, cov)
> head(s1) # looking good
           [,1]        [,2]       [,3]       [,4]       [,5]       [,6]
[1,] -1.2033983  2.16405854  1.8925290 -0.1347842 -0.8800321  1.8201396
[2,]  0.5122555  0.06995376  0.5754062 -0.2193337 -1.6305364 -0.0802548
[3,]  0.1796472 -0.01626371 -0.2864921 -0.1460917  1.2931921 -0.3975117
[4,] -0.8878763  1.28878777  0.4147153  0.6480119 -0.6308978  1.3782255
[5,] -0.6189909  1.18963879  1.3302009 -1.0194905  0.5789616  0.9689868
[6,] -0.6755019  0.74083768  0.5248268 -1.0265009  0.7863574 -0.1428945
           [,7]       [,8]        [,9]       [,10]
[1,] -0.6463804 -0.2958384 -0.26469535  0.11363994
[2,] -0.1766832  1.6925166 -0.35073475 -0.05368432
[3,] -1.0065125  0.5366323 -0.09821094  0.15239918
[4,]  0.4702731 -0.5510015  0.40660522 -1.10826913
[5,] -0.5531109  2.8420892 -0.78926520  0.56090351
[6,]  0.3264251 -0.8211002 -0.91033567  1.95166596
```

Step 3, dicotormize the 1st, 3rd, 5th, 6th, 8th, and 9th variable within a for-loop.

```
> for (i in c(1,3,5,6,8,9)) {
+   s1[,i] <- ifelse(s1[,i] > mean(s1[,i]), 1, 0)
+ }
> head(s1)
     [,1]        [,2] [,3]       [,4] [,5] [,6]       [,7] [,8] [,9]
[1,]    0  2.16405854    1 -0.1347842    0    1 -0.6463804    0    0
[2,]    1  0.06995376    1 -0.2193337    0    0 -0.1766832    1    0
[3,]    1 -0.01626371    0 -0.1460917    1    0 -1.0065125    1    0
[4,]    0  1.28878777    1  0.6480119    0    1  0.4702731    0    1
[5,]    0  1.18963879    1 -1.0194905    1    1 -0.5531109    1    0
[6,]    0  0.74083768    1 -1.0265009    1    0  0.3264251    0    0
           [,10]
[1,]  0.11363994
[2,] -0.05368432
[3,]  0.15239918
[4,] -1.10826913
[5,]  0.56090351
[6,]  1.95166596
```

PART 02: Homework 03 Continued

2.1 Choose the Propensity Score Models

For this homework, I choose the scenario B:

$$Pr[A = 1|W_i] = (1 + e^{-(\beta_0+\beta_1 W_1+\beta_2 W_2+\beta_3 W_3+\beta_4 W_4+\beta_5 W_5+\beta_6 W_6+\beta_7 W_7+\beta_2 W_2 W_2)})^{-1}$$

.

2.2 Using `BinNor` to Generate Observations

In the initial stage for data generating, one problem is if we first generate data based on the correlation matrix and then dichotomize some of the data according to the means, the final correlation will be attenuated comparing to the original one. Based on the Prof.Keller's in-class notes, here I choose the function `BinNor`.

```
> library(BinNor)
> # import the cor matrix
> cmat <- matrix(c(1,0,0,0,0.2,0,0,0,0,0,
+                  0,1,0,0,0,0.9,0,0,0,0,
+                  0,0,1,0,0,0,0,0.2,0,0,
+                  0,0,0,1,0,0,0,0,0.9,0,
+                  0.2,0,0,0,1,0,0,0,0,0,
+                  0,0.9,0,0,0,1,0,0,0,0,
+                  0,0,0,0,0,0,1,0,0,0,
+                  0,0,0.2,0,0,0,0,1,0,0,
+                  0,0,0,0.9,0,0,0,0,1,0,
+                  0,0,0,0,0,0,0,0,0,1),10,10)
>
> # always report warning: All correlations must be in feasible range!
> sigma.star <- compute.sigma.star(no.bin = 6,
+                                  no.nor = 4,
+                                  prop.vec.bin = c(0.5,0.5,0.5,0.5,0.5,0.5),
```

```
+                                    corr.mat = cmat)
>
> mydata <- jointly.generate.binary.normal(no.rows = 500,
+                                          no.bin = 6,
+                                          no.nor = 4,
+                                          prop.vec.bin = c(0.5,0.5,0.5,0.5,0.5,0.5),
+                                          mean.vec.nor = c(0,0,0,0),
+                                          var.nor = c(1,1,1,1),
+                                          sigma.star = sigma.star$sigma_star,
+                                          continue.with.warning = T)
```

By setting the chunk's parameter `eval = FALSE`, the code chunk above won't run. But after several tries, I can't figure out how generate the required data through this `BinNor`. I temporarily skipped this and use the attenuated data to continue this script.

2.3 Rewrite the data generation function

Here, I rewrite the data generation function to ensure each running time will produce a matrix with multivariate data and dichotomous data. But, one should notice, this dataset's correlation matrix is attenuated!!

rewrite the data generation function for several purposes:
- generate data multivariate normal distribution and dichotomous data
- compute the propensity scores - get dichotomous exposure variable A
- get the outcome Y

```
> dat_gen <- function(n, cov){ # n is the number of observations
+   # generate data from multinormal distribution
+   data_01 <- rmvnorm(n, mean=rep(0,10), sigma=cov)
+   # dichotomize the columns
+   for (i in c(1,3,5,6,8,9)) {data_01[,i] <- ifelse(data_01[,i] > mean(data_01[,i]), 1, 0)}
+
+   # compute propensity score
+   linear_part <- (0 + 0.8 * data_01[,1] - 0.25*data_01[,2]+ 0.6*data_01[,3]
+                   -0.4*data_01[,4] - 0.8*data_01[,5] - 0.5*data_01[,6]
+                   +0.7*data_01[,7] - 0.25*data_01[,2]*data_01[,2])/(-1)
+   ps <- sapply(linear_part, function(x){(1 + exp(x))^(-1)})
+   # add the propensity score vector to the matrix as 11th columns
+   data_01 <- cbind(data_01, ps)
+
+   # compute the dichotomous exposure
+   A_expo <- runif(n)
+   A_expo <- ifelse(A_expo < ps, 1, 0)
+   # add the exposure to the matrix as 12th columns
+   data_01 <- cbind(data_01, A_expo)
+
+   # calculate the outcome Y
+   Y_out <- runif(n)
+   outcome_ <- (1+ exp(-(-3.85+0.3*data_01[,1]-0.36*data_01[,2]-0.73*data_01[,3]
+                         -0.2*data_01[,4]+0.71*data_01[,8]-0.19*data_01[,9]
+                         +0.26*data_01[,10]-0.4*data_01[,12])))^(-1)
+   Y_out <- ifelse(Y_out < outcome_,1,0)
+   # add the exposure to the matrix as 12th columns
+   data_01 <- cbind(data_01, Y_out)
```

```
+
+   # also create a column to load the Lee's version of outcome, which is continuous
+   Y_out_lee <- 3.85 + 0.3*data_01[,1]-0.36*data_01[,2]-0.73*data_01[,3]
+                        -0.2*data_01[,4]+0.71*data_01[,8]-0.19*data_01[,9]
+                        +0.26*data_01[,10]-0.4*data_01[,12]
+   # note: the error term is intentionally ignored here
+   data_01 <- cbind(data_01, Y_out_lee)
+   return(data_01)
+ }
>
> # randomly generate a dataset to see the structure
> data_test <- dat_gen(500, cov)
> round(head(data_test),2)
                                          ps A_expo Y_out Y_out_lee
[1,] 0  0.01 0 -0.06 1 1  0.12 1 0 -0.73 0.23      1     1      3.85
[2,] 0 -0.68 1 -0.59 0 0  0.15 1 0 -0.34 0.73      1     0      3.36
[3,] 0 -0.52 0 -0.25 0 0 -1.23 0 1  0.33 0.33      0     0      4.04
[4,] 0  0.21 1 -1.40 0 1 -0.42 0 0  2.20 0.57      0     0      3.04
[5,] 1 -0.91 1  0.55 1 1 -0.07 0 1 -0.05 0.46      1     0      3.75
[6,] 0 -1.15 1 -0.07 0 0  0.03 1 0 -0.35 0.65      1     0      3.53
```

The dataset looks good.

2.4 Replicate the function

Next, I replicate the function for 1000 times.

```
> R <- 1000
> obs <- 500
> # data_list is a list, each entry is a data matrix with 500 obs
> data_list <- replicate(n = R,
+                        expr = dat_gen(obs,cov = cov),
+                        simplify = FALSE)
```

2.5 Write a Function to estimate the ATE

The first function is the naive function.

```
> get_ate <- function(data_mat){
+   # transform the matrix into dataframe
+   df_ <- as.data.frame(data_mat)
+   ate <- mean(df_[df_$A_expo == 1,14])- mean(df_[df_$A_expo ==0,14])
+   return(ate)
+ }
```

This function runs well. Then put it into the dataframe list.

```
> ate_list <- sapply(data_list, get_ate)
> ate_average <- mean(ate_list)
> ate_average
[1] 0.07248186
```

Based on Lee's continuous outcomes, the 1000 simulated data show that the average treatment effect is 0.0724819.

Next, we use the regression method to get the ate.

```
> get_ate_2 <- function(data_mat){
+   # transform the matrix into dataframe
+   df_ <- as.data.frame(data_mat)
+   model <- lm(df_[,14] ~ df_[,1]+df_[,2]+df_[,3]+df_[,4]+df_[,5]+df_[,6]+
+                 df_[,7]+df_[,8]+df_[,9]+df_[,10] + df_[,12])
+   return(model$coefficients[12])
+ }
```

This function runs well. Then put it into the dataframe list.

```
> ate_list <- sapply(data_list, get_ate_2)
> ate_average <- mean(ate_list)
> ate_average
[1] -4.449186e-18
```

This unbiased estimate of average treatment effect is $-4.4491857 \times 10^{-18}$. This result is very close to 0, which means the treatment effect is almost none.

2.6 Discussion

(a) The biggest challenge is how to generate the unattenuated data through `BinNorm`. There is no enough information on the internet for reference. This homework actually base on the attenuated data.

(b) Comparing to the unbiased estimation, the biased (naive) estimation is unreliable and misleading. For the result from the unbiased estimation shows there is no treatment effect, but the alternative one say yes.