# 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 decided to work on the HSLS:09 open dataset.

Based on the HSLS09 open data, we investigate the potential relationship between ninth-grader's 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 foundation 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 residual.

## 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., $\beta_0$) 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 $\beta_0$, $\beta_1$, residual's variance $\sigma^2$ 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")
>
> # write a function to calculate the estimates
> est_out <- function(esti_mat, size){
+   # 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(esti_mat),6 , byrow = T)
+   # use the estimates matrix minus the parameter matrix
+   est_cent <-esti_mat-theta_m
+   # use apply to get the mse for each estimates
+   estimates_hat_mean <- round(apply(esti_mat,2,mean),3)
+   estimates_hat_var <- round(apply(esti_mat,2,var),3)
+   estimates_hat_se <- round(apply(esti_mat,2,function(x) sd(x)/sqrt(size)),3)
+   estimates_mse <- round(apply(est_cent,2,function(x) sum(x^2)/size),3)
+   results <- rbind(estimates_hat_mean,estimates_hat_var,
+                    estimates_hat_se,estimates_mse)
+   rownames(results) <- c("Mean", "Variance","SE", "MSE")
+   results_trans <- as.data.frame(t(results))
+   # add a column to calculate the bias
+   par_m <- matrix(c(-0.265, 0.053,0.7693), 6,1)
+   results_trans$Bias <- results_trans$Mean - par_m
```

```
+   results_trans$Parameter <- par_m
+   results_trans <- results_trans[,c(6,1,3,2,5,4)]
+   return(results_trans)
+ }
> (mc_out <- est_out(estimates, R))
         Parameter    Mean     SE   Variance     Bias        MSE
b0         -0.2650  -0.265  0.020      0.415   0.0000      0.415
b1          0.0530   0.053  0.000      0.000   0.0000      0.000
sig_sq      0.7693   0.585  0.004      0.017  -0.1843      0.051
b0_a       -0.2650  -5.143  1.887   3558.920  -4.8780   3579.161
b1_a        0.0530   0.146  0.036      1.317   0.0930      1.324
sig_sq_a    0.7693 121.727 29.594 875788.198 120.9577 889543.113
```

## 5.0 Bootstrap method

The bootstrap method is a statistical technique that involves the repeated sampling with replacement from the original data to obtain estimates of variability and uncertainty of a statistic of interest. Specifically, the statistic is computed on each of the resamples, and the resulting distribution is used to calculate confidence intervals or conduct hypothesis tests. Since we have a single sample of 40 observations in this context, the bootstrap method is proposed to be applied to the row indices of the original dataset. Each resample would be created based on the shuffled indices of the original data, thus allowing for the generation of multiple estimates of the statistic of interest.

The R function for bootstrap is as follows.

```
> # generate a single dataset
> data_b <-dat_gen(size = 40,betas = c(-0.265,0.053),
+                  iv_mean = 51.24985, iv_var = 100.6209,
+                  error_sd = 0.7693)
> # run bootstrapping on this single dataset
> B = 1000
> # shuffle the 1:40 index rather than data_b
> boot_index <- replicate(n=B,
+                         expr = sample(1:40, 40, TRUE),
+                         simplify = FALSE)
> # use the bootstrapped indices to extracted the data
> boot_samp <- list()
> for (i in 1:1000) {
+   boot_unit <- data_b[boot_index[[i]],]
+   boot_samp[[i]] <- boot_unit
+ }
> estimates <- sapply(X = boot_samp,
+                     FUN = reg,
+                     simplify = TRUE)
> estimates <- t(estimates)
> colnames(estimates) <- c("b0", "b1", "sig_sq", "b0_a", "b1_a", "sig_sq_a")
>
> (bs_out<-est_out(estimates, B))
         Parameter     Mean       SE     Variance      Bias          MSE
b0         -0.2650   -0.568    0.019 3.670000e-01   -0.3030 4.580000e-01
b1          0.0530    0.058    0.000 0.000000e+00    0.0050 0.000000e+00
sig_sq      0.7693    0.477    0.004 1.200000e-02   -0.2923 9.800000e-02
b0_a       -0.2650    2.819    7.072 5.000976e+04    3.0840 4.996926e+04
```

```
b1_a        0.0530   -0.010    0.139 1.922100e+01   -0.0630 1.920600e+01
sig_sq_a    0.7693 3220.335 2961.135 8.768318e+09 3219.5657 8.769915e+09
```

## 6.0 Jackknife method

The jackknife method is a statistical technique to estimate the bias and variance of a statistic by systematically leaving out one observation at a time from the dataset, creating a series of "jackknife samples". By recalculating the statistic of interest on each of these samples, we can obtain an approximation of the distribution of the statistic, and use it to estimate its bias and standard error. In the present context, we applied the jackknife method to the same 40-observation dataset.

```
> jack_list <- list()
> for (i in 1:40) {
+   # each round drop the ith observation
+   data_loov <- data_b[-i,]
+   # make a list to load each jacknife sample
+   jack_list[[i]] <- data_loov
+ }
>
> estimates <- sapply(X = jack_list,
+                     FUN = reg,
+                     simplify = TRUE)
> estimates <- t(estimates)
> # this is a 40 x 6 matrix
> colnames(estimates) <- c("b0", "b1", "sig_sq", "b0_a", "b1_a", "sig_sq_a")
>
> # since the population parameter is known, we use them directly
> (jn_out<-est_out(estimates, 40))
         Parameter   Mean     SE   Variance    Bias        MSE
b0         -0.2650 -0.533  0.016      0.010 -0.2680      0.082
b1          0.0530  0.057  0.000      0.000  0.0040      0.000
sig_sq      0.7693  0.510  0.003      0.000 -0.2593      0.068
b0_a       -0.2650 -4.347  6.444   1661.026 -4.0820   1636.160
b1_a        0.0530  0.133  0.129      0.667  0.0800      0.657
sig_sq_a    0.7693 96.400 87.796 308322.929 95.6307 309760.012
```

One should notice that the population information is available in this simulated study. Therefore, we chose to not use the plug-in principle to estimate the bias or MSE.

## 7.0 Comparing the results

*Table 1* combines the outcomes from all three methods.

**Table 1**
*Statistics from Monte Carlo, Bootstrap, and Jacknife methods*

| Method | Estimate | Parameter | Mean | SE | Variance | Bias | MSE |
|---|---|---|---|---|---|---|---|
| Monte Carlo | $\widehat{\beta_1}$ | 0.053 | 0.053 | 0 | 0 | 0 | 0 |
| | $\hat{\sigma}^2$ | 0.7693 | 0.585 | 0.004 | 0.017 | -0.1843 | 0.051 |
| | $\beta_1^a$ | 0.053 | 0.146 | 0.036 | 1.317 | 0.093 | 1.324 |
| | $\sigma_a^2$ | 0.7693 | 121.727 | 29.594 | 875788.198 | 120.9577 | 889543.113 |
| Bootstrap | $\widehat{\beta_1}$ | 0.053 | 0.058 | 0 | 0 | 0.005 | 0 |
| | $\hat{\sigma}^2$ | 0.7693 | 0.477 | 0.004 | 0.012 | -0.2923 | 0.098 |
| | $\beta_1^a$ | 0.053 | -0.01 | 0.139 | 19.221 | -0.063 | 19.206 |
| | $\sigma_a^2$ | 0.7693 | 3220.335 | 2961.135 | 8768318050 | 3219.5657 | 8769915333 |
| Jacknife | $\widehat{\beta_1}$ | 0.053 | 0.057 | 0 | 0 | 0.004 | 0 |
| | $\hat{\sigma}^2$ | 0.7693 | 0.51 | 0.003 | 0 | -0.2593 | 0.068 |
| | $\beta_1^a$ | 0.053 | 0.133 | 0.129 | 0.667 | 0.08 | 0.657 |
| | $\sigma_a^2$ | 0.7693 | 96.4 | 87.796 | 308322.929 | 95.6307 | 309760.012 |

For the estimation of both $\beta_1$ and $\sigma^2$, the OLS method shows lower variance,bias, and therefore lower MSE than the alternative way in all data simulation methods. The alternative estimator of the $\sigma_2$ is much larger than it under the OLS method. One of the reason is the alternative way is sensitive to the outliers during the resampling process. Comparing to the average value, the median looks more reasonable.

## 8.0 Discussion

### 8.1 The OLS method versus the alternative method

OLS estimator of $\beta_1$ is

$$\hat{\beta}_1 = \frac{S_{xy}}{S_x^2},$$

Which can write into:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} = \sum_{i=1}^n \frac{(x_i - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \frac{(y_i - \bar{y})}{(x_i - \bar{x})}.$$

Since this term is the slope of the line between point i to the center point of the data

$$\frac{(y_i - \bar{y})}{(x_i - \bar{x})},$$

we can conclude $\beta_1$ is a weighted average of slope of the line that connects the i-th point to the average of all points, the weight is determined by the distance of x to mean of x. The longer distance would be weighted more since small changes in its position will affect the slope connecting it to the center point more than closer points. We expect estimated Beta1 in OLS would be close to true Beta.

$$\hat{\sigma}^2 = \frac{SSE}{n-2}$$

This is an unbiased estimate of the population $\sigma^2$ So when we generate a large number of samples and the average should be close to the true $\sigma^2$.

The alternative estimator of $\beta_1$ is

$$\beta_1^a = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \bar{y})}{(x_i - \bar{x})}.$$

$\beta_1$ is the average slope for all the points in the sample, each point is equally weighted compared to OLS. This estimator would be less sensitive to the points far to the center. So it can not capture the change in y efficiently when the point is far from the center. Then the estimated Beta1 in the alternative method would be less close to true Beta.

$$\sigma_a^2 = \frac{SSE}{n}$$

The alternative estimator for sample variance is a biased estimator which will underestimate the true $\sigma^2$. But since the SSE was based on Alternative $\beta_1$, it would depend on the estimated value of y heavily. And since the Alternative estimator for Beta1 would be less accurate, the sample variance would be larger than expected.

### 8.2 Monte Carlo vs. Bootstrap vs. Jackknife

From the result table and compared to the theory, it basically matches our guessing. Using the OLS approach, the $\beta_1$ estimation was very close to the true beta, with 0 standard error and variance. Which means in every sample we generated, the estimator of $\beta_1$ is identical. The result in Bootstrap and Jackknife is slightly different from true beta which can be due to the replacement of the points in each sample in bootstrap and leaving-one-out for Jackknife.

As for the Alternative approach, we could find these estimated $\beta_1$ are not even close to the true beta, which leads to a huge variance in alternative estimator for variance.

### 8.3 The optimal procedure

Monte Carlo seems to have a better bias, variance and SEM since it was generating data from the sample distribution, while jackknife and bootstrap are using the real observed data.

### 8.4 Pros and cons for the three procedures

Jackknife's pros is more friendly to computational power, but the cons it might be less accurate than Monte Carlo and bootstrap since it can only provide n samples for studying. So when sample size is small this is not a good resampling method.

Bootstrap provides sample distributions with replacement, the pros is it provide a good estimates of uncertainty. The cons is if we did not generate a big enough number of samples,the information from one point might be more heavy than other points in the estimation. So the bootstrap is sensitive to number of samples we generate. Another problem is if the real sample does not represent the population, we can not obtain any useful estimation towards the population, since bootstrap only repeat the observation in the sample. Finally, bootstrap is computational intensive.

Monte Carlo can be useful when we have a correct belief about the population distribution. It is a good way to estimate uncertainty. The cons is very sensitive to the input of generating process and cost much computational power.