An Analysis of Math 120 Scores

Presented to

CSUF

COLLEGE OF

Natural Sciences and Mathematics

California State University, Fullerton

Math 537 Summer 2023

Prepared by

Emilio VASQUEZ

August 1, 2023

Contents

1 Overview and Summary

Using Math 120 (Elementary Statistics) exam scores from the California State University Math Department, the department wanted to find ranges for the average scores of three exams that were given throughout the semester. The issue at hand is that some the exams have missing records due to students not taking the exams which would skew these ranges potentially by a significant margin. This analysis addresses a way to estimate these average ranges by using the EM algorithm to impute missing data. Below is a summary of the findings at hand:

- Imputed Average: $\begin{bmatrix} 75.35465 & 78.70035 & 78.14713 \end{bmatrix}$
- Imputed Covariance Matrix: $\begin{bmatrix} 197.54805 & 83.62426 & 87.81434 \\ 83.62426 & 64.82909 & 47.25546 \\ 87.81434 & 47.25546 & 59.87850 \end{bmatrix}$
- The average scores for exam 1,2, and the final exam will on average fall simultaneously below 95% of the time:

Exam	Lower Bound	Upper Bound
Exam 1	72.08706	78.62224
Exam 2	76.82848	80.57223
Final Exam	76.34814	79.94611

2 Analysis

As mentioned above, there were various instances where students did not take either 1 or 2 exams throughout the course of the semester for Math 120. This is where the EM algorithm comes into play as it is a cornerstone in estimating parameters, such as mu and the covariance matrix request, when data is missing through an iterative process. In addition it can be used to impute these missing values.

2.1 Imputed Data Results

Using the EM algorithm allows one to impute missing data. Below are results of what the EM algorithm estimated missing data points to be. Of the 3 exams provided, Exam 2 had the largest amount of missing data with 9 exam scores missing. Figure 1 shows the letter grade distribution before and after data imputation. The grade ranges are common ranges used where an A is 90 and above, B is 80 through 89, so on and so forth. Across all exams, it was mainly the mid range grade grades B, C, and D that saw data imputed as A and F letter grades were unaffected.

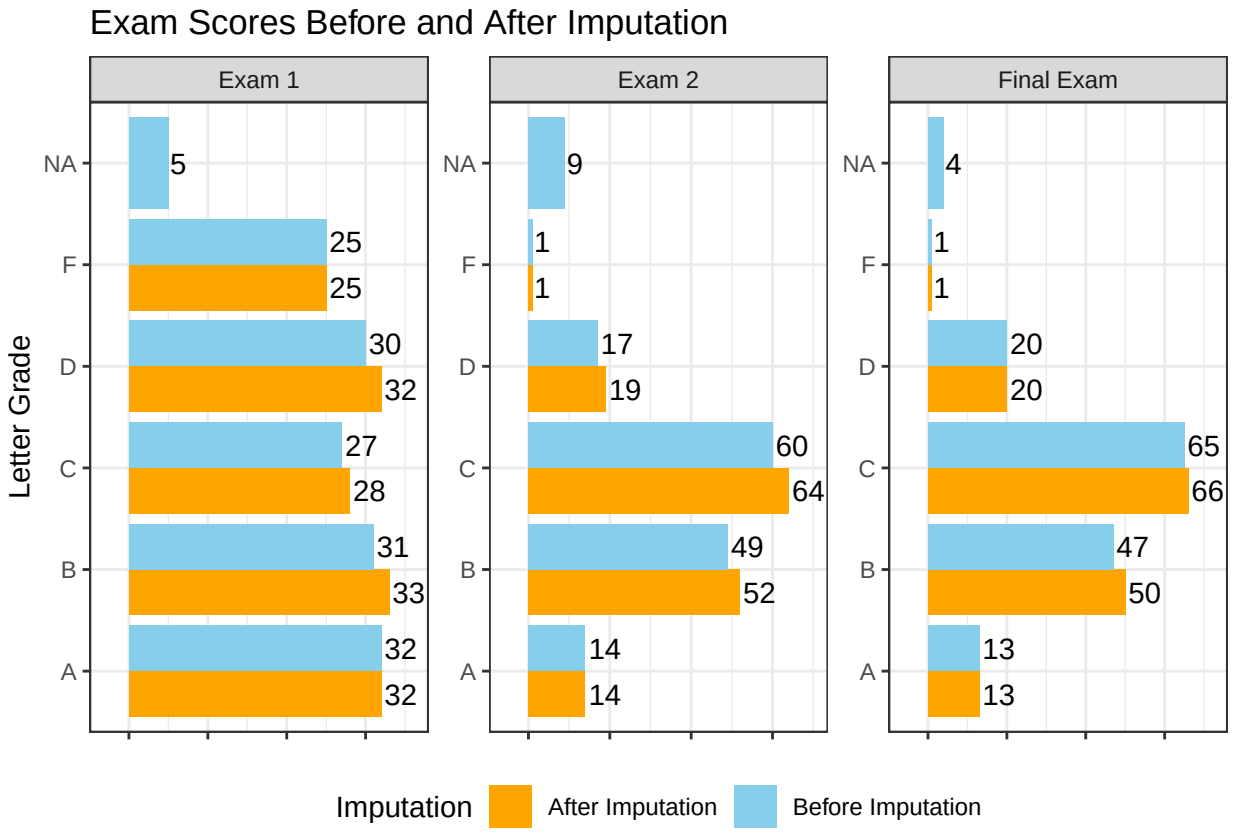

Figure 1. Exam grades before and after imputation

2.2 Simultaneous Confidence Intervals

Below are the exam score distributions along with their 95% simultaneous confidence interval (CI) that were calculated by using the EM algorithm. This is essentially saying that the exam scores will simultaneously fall within the below ranges 95% of the time despite having missing exam scores. It is important to use a simultaneous confidence interval because it takes into account the potential relationship in grades between the three exams as opposed to a marginal confidence interval where that relationship is not taken into account.

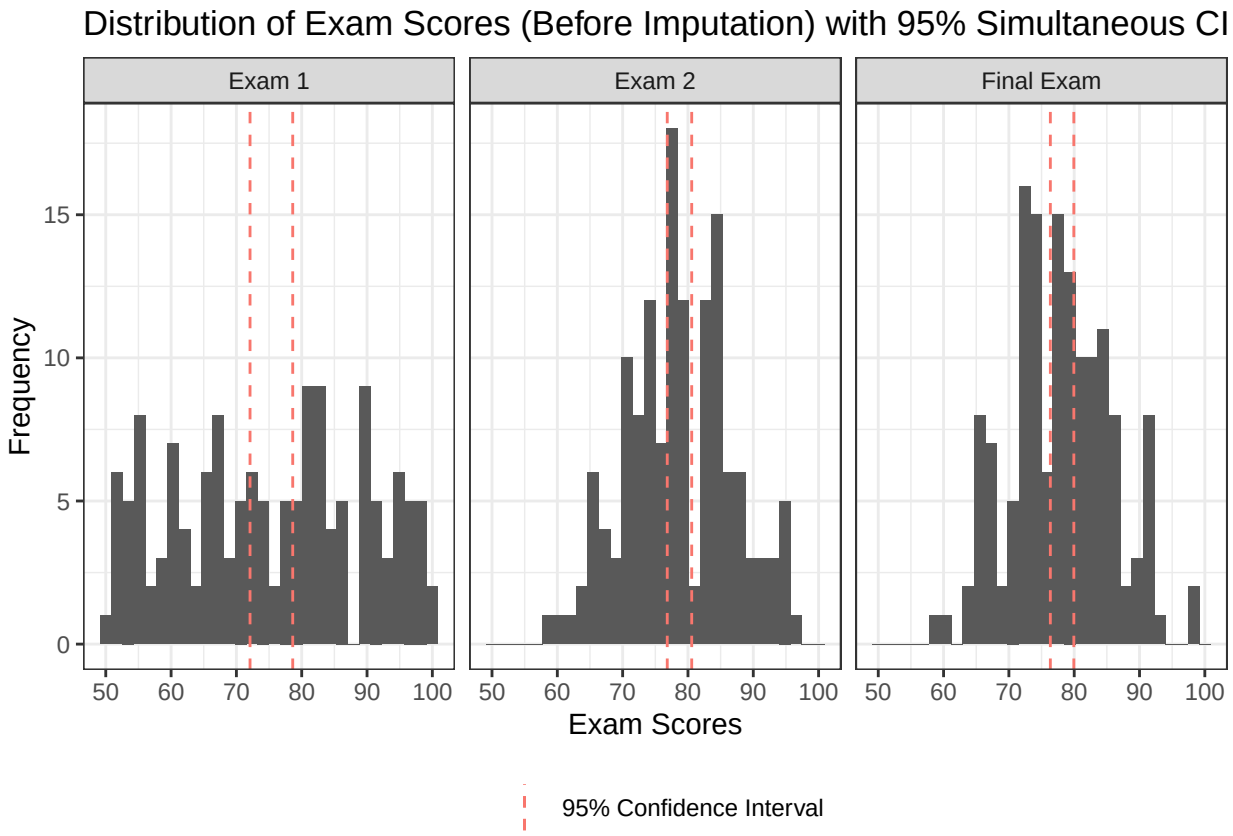

Figure 2. Distribution of exam grades with 95% simultaneous confidence intervals plotted

Exam	Lower Bound	Upper Bound
Exam 1	72.08706	78.62224
Exam 2	76.82848	80.57223
Final Exam	76.34814	79.94611

Figure 3. Table of calculated simulataneous confidence intervals

3 Appendix

3.1 A.1 EM Algorithm Code

```
# EM algorithm as a function
# Mixture of old 534 HW and Jonathan's super optimized, better than mine, code

EM <- function(y, mu, sig, maxit = 1000) {

  ########
  # Inputs
  ########

  # y: data set given to us
  # mu: initial value for mu
  # sig: initial value for sigma
  # maxit: max iterations to run EM algo. Default set to 20

  # In 534, we had convergence criteria that was also included.
  # In the essence of time, convergence criteria will be omitted.
  # HOWEVER! The algorithm did reach convergence after about 30 or so iterations

  # Grab dimensions of data
  n <- dim(y)[1]  # Number of observations
  p <- dim(y)[2]  # Number of columns

  # Calculate observed mu's and place in vector
  mu0 <- matrix(0, p, 1)
  for (i in 1:p) {
    # Calculate mean of obs values for each variable
    mu0[i] <- sum(na.omit(y[i])) / sum(!is.na(y[i]))
  }
```

```r
# Loop for a max number of iterations

for (i in 1:maxit) {

  ############
  # E Step - impute missing values and update observed mu's
  ############

  # Initialize matrix that sums observed and missing val
  mu_new <- matrix(0, p, 1)
  # Initialize matrix for that stores observed and imputted values
  y_imputed <- matrix(0, n, p)

  #Loop through all the records in the data
  for (i in 1:n) {

    # Identify what is missing and what is not in the data
    obs <- which(!is.na(y[i,]))  # Indices of observed values
    mis <- which(is.na(y[i,]))  # Indices of missing values

    # Create cases for patterns of data
    # The row is either complete or it is not

    # Case 1: Working with incomplete data
    if (length(mis) > 0) {
        ystar_m <- t(t(mu[mis])) +
          sig[mis, obs] %*% solve(sig[obs, obs]) %*%
          t((y[i, obs] - mu0[obs]))

        # Summing the imputed missing values
        mu_new[mis] <- mu_new[mis] + ystar_m

        # Impute data into missing rows
```

```r
      y_imputed[i, mis] <- ystar_m
      y_imputed[i, obs] <- as.matrix(y[i, obs])
    }

  # Case 2: Working with complete data
    if (length(mis) == 0) {
      mu_new <- mu_new + y[i,]  # Summing the observed values
      y_imputed[i,] <- as.matrix(y[i,])
    } else {
      mu_new[obs] = t(t(mu_new[obs])) + y[i,obs]  # Summing the observed values
    }
  }

  mu_new <- mu_new / n  # Calculate the mean
  E_step_results <- (list('mu_new' = t(mu_new), 'y_imputed' = y_imputed))

X <- E_step_results$mu_new
y_imputed <- E_step_results$y_imputed # Store imputed results

######################
# M-step: Update sigma (covariance matrix)
######################

# Initialize matrices to store values for sigma updates
sum <- matrix(0, p, p)
sig_i <- matrix(0, p, p)

for (i in 1:n) {
  obs <- which(!is.na(y[i,]))  # Indices of observed values
  mis <- which(is.na(y[i,]))  # Indices of missing values

  ystar_m <- as.numeric(mu[mis])+sig[mis, obs] %*% solve(sig[obs, obs]) %*%
    t((y[i, obs] - mu0[obs]))
```

```r
  # Case 1: Working with complete data
  if (length(mis) == 0) {
      # Case 1: Complete data
      sig_i = t(y[i,obs]) %*% as.numeric(y[i,obs])
    } else {

      # Case 2: Incomplete data for sigma

      # Formulas from 534 that update the sigma matrix
      # Place calculated areas into the sig matrix based on obs or mis
      # yobs_yobs
      sig_i[obs,obs] = t(y[i,obs]) %*% as.numeric(y[i,obs])

      # y_obs_Ey_mis
      sig_i[obs,mis] = t(y[i,obs]) %*% t(ystar_m)

      # Ey_mis_y_obs
      sig_i[mis,obs] = ystar_m %*% as.numeric(y[i,obs])

      # Ey_mis_y_mis
      sig_i[mis,mis] = sig[mis,mis] - sig[mis,obs] %*%
        solve(sig[obs,obs]) %*% sig[obs,mis] + ystar_m %*% t(ystar_m)
    }
    # Sum of updated sigma
    sum <- sum + sig_i  # Summing up sigma for each observation
  }

S <- sum / n  # Calculate the mean

# Update sigma final time
sig <- (S - X %*% t(X))
```

```r
    # Update mu final time
    mu <- X
  }
  return(list('mu' = mu, 'sigma' = sig, 'y_imputed' = y_imputed))  # Return the updated mu, sig
}
```