HUDM6026 Final Project

Chenguang Pan & Seng Lei

April 28, 2023

1.0 Introduction

High School Longitudinal Study of 2009(HSLS:09) is a nationally representative, longitudinal study of 23,000+ ninth graders from 944 schools in 2009. It provides comprehensive information about student's background, academic performance in both high school and college, personal attitudes towards study and school, etc. Therefore, this current project uses the HSLS:09 open dataset.

This study analyzed a large dataset consisting of 23,503 observations and 9,614 variables to investigate the potential relationship between ninth-grade mathematics foundation and future achievement in STEM fields. To accomplish this objective, a simple linear regression model was employed, which allowed for the estimation of the effect of math proficiency on overall GPA in STEM courses throughout high school. The standardized mathematics assessment of algebraic reasoning, administered during the first semester of grade 9, was utilized to measure students' mathematical abilities at the onset of high school. In turn, the overall GPA in STEM courses was used as a metric of academic performance in STEM subjects throughout the high school years.

After some data cleaning, we kept 199,948 observations for analysis. The simple linear model is:

$$y_i = \beta_0 + \beta_1 \times x_i + e_i,$$

where y_i is the estimated individual outcome for overall STEM GPA, x_i is the student's mathematics assessment score, and e_i is the measurement error.

2.0 Population data descriptions

As a simulated study, we treated cleaned dataset as the population, N=19948. The mean and standard deviation for the dependent variable are 2.440 and .934. And 51.250 and 10.031 for the predictor. The correlation coefficient between these two variables is .567.

```
> model_lm <- lm(X3TGPASTEM ~ X1TXMTSCOR, data = hsls_sub)
> summary(model_lm)

Call:
lm(formula = X3TGPASTEM ~ X1TXMTSCOR, data = hsls_sub)

Residuals:
     Min       1Q   Median       3Q      Max
-2.82822 -0.50345  0.05364  0.55167  2.70153

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.264846   0.028358  -9.339   <2e-16 ***
```

```
X1TXMTSCOR   0.052792   0.000543  97.220   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7693 on 19946 degrees of freedom
Multiple R-squared:  0.3215,    Adjusted R-squared:  0.3215
F-statistic:  9452 on 1 and 19946 DF,  p-value: < 2.2e-16
```

The simple linear regression model presented that the overall model can explain the 32.15% of the variance in outcome, $F(1, 19946) = 9452$, $p < .001$. One score increase in 9th grader's math assessment will be associated with .053 increase in overall STEM GPA and this relation is statistically significant, $\beta_1 = .053$, $p < .001$. The expected value of overall STEM GPA (i.e., β_0) when student gets zero in math assessment is $-.265$, $p < .001$. The negative GPA does not make any sense, but we ignored this issue and move on the study.

3.0 Writing R functions

```
> dat_gen <- function(size= 500,  # smaple size
+                     betas,      # a numeric array of betas
+                     iv_mean,    # predictor's mean
+                     iv_var,     # predictor's variance
+                     error_sd){  # residuals' sd
+   # data mainly are generated from a normal distribution ~ N(iv_mean, iv_sd)
+   X <- rnorm(size, mean = iv_mean, sd= sqrt(iv_var))
+   X_aug <- cbind(1, X)
+   # residuals are generated from ~N(0, sd)
+   Error <- rnorm(size, mean=0, sd=error_sd)
+   # based on the parameters to generate the outcomes
+   Y <- X_aug %*% as.matrix(betas) + Error
+   out <- cbind(Y, X)
+   colnames(out) <- c("Y", "X1")
+   return(as.data.frame(out))
+ }
```

The data generation function takes the sample size, regression coefficients, predictors' mean and variance, and the standard deviation of residual as input. It returns a simulated dataset in `dataframe` format with the outcome in the first column and predictor in the second.

```
> reg <- function(ds) {
+   x <- as.matrix(ds[,2])
+   y <- as.matrix(ds[,1])
+   y_cen <- apply(y, 2, function(x) x-mean(x))
+   x_cen <- apply(x, 2, function(x) x-mean(x))
+   # the OLS method
+   b1 <- sum(x_cen*y_cen)/sum(x_cen^2)
+   b0 <- mean(y - x*b1)
+   y_hat <- b0 + x*b1
+   sse <- sum((y-y_hat)^2)
+   sig_sq <- sse/(nrow(x)-2)
+   # the alternative method
+   b1_a <- sum(y_cen/x_cen)/nrow(x)
+   b0_a <- mean(y - x*b1_a)
+   y_hat_a <- b0_a + x*b1_a
+   sse_a <- sum((y-y_hat_a)^2)
+   sig_sq_a <- sse_a/nrow(x)
+   out_ <- cbind(b0, b1, sig_sq, b0_a, b1_a,sig_sq_a)
+   return(out_)
+ }
```

The estimation function takes the simulated data frame as input, and returns the estimated β_0, β_1, residual's variance σ^2 from both least square and alternative methods.

4.0 Monte Carlo Simulation

The basic idea behind the Monte Carlo simulation is that one can draw a large number of random samples from a probability distribution representing the population being studied, and then these samples are used to estimate its statistical properties. This project used Monte Carlo method to draw 1000 random samples with the size of 40 by using the `dat_gen()` function above.

```
> R <- 1000
> set.seed(666)
> # randomly generate 1000 samples
> dat_list <- replicate(n = R,
+                         expr = dat_gen(size = 40,
+                                        betas = c(-0.265,0.053),
+                                        iv_mean = 51.24985, iv_var = 100.6209,
+                                        error_sd = 0.7693),
+                         simplify = FALSE)
> # estimated the simple regression model on each sample
> estimates <- sapply(X = dat_list,
+                     FUN = reg,
+                     simplify = TRUE)
> estimates <- t(estimates)
> colnames(estimates) <- c("b0", "b1", "sig_sq", "b0_a", "b1_a", "sig_sq_a")
> (estimates_hat_mean <- round(apply(estimates,2,mean),3))
      b0       b1   sig_sq     b0_a     b1_a sig_sq_a
  -0.265    0.053    0.585   -5.143    0.146  121.727
> (estimates_hat_var <- round(apply(estimates,2,var),3))
        b0          b1      sig_sq        b0_a        b1_a    sig_sq_a
     0.415       0.000       0.017    3558.920       1.317  875788.198
> (estimates_hat_se <- round(apply(estimates,2,function(x) sd(x)/sqrt(R)),3))
      b0       b1   sig_sq     b0_a     b1_a sig_sq_a
   0.020    0.000    0.004    1.887    0.036   29.594
>
> # to calculate the MSE
> # first to make a parameter matrix in shape of the estimates matrix
> theta_m <- matrix(c(-0.265, 0.053,0.7693), nrow(estimates),6 , byrow = T)
> # use the estimates matrix minus the parameter matrix
> est_cent <-estimates-theta_m
> # use apply to get the mse for each estimates
> (estimates_mse <- round(apply(est_cent,2,function(x) sum(x^2)/R),3))
        b0          b1      sig_sq        b0_a        b1_a    sig_sq_a
     0.415       0.000       0.051    3579.161       1.324  889543.113
```

Next, we wrote a function to calculate the mean squared error of estimators based on trimmed mean.