Statistical Learning Lab

Assignment - 4

Cross-validation and Bootstrapping

Submitted by,

Ben Abraham Biju

22IM10048

1. Load the dataset “manufacturing.csv”. Display the first few rows of the dataset. Take “Quality Rating” as the response variable.

The dataset was loaded using the read.csv() function and stored in the variable data.

```
> data <- read.csv("C:/Users/benab/OneDrive - iitkgp.ac.in/Desktop/Sem 6/SL Lab/Lab 2/manufacturing.csv", header = TRUE, stringsAsFactors = FALSE)
> head(data, 10)
   Temperature...C. Pressure..kPa. Temperature.x.Pressure Material.Fusion.Metric Material.Transformation.Metric Quality.Rating
1          209.7627       8.050855               1688.769               44522.22                        9229576       99.99997
2          243.0379      15.812068               3842.931               63020.76                       14355367       99.98570
3          220.5527       7.843130               1729.823               49125.95                       10728389       99.99976
4          208.9766      23.786089               4970.737               57128.88                        9125702       99.99997
5          184.7310      15.797812               2918.345               38068.20                        6303792      100.00000
6          229.1788       8.498306               1947.632               53136.69                       12037072       99.99879
7          187.5174      19.412851               3640.248               42478.69                        6593260      100.00000
8          278.3546       7.070944               1968.230               77834.82                       21567222       95.73272
9          292.7326      20.432896               5981.374               94223.15                       25084522       64.62360
10         176.6883      14.145782               2499.394               34049.37                        5515789      100.00000
> |
```

2. Fit polynomial models between Quality ~ Temp. Vary the degree of polynomial on temperature from 1 to 5 (temp, temp^2, temp^3 etc.). Perform LOOCV, k-fold CV for k=5 and 10 and compare the cross-validation MSE errors for different degrees of polynomials. Create a table showing the CV errors for different degrees of polynomials and for different CV techniques. Plot the results. Discuss which degree of polynomial is preferable.

The range of degrees of the polynomial is set from 1 to 5, and a null list is made for each type of cross validation method. The dataset is stored in the variable data sample.

```
data_sample <- data


degrees <- 1:5
cv.error.loocv <- rep(0, length(degrees))
cv.error.5fold <- rep(0, length(degrees))
cv.error.10fold <- rep(0, length(degrees))

for (i in degrees) {
  start_time <- Sys.time()
  cat("Starting polynomial degree:", i, "\n")

  glm.fit <- glm(Quality ~ poly(Temperature, i, raw = TRUE), data = data_sample)

  # LOOCV
  cv.loocv <- cv.glm(data_sample, glm.fit, K = nrow(data_sample))
  cv.error.loocv[i] <- cv.loocv$delta[1]

  # 5-fold CV
  cv.5 <- cv.glm(data_sample, glm.fit, K = 5)
  cv.error.5fold[i] <- cv.5$delta[1]

  # 10-fold CV
  cv.10 <- cv.glm(data_sample, glm.fit, K = 10)
  cv.error.10fold[i] <- cv.10$delta[1]

  end_time <- Sys.time()
  cat("Finished degree:", i, "in", round(difftime(end_time, start_time, units = "secs"), 2), "seconds\n")
}
```

```
Starting polynomial degree: 1
Finished degree: 1 in 20.3 seconds
Starting polynomial degree: 2
Finished degree: 2 in 21.97 seconds
Starting polynomial degree: 3
Finished degree: 3 in 25.13 seconds
Starting polynomial degree: 4
Finished degree: 4 in 28.43 seconds
Starting polynomial degree: 5
Finished degree: 5 in 32.67 seconds
```

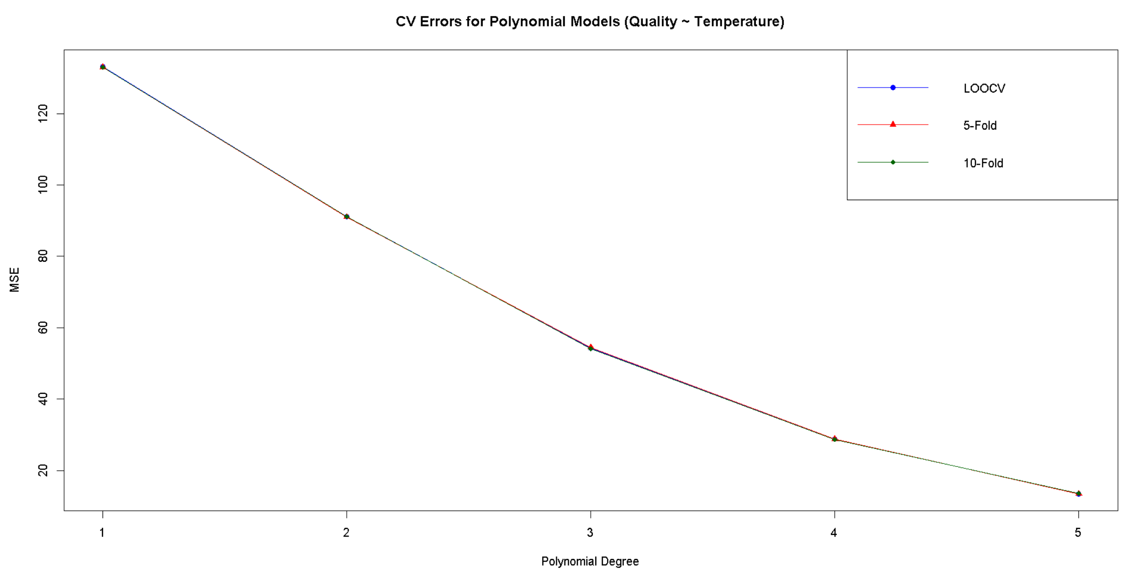
Once the loop has been executed for all degrees, the errors are stored as a dataframe and printed.

```
poly.cv.results <- data.frame(
  Degree = degrees,
  LOOCV_MSE = cv.error.loocv,
  CV5_MSE = cv.error.5fold,
  CV10_MSE = cv.error.10fold
)
```

```
[1] "CV Errors for Polynomial Models (Quality ~ Temperature):"
> print(poly.cv.results)
  Degree LOOCV_MSE   CV5_MSE  CV10_MSE
1      1 133.07880 132.88638 133.36005
2      2  91.19322  91.26217  91.21856
3      3  54.23607  54.20037  54.57484
4      4  28.78949  28.64342  28.79402
5      5  13.52725  13.46535  13.47917
```

The errors for different degrees of polynomial and different methods are plotted

```
plot(degrees, cv.error.loocv, type = "o", col = "blue", pch = 16,
     xlab = "Polynomial Degree", ylab = "MSE",
     main = "CV Errors for Polynomial Models (Quality ~ Temperature)")
lines(degrees, cv.error.5fold, type = "o", col = "red", pch = 17)
lines(degrees, cv.error.10fold, type = "o", col = "darkgreen", pch = 18)
legend("topright", legend = c("LOOCV", "5-Fold", "10-Fold"),
       col = c("blue", "red", "darkgreen"), pch = c(16,17,18), lty = 1)
```



The results show that the least MSE occurs for 5-fold validation, when the degree of temperature is 5. Although LOOCV is computationally more intense, here, 5-fold validation appears to show slightly better results. When applied to the whole dataset, the MSE values of all CV methods are almost similar.

3. Perform the analysis in problem no. 2, but this time, fit linear models with different combinations of X variables, without interaction. Discuss which model is most preferable based on the cross-validation results. Plot the results and on X-axis labels, provide the X-variable combinations used in the model, e.g. (temp, temp-press, temp-matfus, temp-matfus-mattr etc.)

To train linear models on different combinations of X, we have to create a formula list which enlists all the different possible combinations among the five predictor variables. So, disregarding the null model, there will be a total of $2^5 - 1 = 31$ possible combinations

```
predictors <- c("Temperature", "Pressure", "Temp_x_Press", "MatFusion", "MatTransform")
model.formulas <- list()

# Generate all possible predictor combinations
for (i in 1:length(predictors)) {
  cmb <- combn(predictors, i, simplify = FALSE)
  for (combo in cmb) {
    formula_str <- paste("Quality ~", paste(combo, collapse = " + "))
    formula_obj <- as.formula(formula_str)
    model.formulas[[formula_str]] <- formula_obj
  }
}

n.models <- length(model.formulas)
```

Again, creating an empty list is created for storing errors and the models are trained.

```
cv.error.loocv <- rep(0, n.models)
cv.error.5fold <- rep(0, n.models)
cv.error.10fold <- rep(0, n.models)

model.names <- names(model.formulas)

for (i in 1:n.models) {
  start_time <- Sys.time()
  cat("Starting model", i, ":", model.names[i], "\n")

  glm.fit <- glm(model.formulas[[i]], data = data_sample)

  # LOOCV
  cv.loocv <- cv.glm(data_sample, glm.fit)
  cv.error.loocv[i] <- cv.loocv$delta[1]

  # 5-fold CV
  cv.5 <- cv.glm(data_sample, glm.fit, K = 5)
  cv.error.5fold[i] <- cv.5$delta[1]

  # 10-fold CV
  cv.10 <- cv.glm(data_sample, glm.fit, K = 10)
  cv.error.10fold[i] <- cv.10$delta[1]

  end_time <- Sys.time()
  duration <- round(as.numeric(difftime(end_time, start_time, units = "secs")), 2)
  cat("Finished model", i, "in", duration, "seconds. LOOCV MSE =", cv.error.loocv[i],
      "5-fold MSE =", cv.error.5fold[i], "10-fold MSE =", cv.error.10fold[i], "\n\n")
}
```

The first few and last few rows of training output are shown below.

```
Starting model 1 : Quality ~ Temperature
Finished model 1 in 16.39 seconds. LOOCV MSE = 133.0788 5-fold MSE = 133.0121 10-fold MSE = 133.0904

Starting model 2 : Quality ~ Pressure
Finished model 2 in 16.31 seconds. LOOCV MSE = 168.8959 5-fold MSE = 168.9318 10-fold MSE = 168.8616

Starting model 3 : Quality ~ Temp_x_Press
Finished model 3 in 17 seconds. LOOCV MSE = 157.702 5-fold MSE = 157.6551 10-fold MSE = 157.7967
```

……….

……….

```
Starting model 28 : Quality ~ Temperature + Pressure + MatFusion + MatTransform
Finished model 28 in 21.09 seconds. LOOCV MSE = 83.76925 5-fold MSE = 84.0066 10-fold MSE = 83.68985

Starting model 29 : Quality ~ Temperature + Temp_x_Press + MatFusion + MatTransform
Finished model 29 in 21.13 seconds. LOOCV MSE = 84.30565 5-fold MSE = 84.23364 10-fold MSE = 84.45012

Starting model 30 : Quality ~ Pressure + Temp_x_Press + MatFusion + MatTransform
Finished model 30 in 21.21 seconds. LOOCV MSE = 85.96859 5-fold MSE = 86.25406 10-fold MSE = 86.16316

Starting model 31 : Quality ~ Temperature + Pressure + Temp_x_Press + MatFusion + MatTransform
Finished model 31 in 23.48 seconds. LOOCV MSE = 83.83607 5-fold MSE = 84.00075 10-fold MSE = 83.79882
```

The error lists are converted to a dataframe and plotted.

```
combination.cv.results <- data.frame(
  Model = model.names,
  LOOCV_MSE = cv.error.loocv,
  CV5_MSE = cv.error.5fold,
  CV10_MSE = cv.error.10fold
)
```

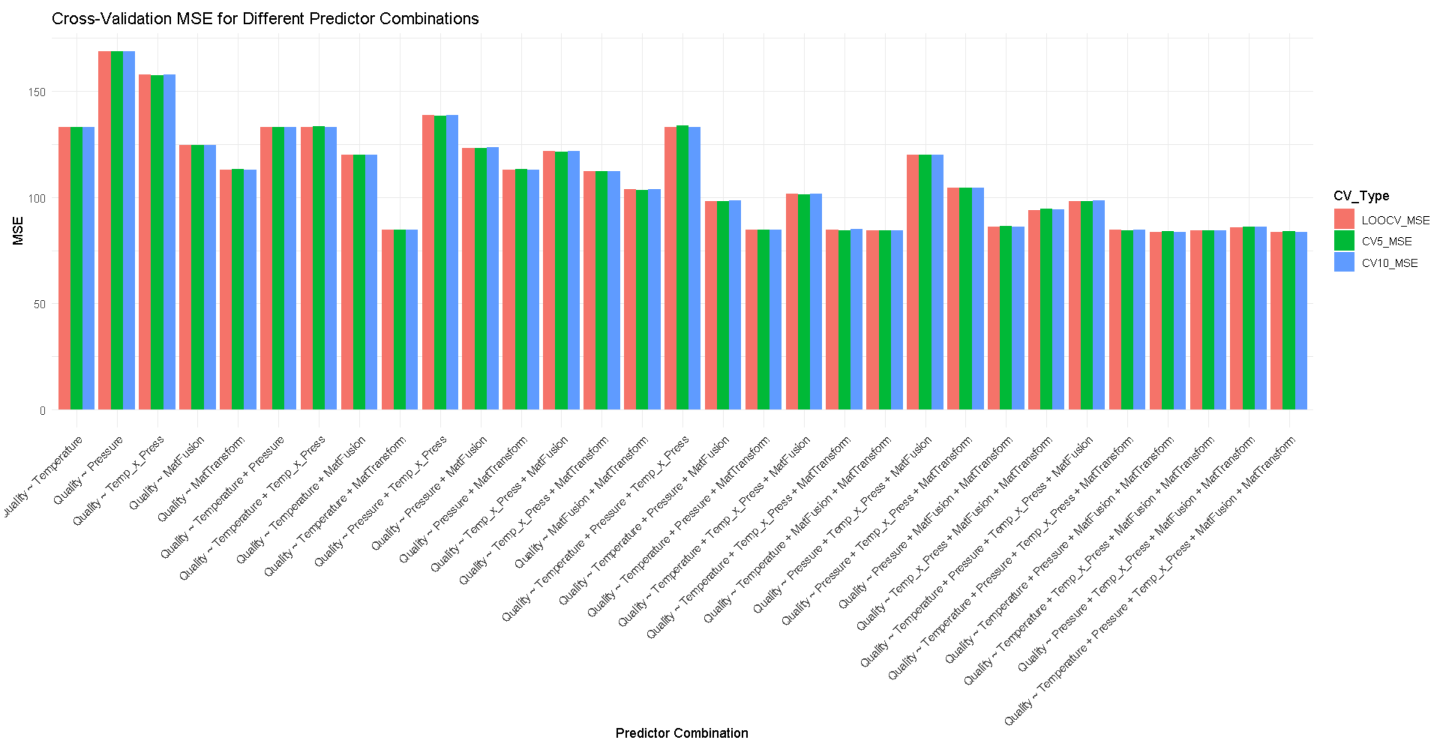
```
[1] "Cross-Validation Errors for Different Predictor Combinations:"
> print(combination.cv.results)
                                                                 Model LOOCV_MSE   CV5_MSE  CV10_MSE
1                                                 Quality ~ Temperature 133.07880 133.01212 133.09036
2                                                    Quality ~ Pressure 168.89589 168.93182 168.86156
3                                                Quality ~ Temp_x_Press 157.70200 157.65512 157.79669
4                                                   Quality ~ MatFusion 124.77012 124.77034 124.65829
5                                                Quality ~ MatTransform 113.07841 113.24593 112.91921
6                                      Quality ~ Temperature + Pressure 133.14410 133.22270 133.21400
7                                  Quality ~ Temperature + Temp_x_Press 133.19586 133.62983 133.25670
8                                     Quality ~ Temperature + MatFusion 119.93438 119.98925 119.92527
9                                  Quality ~ Temperature + MatTransform  84.59484  84.73901  84.72504
10                                    Quality ~ Pressure + Temp_x_Press 138.69882 138.60187 138.70359
11                                       Quality ~ Pressure + MatFusion 123.41995 123.26544 123.45438
12                                    Quality ~ Pressure + MatTransform 113.13425 113.22824 113.07118
13                                   Quality ~ Temp_x_Press + MatFusion 121.81381 121.60052 121.76785
14                                Quality ~ Temp_x_Press + MatTransform 112.26011 112.17882 112.12062
15                                   Quality ~ MatFusion + MatTransform 103.82232 103.60521 103.78660
16                      Quality ~ Temperature + Pressure + Temp_x_Press 133.25959 133.94078 133.08580
17                         Quality ~ Temperature + Pressure + MatFusion  98.12928  98.18716  98.37655
18                      Quality ~ Temperature + Pressure + MatTransform  84.63338  84.71118  84.72145
19                     Quality ~ Temperature + Temp_x_Press + MatFusion 101.81007 101.49119 101.76008
20                  Quality ~ Temperature + Temp_x_Press + MatTransform  84.66181  84.48618  84.94776
21                     Quality ~ Temperature + MatFusion + MatTransform  84.50945  84.45073  84.47078
22                        Quality ~ Pressure + Temp_x_Press + MatFusion 120.03935 120.23471 120.20577
23                     Quality ~ Pressure + Temp_x_Press + MatTransform 104.47502 104.51322 104.67427
24                        Quality ~ Pressure + MatFusion + MatTransform  86.29017  86.48656  86.28865
25                    Quality ~ Temp_x_Press + MatFusion + MatTransform  94.06184  94.69683  94.22870
26          Quality ~ Temperature + Pressure + Temp_x_Press + MatFusion  98.23669  98.21705  98.42299
27       Quality ~ Temperature + Pressure + Temp_x_Press + MatTransform  84.71167  84.44575  84.72333
28          Quality ~ Temperature + Pressure + MatFusion + MatTransform  83.76925  84.00660  83.68985
29      Quality ~ Temperature + Temp_x_Press + MatFusion + MatTransform  84.30565  84.23364  84.45012
30         Quality ~ Pressure + Temp_x_Press + MatFusion + MatTransform  85.96859  86.25406  86.16316
31 Quality ~ Temperature + Pressure + Temp_x_Press + MatFusion + MatTransform  83.83607  84.00075  83.79882
```

From different combinations, we check which combination gives the least MSE for any CV method applied.

```
> print(paste("Minimum MSE:", min_mse_value))
[1] "Minimum MSE: 83.6898466534485"
> print("Row(s) with the minimum MSE:")
[1] "Row(s) with the minimum MSE:"
> print(min_mse_row)
                                                         Model LOOCV_MSE CV5_MSE CV10_MSE
28 Quality ~ Temperature + Pressure + MatFusion + MatTransform  83.76925 84.0066 83.68985
```

The minimum MSE observed is **83.6898**. The predictor variables involved are ***Temperature, Pressure, Material Fusion Metric*** and ***Material Transform Metric***. The minimum MSE was observed with the **10-fold validation method**.

```r
cv.melted <- melt(combination.cv.results, id.vars = "Model", variable.name = "CV_Type", value.name = "MSE")

ggplot(cv.melted, aes(x = Model, y = MSE, fill = CV_Type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Cross-Validation MSE for Different Predictor Combinations",
       x = "Predictor Combination",
       y = "MSE") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



4. Generate 50 random numbers from Normal Distribution $N(\mu = 50, \sigma^2 = 2)$. Now create 100 bootstrap samples with 20 data points each, with replacement. Estimate the mean and variance of the population from the bootstrap samples.

A new seed is set to generate 50 random numbers from the distribution $N(\mu = 50, \sigma^2 = 2)$.

```
> set.seed(789)
> pop_sample <- rnorm(50, mean = 50, sd = sqrt(2))
> head(pop_sample)
[1] 50.74118 46.80279 49.97217 50.25900 49.48897 49.31484
```

The bootstrap sample size is set as 20 and 100 such samples are generated after setting a new seed.

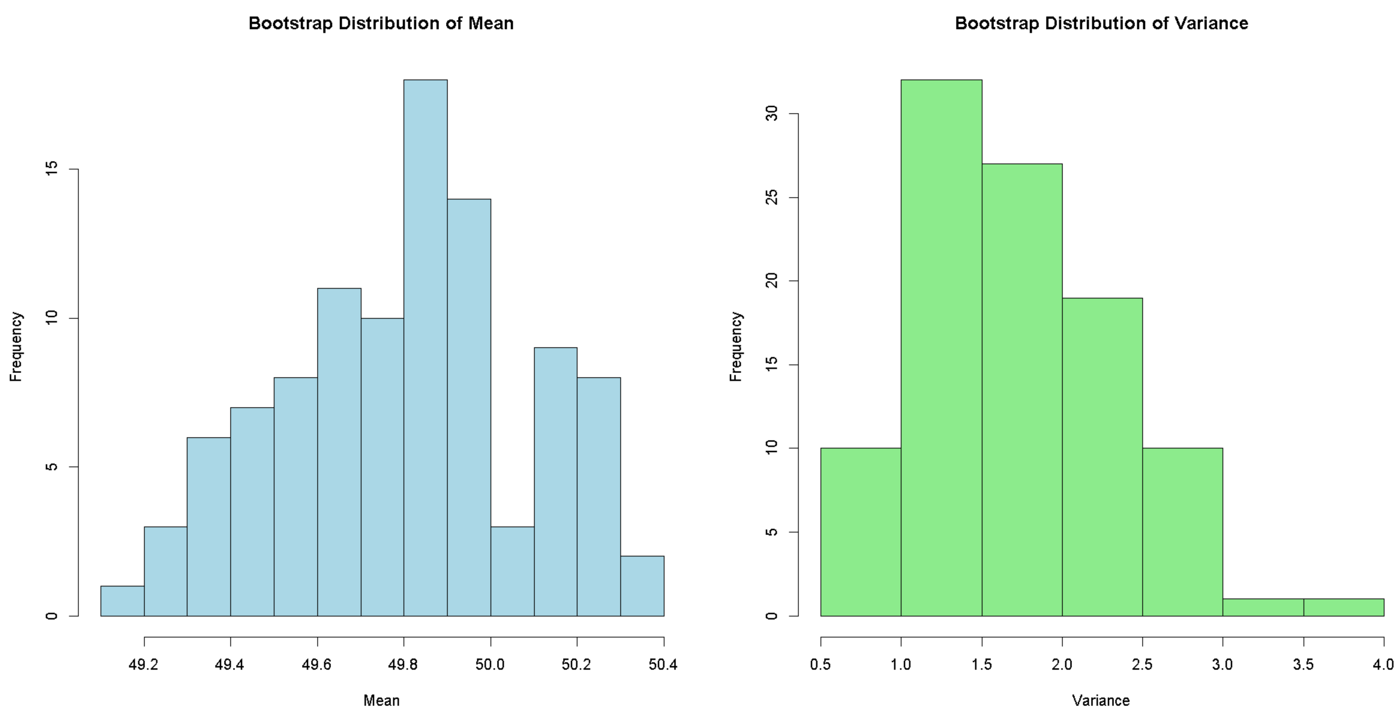
```r
n_boot <- 100
boot_sample_size <- 20

boot.means <- rep(0, n_boot)
boot.vars  <- rep(0, n_boot)

set.seed(456)

for (i in 1:n_boot) {
  boot.sample <- sample(pop_sample, size = boot_sample_size, replace = TRUE)
  boot.means[i] <- mean(boot.sample)
  boot.vars[i]  <- var(boot.sample)
}

boot.mean.estimate <- mean(boot.means)
boot.var.estimate  <- mean(boot.vars)
```

The estimated mean and variance from different samples are as follows.

```
> cat("Bootstrap Estimation Results (", n_boot, "samples of size", boot_sample_size, "):\n")
Bootstrap Estimation Results ( 100 samples of size 20 ):
> cat("Estimated Mean:", boot.mean.estimate, "\n")
Estimated Mean: 49.79905
> cat("Estimated Variance:", boot.var.estimate, "\n")
Estimated Variance: 1.699106
```

The frequency chart is plotted from the means and variance estimated from different samples.

```
par(mfrow = c(1,2))
hist(boot.means, col = "lightblue", main = "Bootstrap Distribution of Mean",
     xlab = "Mean", breaks = 10)
hist(boot.vars, col = "lightgreen", main = "Bootstrap Distribution of Variance",
     xlab = "Variance", breaks = 10)
par(mfrow = c(1,1))
```



Conclusions and Discussion:

- 5-fold validation yields lower MSE thanLOOCV and 10-fold validation, even though LOOCV requires increased computational resources. Higher order polynomials tend to have lower MSE values across all CV methods, but have a risk of overfitting and loss of generalization.
- Among the 31 possible combinations of predictor variables, the model based on *Pressure, Temperature, Material Fusion Metric and Material Transform Metric,* gave the lowest MSE, using 10-fold technique. This shows that interaction effects and selection of suitable predictors affect the model performance. This might be an exception when 10-fold validation gives better results than LOOCV.

- Bootstrap resampling provided an estimate of the population mean and variance from 100 resampled datasets.The predicted variance was close to the actual variance ($\sigma^2 = 2$),
showing the effectiveness in predicting population parameters.