Missing Data | Homework 3

Brian Cervantes Alvarez
March 5, 2025

Main Findings

Problem 1

Part A Results

I simulated data for 1000 subjects, each with 5 covariates measured at 3 time points. I randomly chose the true parameters and padded each subject's outcome vector to have 3 measurements (missing values were marked "NA" for those who dropped out). I then used the ECM algorithm to estimate the model parameters.

- **Convergence:** The Q-function started at -2980.9778 (Iteration 1) and improved until it reached -287.5468 at Iteration 31.

- **Iteration Count:** The algorithm converged in **31 iterations**, with a final difference of about 7×10^{-6}.

Below is the estimated covariance matrix:

$$\hat{\Sigma} = \begin{pmatrix} 3.7669305 & 2.9530130 & -0.9207553 \\ 2.9530129 & 2.7803850 & -2.1797457 \\ -0.9207553 & -2.1797460 & 4.8424833 \end{pmatrix}.$$

The table shows the **final coefficient estimates**, their **standard errors**, and the **95% confidence intervals**:

Parameter	Estimate	Std. Error	95% CI Lower	95% CI Upper
β_1	0.0839916	0.001794170	0.08047503	0.08750817
β_2	1.1878703	0.001806340	1.18432983	1.19141068
β_3	-0.8272324	0.001809628	-0.83077925	-0.82368551
β_4	-0.3056589	0.001780572	-0.30914881	-0.30216897
β_5	-0.9184975	0.001825956	-0.92207639	-0.91491865

The very narrow confidence intervals suggest a high level of precision in our estimates—this makes sense given our large sample size and the way I set up the simulation. In most repeated simulations, these intervals would be expected to include the true values used to generate the data.

Part B Results

The Q-function went from about -20231.48 to -19889.58 over **29 iterations**.

Below is the estimated covariance matrix:

$$\hat{\Sigma} = \begin{pmatrix} 194.5852 & 163.1883 & 160.0425 \\ 163.1883 & 418.2026 & 374.7859 \\ 160.0425 & 374.7859 & 496.2113 \end{pmatrix}.$$

The table below shows the **final coefficient estimates**, their **standard errors**, and the **95% confidence intervals**:

Parameter	**Estimate**	**Std. Error**	**95% CI Lower**	**95% CI Upper**
β_1	-5.21513003	0.37056342	-5.94143430	-4.48882572
β_2	-0.86183388	0.37056342	-1.58813820	-0.13552957
β_3	-1.45963556	0.06266164	-1.58245240	-1.33681875
β_4	-0.02500182	0.06266164	-0.14781860	0.09781499

The ECM algorithm converged after 29 iterations. Even though I don't know the *true* parameter values for real data, these estimates and confidence intervals help us see how treatment and time might affect outcomes.

Part C Results

Here, I changed our ECM algorithm into a full EM algorithm by updating both β and Σ fully at each step before moving to the next iteration. I tested this on the same simulated dataset:

- **Iterations:** The Q-function went from about -2976.1468 to -287.5468 across **31 outer iterations**.
- **Final Estimates:**

$$\hat{\Sigma} = \begin{pmatrix} 3.7669305 & 2.9530130 & -0.9207553 \\ 2.9530129 & 2.7803850 & -2.1797457 \\ -0.9207553 & -2.1797460 & 4.8424833 \end{pmatrix}.$$

The table below shows the **final coefficient estimates**, their **standard errors**, and the **95% confidence intervals**:

Parameter	Estimate	Std. Error	95% CI Lower	95% CI Upper
β_1	0.08399159	0.001794170	0.08047502	0.08750817
β_2	1.18787026	0.001806340	1.18432983	1.19141069
β_3	-0.82723238	0.001809628	-0.83077925	-0.82368551
β_4	-0.30565888	0.001780572	-0.30914880	-0.30216896
β_5	-0.91849752	0.001825956	-0.92207639	-0.91491864

After 31 outer iterations, the EM algorithm's parameter estimates are nearly the same as those from ECM. The standard errors and confidence intervals also remain very narrow, showing that both ECM and EM produce precise results for this simulated data.

Appendix

Problem 1

Part A

```
set.seed(032025)
n <- 1000
p <- 5
K <- 3
betaTrue <- rnorm(p)
L <- matrix(rnorm(K * K), K, K)
sigmaTrue <- L %*% t(L)
phi <- rnorm(p)
xList <- vector("list", n)
yList <- vector("list", n)
dropout <- integer(n)
for(i in seq_len(n)) {
  xList[[i]] <- matrix(rnorm(p * K), nrow = K, ncol = p)
  yFull <- xList[[i]] %*% betaTrue + MASS::mvrnorm(mu = rep(0, K), Sigma
    = sigmaTrue)
  yFull <- as.numeric(yFull)
  logitPM <- xList[[i]] %*% phi + c(-2, -1, 1)
  pM <- exp(logitPM) / sum(exp(logitPM))
  d_i <- rmultinom(1, 1, pM)
  dropout[i] <- which(d_i == 1)
  ySim <- yFull
  if(dropout[i] < K) ySim[(dropout[i] + 1):K] <- NA
  yList[[i]] <- ySim
}

conditionalMoments <- function(yObs, xObs, xFull, beta, sigma) {
  muFull <- xFull %*% beta
  obsIdx <- which(!is.na(yObs))
  misIdx <- which(is.na(yObs))
  muObs <- muFull[obsIdx]
  muMis <- muFull[misIdx]
  sigmaObsObs <- sigma[obsIdx, obsIdx, drop = FALSE]
  sigmaObsMis <- sigma[obsIdx, misIdx, drop = FALSE]
  sigmaMisObs <- sigma[misIdx, obsIdx, drop = FALSE]
  sigmaMisMis <- sigma[misIdx, misIdx, drop = FALSE]
  if(length(misIdx) == 0) {
```

```r
    eYFull <- as.vector(yObs)
    eYYFull <- as.vector(yObs) %o% as.vector(yObs)
  } else {
    misMean <- as.vector(muMis + sigmaMisObs %*% solve(sigmaObsObs) %*%
    (yObs[obsIdx] - muObs))
    eYFull <- numeric(length(muFull))
    eYFull[obsIdx] <- yObs[obsIdx]
    eYFull[misIdx] <- misMean
    misCov <- sigmaMisMis - sigmaMisObs %*% solve(sigmaObsObs) %*%
    sigmaObsMis
    eMisOuter <- misCov + (misMean %o% misMean)
    eYYFull <- matrix(0, nrow = length(muFull), ncol = length(muFull))
    eYYFull[obsIdx, obsIdx] <- as.vector(yObs[obsIdx]) %o%
    as.vector(yObs[obsIdx])
    eYYFull[obsIdx, misIdx] <- as.vector(yObs[obsIdx]) %o% misMean
    eYYFull[misIdx, obsIdx] <- misMean %o% as.vector(yObs[obsIdx])
    eYYFull[misIdx, misIdx] <- eMisOuter
  }
  list(eYFull = eYFull, eYYFull = eYYFull)
}

ECMFit <- function(xList, yList, p, K, tol = 1e-5, maxiter = 100,
    verbose = TRUE) {
  n <- length(xList)
  betaHat <- rep(0, p)
  sigmaHat <- diag(1, K)
  qOld <- -Inf
  iter <- 0
  repeat {
    iter <- iter + 1
    eY <- vector("list", n)
    eYY <- vector("list", n)
    for(i in seq_len(n)) {
      yObs <- yList[[i]]
      xObs <- xList[[i]][seq_along(yObs), , drop = FALSE]
      xFull <- xList[[i]]
      out <- conditionalMoments(yObs, xObs, xFull, betaHat, sigmaHat)
      eY[[i]] <- out$eYFull
      eYY[[i]] <- out$eYYFull
    }
    A <- matrix(0, p, p)
    B <- numeric(p)
    for(i in seq_len(n)) {
```

```
      Xi <- xList[[i]]
      A <- A + t(Xi) %*% solve(sigmaHat) %*% Xi
      B <- B + t(Xi) %*% solve(sigmaHat) %*% eY[[i]]
    }
    betaNew <- solve(A, B)
    S <- matrix(0, K, K)
    for(i in seq_len(n)) {
      Xi <- xList[[i]]
      yBar <- as.vector(eY[[i]])
      xb <- as.vector(Xi %*% betaNew)
      EYYTerm <- eYY[[i]]
      crossTerm <- yBar %*% t(xb) + xb %*% t(yBar)
      newResi <- EYYTerm - crossTerm + xb %*% t(xb)
      S <- S + newResi
    }
    sigmaNew <- S / n
    invSigmaNew <- solve(sigmaNew)
    logdetSig <- determinant(sigmaNew, logarithm = TRUE)$modulus
    qNew <- -0.5 * n * logdetSig
    accum <- 0
    for(i in seq_len(n)) {
      Xi <- xList[[i]]
      ybari <- eY[[i]] - Xi %*% betaNew
      ERes <- eYY[[i]] - eY[[i]] %*% t(Xi %*% betaNew) - (Xi %*%
    betaNew) %*% t(eY[[i]]) + (Xi %*% betaNew) %*% t(Xi %*% betaNew)
      accum <- accum + sum(diag(ERes %*% invSigmaNew))
    }
    qNew <- qNew - 0.5 * accum
    if(verbose) cat(sprintf("Iteration %d: Q=%.4f, diff=%.6f\n", iter,
        qNew, qNew - qOld))
    if((qNew - qOld) < tol || iter >= maxiter) break
    betaHat <- betaNew
    sigmaHat <- sigmaNew
    qOld <- qNew
  }
  list(beta = betaHat, sigma = sigmaHat, iter = iter)
}

computeObservedInfoBeta <- function(xList, yList, betaHat, sigmaHat) {
  n <- length(xList)
  I_beta <- matrix(0, nrow = length(betaHat), ncol = length(betaHat))
  for(i in seq_len(n)) {
    Xi <- xList[[i]]
```

```
    yObs <- yList[[i]]
    xObs <- Xi[seq_along(yObs), , drop = FALSE]
    out <- conditionalMoments(yObs, xObs, Xi, betaHat, sigmaHat)
    eY <- out$eYFull
    eYY <- out$eYYFull
    xb <- as.vector(Xi %*% betaHat)
    R_i <- eYY - (eY %*% t(xb)) - (xb %*% t(eY)) + (xb %*% t(xb))
    I_i <- t(Xi) %*% solve(sigmaHat) %*% R_i %*% solve(sigmaHat) %*% Xi
    I_beta <- I_beta + I_i
  }
  return(I_beta)
}

fitEcm <- ECMFit(xList, yList, p, K)
```

```
Iteration 1: Q=-2980.9778, diff=Inf
Iteration 2: Q=-2491.4732, diff=489.504637
Iteration 3: Q=-2083.7687, diff=407.704476
Iteration 4: Q=-1730.8239, diff=352.944777
Iteration 5: Q=-1392.5059, diff=338.318015
Iteration 6: Q=-1074.1472, diff=318.358726
Iteration 7: Q=-798.0400, diff=276.107131
Iteration 8: Q=-586.4890, diff=211.551031
Iteration 9: Q=-446.8267, diff=139.662340
Iteration 10: Q=-366.6851, diff=80.141514
Iteration 11: Q=-325.2214, diff=41.463768
Iteration 12: Q=-305.0757, diff=20.145675
Iteration 13: Q=-295.6101, diff=9.465590
Iteration 14: Q=-291.2358, diff=4.374336
Iteration 15: Q=-289.2302, diff=2.005581
Iteration 16: Q=-288.3141, diff=0.916125
Iteration 17: Q=-287.8963, diff=0.417748
Iteration 18: Q=-287.7060, diff=0.190335
Iteration 19: Q=-287.6193, diff=0.086687
Iteration 20: Q=-287.5798, diff=0.039472
Iteration 21: Q=-287.5619, diff=0.017971
Iteration 22: Q=-287.5537, diff=0.008181
Iteration 23: Q=-287.5500, diff=0.003724
Iteration 24: Q=-287.5483, diff=0.001695
Iteration 25: Q=-287.5475, diff=0.000771
Iteration 26: Q=-287.5471, diff=0.000351
Iteration 27: Q=-287.5470, diff=0.000160
Iteration 28: Q=-287.5469, diff=0.000073
```

```
Iteration 29: Q=-287.5469, diff=0.000033
Iteration 30: Q=-287.5469, diff=0.000015
Iteration 31: Q=-287.5468, diff=0.000007
```

```
fitEcm$beta
```

```
           [,1]
[1,]  0.0839916
[2,]  1.1878703
[3,] -0.8272324
[4,] -0.3056589
[5,] -0.9184975
```

```
fitEcm$sigma
```

```
           [,1]      [,2]       [,3]
[1,]  3.7669305  2.953013 -0.9207553
[2,]  2.9530129  2.780385 -2.1797457
[3,] -0.9207553 -2.179746  4.8424833
```

```
fitEcm$iter
```

```
[1] 31
```

```
I_beta <- computeObservedInfoBeta(xList, yList, fitEcm$beta,
    fitEcm$sigma)
varBeta <- solve(I_beta)
seBeta <- sqrt(diag(varBeta))

ciLower <- fitEcm$beta - 1.96 * seBeta
ciUpper <- fitEcm$beta + 1.96 * seBeta

list(standardErrors = seBeta, ciLower = ciLower, ciUpper = ciUpper)
```

```
$standardErrors
[1] 0.001794170 0.001806340 0.001809628 0.001780572 0.001825956

$ciLower
```

```
           [,1]
[1,]  0.08047503
[2,]  1.18432983
[3,] -0.83077925
[4,] -0.30914881
[5,] -0.92207639

$ciUpper
           [,1]
[1,]  0.08750817
[2,]  1.19141068
[3,] -0.82368551
[4,] -0.30216897
[5,] -0.91491865
```

Part B

```r
library(Surrogate)
library(dplyr)
data("Schizo_PANSS")
hwData <- Schizo_PANSS[, c("Id", "Treat", "Week1", "Week4", "Week8")]
hwData <- subset(hwData, (!is.na(Week1) & !is.na(Week4) & !is.na(Week8))
    |
                     (!is.na(Week1) & !is.na(Week4) & is.na(Week8)) |
                     (!is.na(Week1) & is.na(Week4) & is.na(Week8)))
ids <- unique(hwData$Id)
xListReal <- list()
yListReal <- list()
times <- c(1, 4, 8)
for(i in seq_along(ids)) {
  tmp <- hwData[hwData$Id == ids[i], ]
  xMat <- matrix(NA, nrow = length(times), ncol = 4)
  yVec <- rep(NA, length(times))
  for(j in seq_along(times)) {
    tt <- times[j]
    varName <- paste0("Week", tt)
    xMat[j, ] <- c(1, tmp$Treat[1], tt, tmp$Treat[1] * tt)
    yVec[j] <- tmp[[varName]][1]
  }
  xListReal[[length(xListReal) + 1]] <- xMat
  yListReal[[length(yListReal) + 1]] <- yVec
}


fitEcmReal <- ECMFit(xListReal, yListReal, p = 4, K = 3)
```

```
Iteration 1: Q=-20231.4826, diff=Inf
Iteration 2: Q=-20081.7622, diff=149.720354
Iteration 3: Q=-19972.5625, diff=109.199670
Iteration 4: Q=-19925.0304, diff=47.532182
Iteration 5: Q=-19905.2298, diff=19.800542
Iteration 6: Q=-19896.7880, diff=8.441831
Iteration 7: Q=-19893.0446, diff=3.743372
Iteration 8: Q=-19891.3113, diff=1.733353
Iteration 9: Q=-19890.4740, diff=0.837214
Iteration 10: Q=-19890.0540, diff=0.420047
Iteration 11: Q=-19889.8364, diff=0.217614
Iteration 12: Q=-19889.7207, diff=0.115657
```

```
Iteration 13: Q=-19889.6581, diff=0.062672
Iteration 14: Q=-19889.6236, diff=0.034445
Iteration 15: Q=-19889.6045, diff=0.019121
Iteration 16: Q=-19889.5938, diff=0.010688
Iteration 17: Q=-19889.5878, diff=0.006002
Iteration 18: Q=-19889.5844, diff=0.003381
Iteration 19: Q=-19889.5825, diff=0.001908
Iteration 20: Q=-19889.5814, diff=0.001078
Iteration 21: Q=-19889.5808, diff=0.000610
Iteration 22: Q=-19889.5805, diff=0.000345
Iteration 23: Q=-19889.5803, diff=0.000195
Iteration 24: Q=-19889.5802, diff=0.000111
Iteration 25: Q=-19889.5801, diff=0.000063
Iteration 26: Q=-19889.5801, diff=0.000035
Iteration 27: Q=-19889.5801, diff=0.000020
Iteration 28: Q=-19889.5800, diff=0.000011
Iteration 29: Q=-19889.5800, diff=0.000006
```

```r
fitEcmReal$beta
```

```
            [,1]
[1,] -5.21513003
[2,] -0.86183388
[3,] -1.45963556
[4,] -0.02500182
```

```r
fitEcmReal$sigma
```

```
         [,1]     [,2]     [,3]
[1,] 194.5852 163.1883 160.0425
[2,] 163.1883 418.2026 374.7859
[3,] 160.0425 374.7859 496.2113
```

```r
fitEcmReal$iter
```

```
[1] 29
```

```r
I_beta_real <- computeObservedInfoBeta(xListReal, yListReal,
    fitEcmReal$beta, fitEcmReal$sigma)
```

```r
varBeta_real <- solve(I_beta_real)
seBeta_real <- sqrt(diag(varBeta_real))

ciLower_real <- fitEcmReal$beta - 1.96 * seBeta_real
ciUpper_real <- fitEcmReal$beta + 1.96 * seBeta_real
list(standardErrors = seBeta_real, ciLower = ciLower_real, ciUpper =
    ciUpper_real)
```

```
$standardErrors
[1] 0.37056342 0.37056342 0.06266164 0.06266164

$ciLower
           [,1]
[1,] -5.9414343
[2,] -1.5881382
[3,] -1.5824524
[4,] -0.1478186

$ciUpper
            [,1]
[1,] -4.48882572
[2,] -0.13552957
[3,] -1.33681875
[4,]  0.09781499
```

```r
EMFit <- function(xList, yList, p, K, tol = 1e-5, maxiter = 100, verbose
    = TRUE) {
  n <- length(xList)
  betaHat <- rep(0, p)
  sigmaHat <- diag(1, K)
  qOld <- -Inf
  iter <- 0
  repeat {
    iter <- iter + 1
    eY <- vector("list", n)
    eYY <- vector("list", n)
    for(i in seq_len(n)) {
      yObs <- yList[[i]]
      xObs <- xList[[i]][seq_along(yObs), , drop = FALSE]
      xFull <- xList[[i]]
      out <- conditionalMoments(yObs, xObs, xFull, betaHat, sigmaHat)
      eY[[i]] <- out$eYFull
      eYY[[i]] <- out$eYYFull
    }
    betaInner <- betaHat
    sigmaInner <- sigmaHat
    repeat {
      A <- matrix(0, p, p)
      B <- numeric(p)
      for(i in seq_len(n)) {
        Xi <- xList[[i]]
        A <- A + t(Xi) %*% solve(sigmaInner) %*% Xi
        B <- B + t(Xi) %*% solve(sigmaInner) %*% eY[[i]]
      }
      betaNew <- solve(A, B)
      S <- matrix(0, K, K)
      for(i in seq_len(n)) {
        Xi <- xList[[i]]
        yBar <- as.vector(eY[[i]])
        xb <- as.vector(Xi %*% betaNew)
        EYYTerm <- eYY[[i]]
        crossTerm <- yBar %*% t(xb) + xb %*% t(yBar)
        newResi <- EYYTerm - crossTerm + xb %*% t(xb)
        S <- S + newResi
      }
      sigmaNew <- S / n
```

```
      if(sqrt(sum((betaNew - betaInner)^2)) < 1e-8 &&
         sqrt(sum((sigmaNew - sigmaInner)^2)) < 1e-8) break
      betaInner <- betaNew
      sigmaInner <- sigmaNew
    }
    invSigmaNew <- solve(sigmaNew)
    logdetSig <- determinant(sigmaNew, logarithm = TRUE)$modulus
    qNew <- -0.5 * n * logdetSig
    accum <- 0
    for(i in seq_len(n)) {
      Xi <- xList[[i]]
      ybari <- eY[[i]] - Xi %*% betaNew
      ERes <- eYY[[i]] - eY[[i]] %*% t(Xi %*% betaNew) - (Xi %*%
    betaNew) %*% t(eY[[i]]) + (Xi %*% betaNew) %*% t(Xi %*% betaNew)
      accum <- accum + sum(diag(ERes %*% invSigmaNew))
    }
    qNew <- qNew - 0.5 * accum
    if(verbose) cat(sprintf("Iteration %d: Q=%.4f, diff=%.6f\n", iter,
        qNew, qNew - qOld))
    if((qNew - qOld) < tol || iter >= maxiter) break
    betaHat <- betaNew
    sigmaHat <- sigmaNew
    qOld <- qNew
  }
  list(beta = betaHat, sigma = sigmaHat, iter = iter)
}

fitEm <- EMFit(xList, yList, p, K)
```

```
Iteration 1: Q=-2976.1468, diff=Inf
Iteration 2: Q=-2474.9909, diff=501.155941
Iteration 3: Q=-2068.4368, diff=406.554097
Iteration 4: Q=-1715.1819, diff=353.254843
Iteration 5: Q=-1376.2445, diff=338.937423
Iteration 6: Q=-1058.5003, diff=317.744209
Iteration 7: Q=-784.7490, diff=273.751330
Iteration 8: Q=-576.8437, diff=207.905247
Iteration 9: Q=-440.8681, diff=135.975664
Iteration 10: Q=-363.4477, diff=77.420354
Iteration 11: Q=-323.6012, diff=39.846543
Iteration 12: Q=-304.3006, diff=19.300585
Iteration 13: Q=-295.2476, diff=9.053027
Iteration 14: Q=-291.0680, diff=4.179557
```

```
Iteration 15: Q=-289.1529, diff=1.915064
Iteration 16: Q=-288.2786, diff=0.874364
Iteration 17: Q=-287.8800, diff=0.398545
Iteration 18: Q=-287.6985, diff=0.181518
Iteration 19: Q=-287.6159, diff=0.082641
Iteration 20: Q=-287.5783, diff=0.037617
Iteration 21: Q=-287.5611, diff=0.017121
Iteration 22: Q=-287.5533, diff=0.007792
Iteration 23: Q=-287.5498, diff=0.003546
Iteration 24: Q=-287.5482, diff=0.001613
Iteration 25: Q=-287.5475, diff=0.000734
Iteration 26: Q=-287.5471, diff=0.000334
Iteration 27: Q=-287.5470, diff=0.000152
Iteration 28: Q=-287.5469, diff=0.000069
Iteration 29: Q=-287.5469, diff=0.000031
Iteration 30: Q=-287.5469, diff=0.000014
Iteration 31: Q=-287.5468, diff=0.000006
```

```r
fitEm$beta
```

```
            [,1]
[1,]  0.08399159
[2,]  1.18787026
[3,] -0.82723238
[4,] -0.30565888
[5,] -0.91849752
```

```r
fitEm$sigma
```

```
           [,1]      [,2]       [,3]
[1,]  3.7669305  2.953013 -0.9207553
[2,]  2.9530129  2.780385 -2.1797457
[3,] -0.9207553 -2.179746  4.8424833
```

```r
fitEm$iter
```

```
[1] 31
```

```
emSteps <- fitEm$iter
ecmSteps <- fitEcm$iter
emSteps
```

```
[1] 31
```

```
ecmSteps
```

```
[1] 31
```

```
I_beta_em <- computeObservedInfoBeta(xList, yList, fitEm$beta,
    fitEm$sigma)
varBeta_em <- solve(I_beta_em)
seBeta_em <- sqrt(diag(varBeta_em))

ciLower_em <- fitEm$beta - 1.96 * seBeta_em
ciUpper_em <- fitEm$beta + 1.96 * seBeta_em
list(standardErrors = seBeta_em, ciLower = ciLower_em, ciUpper =
    ciUpper_em)
```

```
$standardErrors
[1] 0.001794170 0.001806340 0.001809628 0.001780572 0.001825956

$ciLower
            [,1]
[1,]  0.08047502
[2,]  1.18432983
[3,] -0.83077925
[4,] -0.30914880
[5,] -0.92207639

$ciUpper
            [,1]
[1,]  0.08750817
[2,]  1.19141069
[3,] -0.82368551
[4,] -0.30216896
[5,] -0.91491864
```