Final Project HUDM 6026

Getong Zhong & Ruoqiao Li

2023-05-09

Contents

Getong Zhong and Ruoqiao Li worked together in selecting the appropriate data for the project. For the coding aspect, Getong handled the data generation and the estimators. Ruoqiao wrote the code for Monte Carlo simulation and Bootstrap resampling. Both Getong and Ruoqiao collaborated to work on the Jackknife resampling method. In terms of the write-up, Getong took responsibility for the first three sections and the conclusion. Ruoqiao contributed to analysis the results of each methods

1 Select a motivating data set

Our main focus on this project is analyzing Walmart's sales records. Walmart, being one of the most recognized retail in the U.S., has a vast amount of data at its disposal. The data set we have chosen spans from 2010 to 2012 and contains the weekly sales records from 45 different Walmart stores and it is publicly available on Kaggle.

For our analysis, we plan to utilize a numeric outcome variable and a numeric predictor variable. The numeric outcome variable will be the weekly sales, which is a continuous variable. On the other hand, the numeric predictor variable could be one or multiple factors, including the Consumer Price Index (CPI), Unemployment Index, Temperature, Fuel price, and Holiday Index (whether the week is a special holiday week). Due to the requirement of this project, we only include CPI as the only predictor to make our model a simple linear regression model.

2 Data generation

Data generation part in this project in quite straightforward. We are going to simulate data from the simple linear regression model we create for the "Walmart_Store_sales" data set,with "Weekly_Sales" as the response variable and "CPI" as the predictor. After We got the value of intercept, and slope for the predictor from the model summary, we are going to record it for latter use in the data generation function to generate our own data from the real-life data.

```r
# Import data set
data <- read.csv("Walmart_Store_sales.csv", header = TRUE)
head(data)
```

```
##   Store       Date Weekly_Sales Holiday_Flag Temperature Fuel_Price      CPI
## 1     1 05-02-2010      1643691            0       42.31      2.572 211.0964
## 2     1 12-02-2010      1641957            1       38.51      2.548 211.2422
## 3     1 19-02-2010      1611968            0       39.93      2.514 211.2891
## 4     1 26-02-2010      1409728            0       46.63      2.561 211.3196
## 5     1 05-03-2010      1554807            0       46.50      2.625 211.3501
## 6     1 12-03-2010      1439542            0       57.79      2.667 211.3806
##   Unemployment
## 1        8.106
## 2        8.106
## 3        8.106
## 4        8.106
## 5        8.106
## 6        8.106
```

```r
# Run a simple linear regression model
model <- lm (Weekly_Sales ~ CPI, data = data)
summary(model)
```

```
##
## Call:
## lm(formula = Weekly_Sales ~ CPI, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -883689 -480911 -112047  385944 2783144
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1225673.7    31389.4  39.047  < 2e-16 ***
## CPI           -1041.6      178.3  -5.841 5.44e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 562900 on 6433 degrees of freedom
## Multiple R-squared:  0.005276,   Adjusted R-squared:  0.005121
## F-statistic: 34.12 on 1 and 6433 DF,  p-value: 5.438e-09
```

In the data generation function we wrote "dat_gen", we take 5 parameters: "n" determines the size of data we would like to generate; "beta_0" is the intercept value we got from the previous simple linear regression model; "beta_1" is the slope of the predictor we got from the previous simple linear regression model; "x_dist" is the distribution function we use to generate the error; and sigma is the standard deviation from the previous model, we use it to generate the error.

```
dat_gen <- function(n, beta_0, beta_1, x_dist, sigma) {
  # create new random variable
  x1 <- x_dist(n)
  # create new response variable
  y <- beta_0 + beta_1 * x1 + rnorm(n, mean = 0, sd = sigma)
  return(data.frame(X1 = x1, Weekly_sales = y))
}

data_simulation <- dat_gen(n = nrow(data), beta_0 = model$coefficients[1],
                           beta_1 = model$coefficients[2], x_dist = rnorm,
                           sigma = summary(model)$sigma)
head(data_simulation, 10)
```

```
##             X1 Weekly_sales
## 1   0.24197460     859165.6
## 2   0.25958987    1875475.3
## 3   0.02536233    2391152.8
## 4   0.03595969     624554.2
## 5  -0.84956871     942249.7
## 6  -0.02185795    1816711.1
## 7   0.18157570    1678906.7
## 8   1.96988041    2632571.6
## 9  -0.23744458     905422.8
## 10 -0.57361618    1252960.7
```

3 Estimators

In this section, we have developed a function, reg(), constructed entirely from scratch. This function ingests a data frame generated from the dat_gen() function and outputs estimates for two key parameters: the slope on X (denoted as beta1) and the error variance (denoted as sigma squared).

To generate these estimates, we have utilized two different estimators for each parameter. The first estimator for beta1 is the least squares estimator. It minimizes the sum of the squared residuals, providing the best linear unbiased estimate.

The second estimator for beta1 is an alternative estimator. Unlike the least squares estimator, this method involves averaging the ratio of the difference between each observed value and the mean of the observed values to the corresponding difference for predictor variables.

For sigma squared, the first estimator we used is the usual estimator, which calculates the sum of squared errors divided by the degree of freedom (n-2). This estimator reflects the average squared difference between the observed and predicted values, providing an estimate of the variance of the error term.

The second estimator for sigma squared is an alternative estimator. This estimator divides the sum of squared errors by the sample size (n), rather than by the degree of freedom. This is a more direct estimate of the average of squared errors.

These estimators enable us to obtain key parameters for our regression model, thereby allowing us to better understand the relationship between our predictor and response variables.

```
reg <- function(data) {
  n <- nrow(data)
  xbar <- mean(data[,1])
  ybar <- mean(data[,2])
  # estimate beta1 using the least squares estimator
  beta1 <- sum((data[,1] - xbar) * (data[,2] - ybar)) / sum((data[,1] - xbar)^2)

  # estimate beta1 using the alternative estimator
  beta1a <- 1 / mean((data[,2] - ybar) / (data[,1] - xbar))

  # estimate sigma^2 using the usual estimator
  sig2 <- sum((data[,2] - predict(lm(Weekly_sales ~ X1, data = data))^2) / (n - 2))

  # estimate sigma^2 using the alternative estimator
  sig2a <- sum((data[,2] - predict(lm(Weekly_sales ~ X1, data = data)))^2) / n

  return(list(beta1a = beta1,
              beta1_a = beta1a,
              sigma2 = sig2,
              sigma2a = sig2a))
}
```

```
reg_results <- reg(data_simulation)
```

4 Monte Carlo simulation

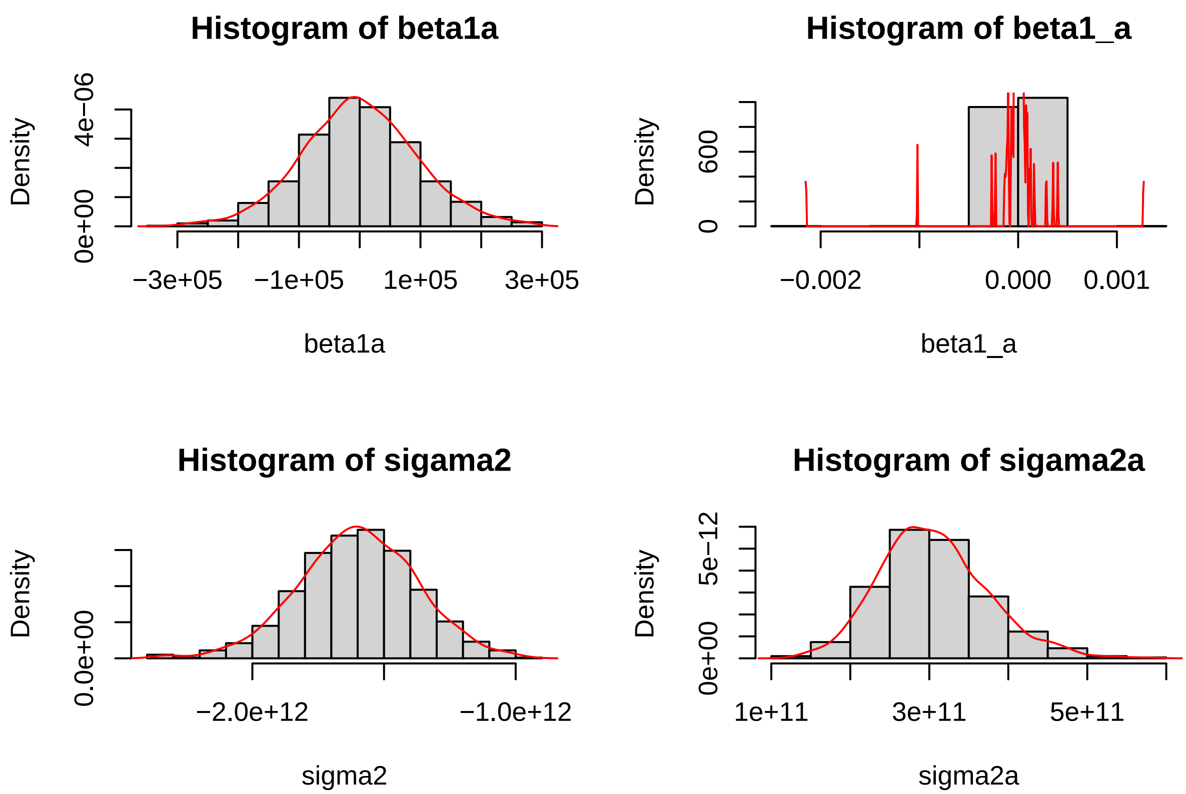
In this section, our goal is to run a Monte Carlo simulation from the dat_gen() function we have created previously. At first we set the sample size n to be 40 and number of replications R to be 1000. Then we defined the true values of the parameters beta_0, beta_1, and sigma using model from the Data generation section. After running the Monte Carlo simulation using replicate() we have named it as dat_list. We then apply the reg() function to our simulation and the result is saved as reg_results_mc.

```
set.seed(123)
library(plotrix)
# Define sample size and replications
n = 40
R = 1000
```

```r
beta_0 = model$coefficients[1]
beta_1 = model$coefficients[2]
# Monte Carlo simulation
dat_list <- replicate(n = R, expr = dat_gen(n = n, beta_0 = model$coefficients[1],
                                            beta_1 = model$coefficients[2],
                                            x_dist = rnorm, sigma = summary(model)$sigma),
                      simplify = FALSE)

# Apply reg() function to replication
reg_results_mc <- lapply(dat_list, reg)
```

The following codes generate histograms of sampling distributions with density plots for each of the four statistics from the function reg(): beta1a, beta1_a, sigma2, and sigma2a. From the shapes of these histograms we can say that they are likely to have normal distributions. The spread of the histogram for beta1_a is quite small compared to others and it is centered at 0. Based on this it seems like beta1_a is not a proper estimator for the data set.

```r
set.seed(123)
# Extract each estimator from reg_results2
beta1a <- sapply(reg_results_mc, function(x) x$beta1a)
beta1_a <- sapply(reg_results_mc, function(x) x$beta1_a)
sigma2 <- sapply(reg_results_mc, function(x) x$sigma2)
sigma2a <- sapply(reg_results_mc, function(x) x$sigma2a)

# Create histogram with density plot for each estimator
par(mfrow = c(2, 2))

hist(beta1a, freq = FALSE, main = "Histogram of beta1a")
lines(density(beta1a), col = "red")

hist(beta1_a, freq = FALSE, main = "Histogram of beta1_a")
lines(density(beta1_a), col = "red")

hist(sigma2, freq = FALSE, main = "Histogram of sigama2")
lines(density(sigma2), col = "red")

hist(sigma2a, freq = FALSE, main = "Histogram of sigama2a")
lines(density(sigma2a), col = "red")
```



We then estimate the means and standard error of the means for each of the 4 estimators. By using the lapply() function we firstly converted reg_results_mc into a data frame called result_mc. Then we obtain the means for beta1a, beta1_a, sigma2, sigma2a as 1.038477e+02, -1.288547e-06, -1.600346e+12, and 3.052965e+11. The standard error of the means for beta1a, beta1_a, sigma2, sigma2a are 3.000346e+03, 2.832999e-06, 7.028230e+09, and 2.142589e+09. Furthermore, we obtained the bias for each estimator from the difference between their means and reg_results. We used the apply() function to obtain their variances and the MSE is calculated as the square of bias plus the variance. The results are as the following:

```
set.seed(123)
# Make reg_results_mc a matrix for calculation
result_mc <- do.call(rbind, lapply(reg_results_mc, unlist))
# Means and standard error for each estimator based on Monte Carlo replications
means <- colMeans(result_mc)
# Estimate the bias, variance and MSE of each estimator
bias_beta1a <- means['beta1a'] - reg_results$beta1a
bias_beta1_a <- means['beta1_a'] - reg_results$beta1_a
bias_sigma2 <- means['sigma2'] - reg_results$sigma2
bias_sigma2a <- means['sigma2a'] - reg_results$sigma2a
bias <- c(bias_beta1a,bias_beta1_a,bias_sigma2,bias_sigma2a)
var <- c(var(beta1a), var(beta1_a), var(sigma2), var(sigma2a))
mse <- bias^2 + var
bias_var <- data.frame(
  Means = means,
  SE = std.error(result_mc),
  Bias = c(bias_beta1a, bias_beta1_a, bias_sigma2, bias_sigma2a),
  Varance = var,
```

```
  MSE = mse
)
bias_var
```

```
##                  Means           SE          Bias      Varance          MSE
## beta1a   1.038477e+02 3.000346e+03 -2.821711e+03 9.002073e+09 9.010035e+09
## beta1_a -1.288547e-06 2.832999e-06 -3.633266e-06 8.025886e-09 8.039087e-09
## sigma2  -1.600346e+12 7.028230e+09 -9.693031e+10 4.939601e+22 5.879150e+22
## sigma2a  3.052965e+11 2.142589e+09 -9.438101e+09 4.590688e+21 4.679766e+21
```

From the results, beta1_a has the minimum bias of -3.638866e-06 and is the closest to 0, making it less bias than the other 3 estimators. It also has the minimum variance of 8.025886e-09 and minimum MSE of 8.039127e-09 which makes it a better estimator overall.

5 Bootstrap

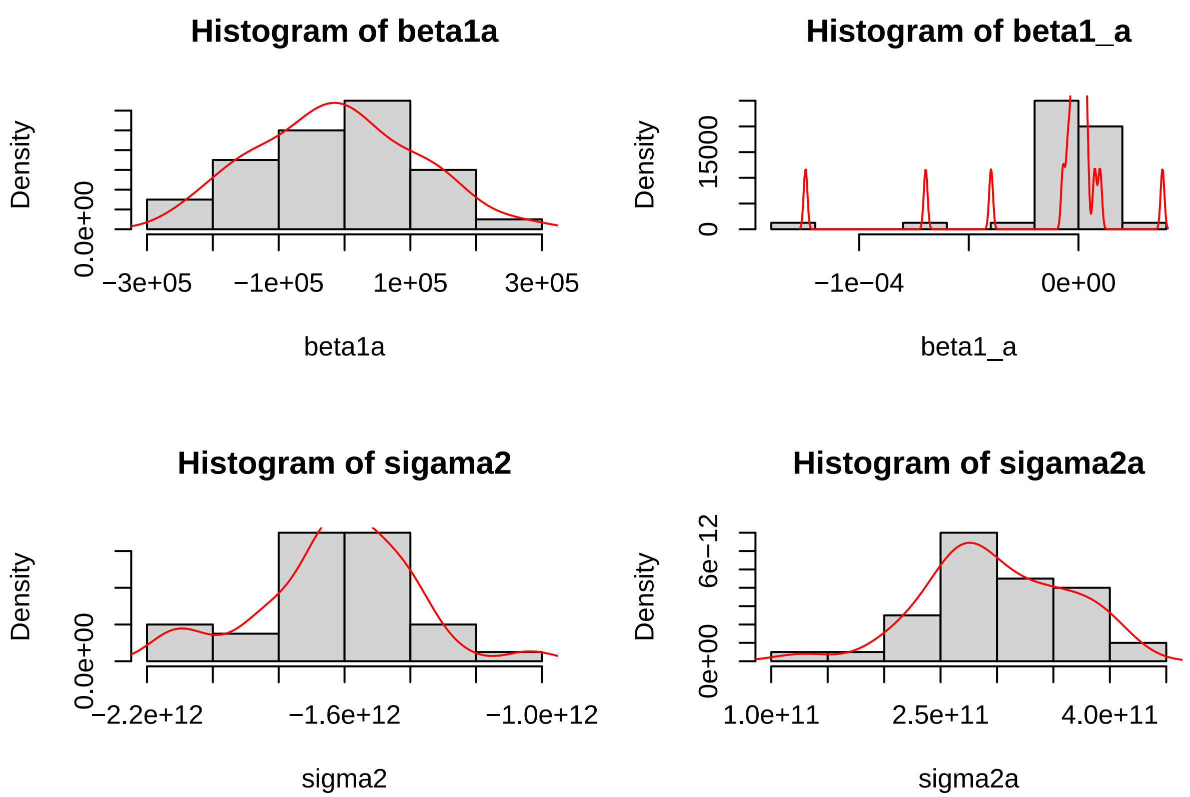
In this section we used the bootstrap method for data generation and we generate a single data set of sample size n =40 from the previous function dat_gen() and also a bootstrap replications B =500. Then we generated another data frame called data_boot utilizing the dat_gen function and we applied the previous reg() function to our bootstrap replications and the results are stored in reg_results_bs.

```
set.seed(123)
# Define sample size and bootstrap replications
n = 40
B = 500
data_boot <- dat_gen(n = n, beta_0 = beta_0, beta_1 = beta_1, x_dist = rnorm,
                     sigma = summary(model)$sigma)
# Perform bootstrap re sampling
dat_list1 <- replicate(n = n,
                       expr = dat_gen(n = n, beta_0 = beta_0, beta_1 = beta_1,
                                      x_dist = rnorm, sigma = summary(model)$sigma)
                       ,simplify = FALSE)


# Apply reg() function
reg_results_bs <- lapply(dat_list1, reg)
```

Similar from the previous section, the following codes generate histograms of sampling distributions with density plots for each of the four statistics from the function reg(): beta1a, beta1_a, sigma2, and sigma2a.Unlike the histograms from the previous section, histograms produced by the bootstrap method don't have a normal distribution in any of them.

```
set.seed(123)
# Extract each estimator from reg_results_bs

beta1a <- sapply(reg_results_bs, function(x) x$beta1a)
beta1_a <- sapply(reg_results_bs, function(x) x$beta1_a)
sigma2 <- sapply(reg_results_bs, function(x) x$sigma2)
sigma2a <- sapply(reg_results_bs, function(x) x$sigma2a)
par(mfrow = c(2, 2))
```

```r
# Create histogram with density plot for each estimator
hist(beta1a, freq = FALSE, main = "Histogram of beta1a")
lines(density(beta1a), col = "red")

hist(beta1_a, freq = FALSE, main = "Histogram of beta1_a")
lines(density(beta1_a), col = "red")

hist(sigma2, freq = FALSE, main = "Histogram of sigama2")
lines(density(sigma2), col = "red")

hist(sigma2a, freq = FALSE, main = "Histogram of sigama2a")
lines(density(sigma2a), col = "red")
```



We then performed a similar calculation on the means and standard error of means for the estimators and beta1_a still have the minimum mean of -4.865521e-06 and standard error of mean of 3.818443e-06. As for bias, variance and MSE, beta1_a also obtain the

```r
set.seed(123)
# Make reg_results_mc a matrix for calculation
result_bs <- do.call(rbind, lapply(reg_results_bs, unlist))
# Means and standard error for each estimator based on Monte Carlo replications
means <- colMeans(result_bs)
# Estimate the bias, variance and MSE of each estimator
bias_beta1a <- means['beta1a'] - reg_results$beta1a
bias_beta1_a <- means['beta1_a'] - reg_results$beta1_a
bias_sigama2 <- means['sigama2'] - reg_results$sigama2
```

```
bias_sigama2a <- means['sigama2a'] - reg_results$sigama2a
bias <- c(bias_beta1a,bias_beta1_a,bias_sigama2,bias_sigama2a)
var <- c(var(beta1a), var(beta1_a), var(sigma2), var(sigma2a))
mse <- bias^2 + var
bias_var <- data.frame(
  Estimator = c("beta1", "beta1a", "sigma2", "sigma2a"),
  Means = means,
  SE = std.error(result_bs),
  Bias = c(bias_beta1a, bias_beta1_a, bias_sigma2, bias_sigma2a),
  Varance = var,
  MSE = mse
)
bias_var
```

```
##         Estimator         Means           SE          Bias      Varance
## beta1a      beta1 -1.432241e+04 1.862947e+04 -1.724797e+04 1.388228e+10
## beta1_a    beta1a -4.865521e-06 3.818443e-06 -7.210241e-06 5.832203e-10
## sigma2     sigma2 -1.626023e+12 3.615355e+10 -9.693031e+10 5.228318e+22
## sigma2a   sigma2a  2.994576e+11 9.820779e+09 -9.438101e+09 3.857908e+21
##                  MSE
## beta1a  1.417977e+10
## beta1_a 6.352079e-10
## sigma2  5.228318e+22
## sigma2a 3.857908e+21
```

6 Jackknife

In this section, jackknife re sampling method was performed using the same data set from the previous bootstrap section. We used the jackknife method to estimate the bias and variance of the 4 estimators. The results are shown as the following. Tthe estimator beta1a still obtained the minimum bias of -2.616012e-05 and the minimum variance of 1.723865e-11.

```
set.seed(123)
# Compute the original estimates
original_estimates <- reg(data_boot)

# Initialize vectors to store the jackknife estimates
beta1_jack <- beta1a_jack <- sigma2_jack <- sigma2a_jack <- numeric(nrow(data_boot))

# Compute the jackknife estimates
for (i in 1:nrow(data_boot)) {
  jackknife_sample <- data_boot[-i, ]
  jackknife_estimates <- reg(jackknife_sample)
  beta1_jack[i] <- jackknife_estimates$beta1a
  beta1a_jack[i] <- jackknife_estimates$beta1_a
  sigma2_jack[i] <- jackknife_estimates$sigma2
  sigma2a_jack[i] <- jackknife_estimates$sigma2a
}

# Compute the jackknife estimate of bias
bias_beta1 <- (nrow(data_boot) - 1) * (mean(beta1_jack) - original_estimates$beta1a)
```

```r
bias_beta1a <- (nrow(data_boot) - 1) * (mean(beta1a_jack) - original_estimates$beta1_a)
bias_sigma2 <- (nrow(data_boot) - 1) * (mean(sigma2_jack) - original_estimates$sigma2)
bias_sigma2a <- (nrow(data_boot) - 1) * (mean(sigma2a_jack) - original_estimates$sigma2a)

# Compute the jackknife estimate of MSE
var_beta1 <- var(beta1_jack)
var_beta1a <- var(beta1a_jack)
var_sigma2 <- var(sigma2_jack)
var_sigma2a <- var(sigma2a_jack)



bias_var <- data.frame(
  Estimator = c("beta1", "beta1a", "sigma2", "sigma2a"),
  Bias = c(bias_beta1, bias_beta1a, bias_sigma2, bias_sigma2a),
  Varance = c(var_beta1, var_beta1a, var_sigma2, var_sigma2a)
)
bias_var
```

```
##   Estimator          Bias      Varance
## 1     beta1  4.746277e+03 2.952463e+08
## 2    beta1a -2.616012e-05 1.723865e-11
## 3    sigma2 -9.938378e+10 1.283803e+21
## 4   sigma2a -1.567491e+10 1.165621e+20
```

The first estimator for beta1 is the least squares estimator, i.e. beta1a. This is theoretically an unbiased estimator which will result in a bias of zero. The variance should be equal to its MSE since the bias is zero. The second estimator for beta1 is the alternative estimator, i.e. beta1_a. The bias of this estimator depends on the situation of the data set, as well as for the variance and MSE. The first estimator for sigma^2 is the usual estimator, i.e. sigma2. This is theoretically an unbiased estimator for the population variance. Similar from the least squares estimator, the variance should be equal to its MSE since its unbiased. The second estimator for sigma^2 is the alternative estimator, i.e. sigma2a. This is theoretically a biased estimator since it's divided by n, not n-2 like the usual estimator, but this difference could be neglected if we have a large sample size n.

Based on the situation of our data set, the Weekly_Sales column contains numbers that are relatively large and this factor contributes to the relative large numbers when we are calculating bias, variance and MSE for the estimators. From the results of the three different procedures, for the least squares estimator beta1a, the obtained bias is relatively small and the variance and MSE are really close which corresponds to the theory. Based on the data set we used, the alternative estimator berta1a obtained the minimum bias from all three procedures. For the last estimator sigma2a, since our sample size is relatively large, the obtained bias is also relatively small and the variance and MSE are close.

Based on the results, we compared the bias, variance and MSE of estimators between Monte Carlo, bootstrap, and jackknife procedures. Besides the bias for estimator beta1_a from Monte Carlo simulation and beta1a from jackknife procedure, all other estimators have negative bias. All three procedures generates relative small variance of the beta1_a estimator, as well as the mean square error for the beta_a estimator.

The Monte Carlo simulation tends to have a relative large variance over the other two procedures, as well as large MSE, and this is most likely caused by its data generation process, that it depends on the number of data sets generated. This makes the Monte Carlo method less reliable than the other two. The Monte Carlo method and bootstrap method have a relative small bias than the jackknife method. The jackknife method has negative bias besides the first estimator meaning only the first estimator overestimate the true value.

7 Conclusion

Monte Carlo simulations offer the advantage of flexibility and universality. These simulations handle very complex systems which are difficult to model analytically. They can incorporate a large number of variables and model their interactions. However the accuracy of Monte Carlo simulations depends largely on the number of iterations run, which can also increase computational load.

Bootstrap methods involve generating many subsamples from the original data and recalculating the estimator for each subsample. Bootstrapping can estimate the distribution of an estimator when the underlying distribution is unknown or complex. One of the main advantages of bootstrapping is that it makes fewer assumptions about the data compared to other methods. However, similar to Monte Carlo simulations, Bootstrap methods are also computationally intensive,

The jackknife method is another resampling technique that take out one observation at a time from the dataset and recalculating the estimator for bias and MSE. It is relatively simple and computationally efficient, especially compared to Bootstrap. Jackknife can provide robust estimates of bias and variance, even if the data contain outliers. However, it might not be efficient when complex estimators are presented.

In conclusion, the choice between Monte Carlo, bootstrap, and jackknife depends on the specific use case, including the nature of the data, the complexity of the estimator, and the computational resources available.

8 Reference

Thain, P. (2005). Jackknife Variance Estimation. Department of Mathematical Sciences, Montana State University. Retrieved from https://math.montana.edu/jobo/thainp/jack.pdf

Keller, B. (2023). Computational Statistics 06 - Bootstrap and Jackknife. Teachers College, Columbia University.

Keller, B. (2023). Computational Statistics 03 - Monte Carlo Simulation Studies. Teachers College, Columbia University.

Yasser, H. (2021). Walmart Sales Dataset. Kaggle. Retrieved from https://www.kaggle.com/datasets/yasserh/walmart-dataset