Homework 1 | Missingness Patterns

Brian Cervantes Alvarez
January 28, 2025

Table of contents

Question 1.1

The majority of participants (62.44%) have no missing data, while other patterns, such as missing only Week 8, occur at lower frequencies. A total of eight distinct missingness patterns were observed, with proportions ranging from 0.46% to 62.44%.

Question 1.2

The treatment group is significantly less likely to have missing data ($p < 0.001$), suggesting a protective effect against missingness, with an estimated decrease in the log odds of missingness by 0.38 compared to the control group. This result suggests that treatment specifically reduces the likelihood of patterns involving missing data in Week 8 or other combinations.

Question 1.3

Out of 2,151 patients, 45 dropped out after Week 1, and 799 experienced intermittent missingness. Among those who dropped out, higher Week 1 PANSS scores were significantly associated with an increased likelihood of dropout ($p < 0.001$), suggesting that patients with more severe symptoms at baseline were more likely to discontinue. In contrast, Week 1 PANSS scores did not significantly predict intermittent missingness ($p = 0.155$).

Question 1.4

The visualization highlights that most participants have complete data for all three weeks, but systematic patterns of missingness, such as missing only Week 8, suggest that missingness may not be MCAR. Instead, MAR is more plausible since missingness could depend on observed Week 1 or Week 4 data.

Question 1.5

The algorithm iteratively estimates the coefficients (β) and the covariance matrix (Σ) using the following steps:

1. **Coefficient Update**:

$$\beta^{(t+1)} = \left(\sum_i X_i^T (\Sigma^{(t)})^{-1} X_i \right)^{-1} \sum_i X_i^T (\Sigma^{(t)})^{-1} y_i$$

   Here, X_i is the design matrix for participant i, and y_i is their response vector.

2. **Covariance Matrix Update**:

$$\Sigma^{(t+1)} = \frac{1}{n} \sum_i (y_i - X_i \beta^{(t)})(y_i - X_i \beta^{(t)})^T$$

   The covariance matrix measures the variability in the residuals after accounting for the effects of predictors.

3. **Convergence**: Repeat these steps until changes in β and Σ are negligible.

The algorithm estimates coefficients (β) and the covariance matrix (Σ) for the linear model:

$$\mathbf{y}_i | \text{Treat}_i \sim \text{Normal}(\mu + \beta_1 \text{Treat}_i + \beta_2 t + \beta_3 \text{Treat}_i \cdot t, \Sigma)$$

The model indicates that PANSS scores decrease over time ($\beta_{\text{time}} = -1.69$), with treatment having a minimal effect ($\beta_{\text{Treat:time}} = -0.09$). The MLE for μ is -7.72, and the covariance matrix Σ suggests residual variability of 297.69.

Question 1.6

Adding random effects accounts for patient-specific variability, with significant fixed effects for time ($p < 0.001$), but weak interaction effects ($\beta_{\text{Treat:time}} = -0.09$, $p = 0.22$). In terms of future research, we could look at:

- Adding baseline PANSS scores as covariates to adjust for initial variability in symptoms.
- Including demographic variables, such as age or gender, as interaction terms with time or treatment.
- Examining potential clustering effects, such as grouping by study site, to refine the model further.

Question 2.1

The $\hat{R}$ values are all close to 1, indicating good convergence and reliable posterior sampling.

Parameter	Rhat
beta0	1.004168
beta1	1.003996
beta2	1.003540
beta3	1.002623

To test if our Bayesian algorithm is correctly implemented, use **simulation-based calibration (SBC)** as described by Talts et al. (2018). Simulate datasets from a known generative model with true parameters (θ_{true}), fit our model, and compute the rank of θ_{true} among posterior samples. Repeat this for many simulations to accumulate rank statistics.

If the algorithm is correct, the ranks should follow a uniform distribution. Alternatively, assess posterior quantiles by computing the posterior CDF at θ_{true} and check if the values are uniformly distributed. Lastly, we can visualize results using histograms or CDF plots and use tests like Kolmogorov-Smirnov for formal checks. This ensures the posterior aligns with the data-generating process!

Question 2.2

The Bayesian model achieved good convergence, providing reliable posterior estimates for all parameters.

Results:

Posterior Quantiles for Beta Parameters

Parameter	2.5%	50%	97.5%
beta_0	-6.6962	-5.8768	-5.0918
beta_1	-1.8022	-0.9962	-0.2186
beta_2	-1.7426	-1.5657	-1.3932
beta_3	-0.2152	-0.0497	0.1109

The 95% posterior intervals for the β parameters estimate the effects of the intercept, treatment, time, and their interaction.

Posterior Quantiles for Correlation Parameters

Parameter	2.5%	50%	97.5%
cor21	0.5488	0.5857	0.6221
cor31	0.4648	0.5093	0.5481
cor32	0.7612	0.7844	0.8054

The 95% posterior intervals for the correlation parameters indicate strong positive correlations between the residuals of different measurement occasions.

Appendix

Question 1.1 Code

```
library(tidyverse)
library(mice)
library(lme4)
library(Surrogate)
library(MCMCpack)
library(posterior)
library(tidyverse)

data("Schizo_PANSS")
schizoData <- Schizo_PANSS %>%
  select(Id, Treat, Week1, Week4, Week8)
missingnessSummary <- schizoData %>%
  select(Week1, Week4, Week8) %>%
  mutate_all(is.na) %>%
  unite("pattern", Week1:Week8, sep = ", ") %>%
  count(pattern) %>%
  mutate(proportion = n / sum(n))
missingnessSummary
```

Question 1.2 Code

```
schizoData <- schizoData %>%
  mutate(missingness = if_any(c(Week1, Week4, Week8), is.na))
treatmentMissingnessModel <- glm(missingness ~ Treat,
  data = schizoData, family = "binomial")
summary(treatmentMissingnessModel)
```

Question 1.3 Code

```
schizoData <- schizoData %>%
  mutate(
    dropout = is.na(Week4) & is.na(Week8),
    intermittent = !dropout & if_any(c(Week1, Week4, Week8), is.na)
  )
dropoutModel <- glm(dropout ~ Week1,
  data = schizoData, family = "binomial")
summary(dropoutModel)
intermittentModel <- glm(intermittent ~ Week1,
  data = schizoData, family = "binomial")
summary(intermittentModel)
```

Question 1.4 Code

```
md.pattern(schizoData %>% select(Week1, Week4, Week8))
```

Question 1.5 Code

```r
completeCases <- schizoData %>%
  filter(complete.cases(Week1, Week4, Week8))
longFormatData <- completeCases %>%
  pivot_longer(cols = c(Week1, Week4, Week8),
    names_to = "time", values_to = "panss") %>%
  mutate(time = as.numeric(str_replace(time, "Week", "")))
designMatrix <- model.matrix(~ Treat * time, data = longFormatData)
responseVector <- longFormatData$panss
n <- nrow(longFormatData)
estimatedBeta <- solve(t(designMatrix) %*% designMatrix) %*%
    t(designMatrix) %*% responseVector
estimatedSigma <- 1 / n * t(responseVector - designMatrix %*%
    estimatedBeta) %*% (responseVector - designMatrix %*% estimatedBeta)
list(beta = estimatedBeta, Sigma = estimatedSigma)
```

Question 1.6 Code

```
expandedModel <- lmer(panss ~ Treat * time + (1 | Id),
  data = longFormatData)
summary(expandedModel)
```

Question 2.1 Code

```r
set.seed(2392)

# Simulated data setup
nSubjects <- 200
timePoints <- 3
pDim <- 4

# True beta
betaTrue <- c(5, -2, -1, -0.5)

# Design matrix setup
subjectIds <- rep(1:nSubjects, each = timePoints)
timeVals <- rep(c(1, 4, 8), times = nSubjects)
treatVec <- rbinom(nSubjects, 1, 0.5)
treatBig <- rep(treatVec, each = timePoints)
xBig <- cbind(1, treatBig, timeVals, treatBig * timeVals)

# True covariance and response simulation
sigmaTrue <- matrix(c(10, 2, 1, 2, 10, 2, 1, 2, 10), nrow=3)
yList <- vector("list", nSubjects)
for (i in 1:nSubjects) {
  idx <- ((i - 1) * timePoints + 1):(i * timePoints)
  meanI <- xBig[idx, ] %*% betaTrue
  yList[[i]] <- MASS::mvrnorm(1, mu = meanI, Sigma = sigmaTrue)
}
yBig <- unlist(yList)

# Bayesian sampling parameters
nChains <- 4
nIter <- 500
mu0 <- rep(0, pDim)
sigma0 <- diag(pDim)
nu0 <- 3
v0 <- diag(timePoints)

betaDraws <- array(NA, dim = c(nIter, pDim, nChains))
sigmaDraws <- array(NA, dim = c(nIter, timePoints, timePoints, nChains))

xTx <- matrix(0, nrow = pDim, ncol = pDim)
xTy <- rep(0, pDim)
```

```r
invSigma0 <- solve(sigma0)

ySplit <- split(yBig, subjectIds)
xSplit <- split.data.frame(as.data.frame(xBig), subjectIds)

posteriorBetaMean <- function(sigmaNow) {
  sigmaInv <- solve(sigmaNow)
  sumXtSigmaInvX <- matrix(0, pDim, pDim)
  sumXtSigmaInvY <- rep(0, pDim)
  for (i in seq_len(nSubjects)) {
    xi <- as.matrix(xSplit[[i]])
    yi <- ySplit[[i]]
    sumXtSigmaInvX <- sumXtSigmaInvX + t(xi) %*% sigmaInv %*% xi
    sumXtSigmaInvY <- sumXtSigmaInvY + t(xi) %*% sigmaInv %*% yi
  }
  sumXtSigmaInvXPost <- sumXtSigmaInvX + invSigma0
  sumXtSigmaInvYPost <- sumXtSigmaInvY + invSigma0 %*% mu0
  vBeta <- solve(sumXtSigmaInvXPost)
  mBeta <- vBeta %*% sumXtSigmaInvYPost
  list(mean = mBeta, sigma = vBeta)
}

# Gibbs sampling
for (b in 1:nChains) {
  betaNow <- MASS::mvrnorm(1, mu0, sigma0)
  sigmaNow <- riwish(nu0, v0)
  for (s in 1:nIter) {
    postInfo <- posteriorBetaMean(sigmaNow)
    betaNow <- MASS::mvrnorm(1, postInfo$mean, Sigma = postInfo$sigma)
    sumRes <- matrix(0, timePoints, timePoints)
    for (i in seq_len(nSubjects)) {
      xi <- as.matrix(xSplit[[i]])
      yi <- ySplit[[i]]
      resI <- yi - xi %*% betaNow
      sumRes <- sumRes + tcrossprod(resI)
    }
    nuPost <- nSubjects + nu0
    vPost <- v0 + sumRes
    sigmaNow <- riwish(v = nuPost, S = vPost)
    betaDraws[s, , b] <- betaNow
    sigmaDraws[s, , , b] <- sigmaNow
  }
```

```
}
# Convergence check
burnIn <- nIter / 2
keepIter <- (burnIn + 1):nIter

rhatVals <- numeric(pDim)
paramNames <- c("beta0", "beta1", "beta2", "beta3")

for (j in 1:pDim) {
  drawsMatrix <- sapply(1:nChains, function(bc) betaDraws[keepIter, j,
    bc])
  rhatVals[j] <- rhat(drawsMatrix)
}

rhatDf <- data.frame(
  parameter = paramNames,
  rhat = rhatVals
)
rhatDf
```

Question 2.2 Code

```
# Filter complete cases and reshape to long format
compCase <- schizoData %>%
  filter(complete.cases(Week1, Week4, Week8))

longComp <- compCase %>%
  pivot_longer(cols = starts_with("Week"),
               names_to = "time",
               values_to = "panss") %>%
  mutate(time = as.numeric(gsub("Week", "", time))) %>%
  arrange(Id, time)

nComp <- length(unique(longComp$Id))
timePoints <- 3
xMatComp <- model.matrix(~ Treat * time, data = longComp)
yComp <- longComp$panss
subjectIdsComp <- longComp$Id

pDim <- ncol(xMatComp)
xSplitComp <- split.data.frame(as.data.frame(xMatComp), subjectIdsComp)
ySplitComp <- split(yComp, subjectIdsComp)

# Priors and sampling setup
mu0 <- rep(0, pDim)
sigma0 <- diag(pDim)
nu0 <- 3
v0 <- diag(timePoints)
nChains <- 4
nIter <- 500
betaDrawsComp <- array(NA, dim = c(nIter, pDim, nChains))
sigmaDrawsComp <- array(NA, dim = c(nIter, timePoints, timePoints,
    nChains))

# Posterior beta mean function
posteriorBetaMeanComp <- function(sigmaNow) {
  sigmaInv <- solve(sigmaNow)
  sumXtSigmaInvX <- matrix(0, pDim, pDim)
  sumXtSigmaInvY <- rep(0, pDim)
  for (i in seq_len(nComp)) {
    xi <- as.matrix(xSplitComp[[i]])
    yi <- ySplitComp[[i]]
```

```
    sumXtSigmaInvX <- sumXtSigmaInvX + t(xi) %*% sigmaInv %*% xi
    sumXtSigmaInvY <- sumXtSigmaInvY + t(xi) %*% sigmaInv %*% yi
  }
  sumXtSigmaInvXPost <- sumXtSigmaInvX + solve(sigma0)
  sumXtSigmaInvYPost <- sumXtSigmaInvY + solve(sigma0) %*% mu0
  vBeta <- solve(sumXtSigmaInvXPost)
  mBeta <- vBeta %*% sumXtSigmaInvYPost
  list(mean = mBeta, sigma = vBeta)
}

# Gibbs sampling
set.seed(2392)
for (b in 1:nChains) {
  betaNow <- MASS::mvrnorm(1, mu0, sigma0)
  sigmaNow <- riwish(nu0, v0)
  for (s in 1:nIter) {
    postInfo <- posteriorBetaMeanComp(sigmaNow)
    betaNow <- MASS::mvrnorm(1, postInfo$mean, Sigma = postInfo$sigma)
    sumRes <- matrix(0, timePoints, timePoints)
    for (i in seq_len(nComp)) {
      xi <- as.matrix(xSplitComp[[i]])
      yi <- ySplitComp[[i]]
      resI <- yi - xi %*% betaNow
      sumRes <- sumRes + tcrossprod(resI)
    }
    nuPost <- nComp + nu0
    vPost <- v0 + sumRes
    sigmaNow <- riwish(v = nuPost, S = vPost)
    betaDrawsComp[s, , b] <- betaNow
    sigmaDrawsComp[s, , , b] <- sigmaNow
  }
}

# Discard burn-in and compute posterior summaries
burnIn <- nIter / 2
keepIter <- (burnIn + 1):nIter

# Beta posterior quantiles in long format
betaPostAll <- betaDrawsComp[keepIter, , ]
betaPostVec <- as.data.frame(
  do.call(rbind, lapply(1:nChains, function(ch) betaPostAll[, , ch]))
)
```

```r
colnames(betaPostVec) <- paste0("beta_", 0:(pDim-1))  # Adjusted to
    start from beta_0

betaSummaryLong <- betaPostVec %>%
  pivot_longer(
    cols = everything(),
    names_to = "parameter",
    values_to = "value"
  ) %>%
  group_by(parameter) %>%
  summarize(
    q2.5 = quantile(value, 0.025, na.rm = TRUE),
    q50 = quantile(value, 0.50, na.rm = TRUE),
    q97.5 = quantile(value, 0.975, na.rm = TRUE)
  ) %>%
  arrange(parameter)

# Correlation parameters
extractCorrelations <- function(sig) {
  c(
    cor21 = sig[2, 1] / sqrt(sig[1, 1] * sig[2, 2]),
    cor31 = sig[3, 1] / sqrt(sig[1, 1] * sig[3, 3]),
    cor32 = sig[3, 2] / sqrt(sig[2, 2] * sig[3, 3])
  )
}

corDraws <- NULL
for (b in 1:nChains) {
  for (s in keepIter) {
    sigNow <- sigmaDrawsComp[s, , , b]
    corVals <- extractCorrelations(sigNow)
    corDraws <- rbind(corDraws, corVals)
  }
}
corDrawsDf <- as.data.frame(corDraws)
colnames(corDrawsDf) <- c("cor21", "cor31", "cor32")

corSummaryLong <- corDrawsDf %>%
  pivot_longer(
    cols = everything(),
    names_to = "parameter",
    values_to = "value"
```

```
  ) %>%
  group_by(parameter) %>%
  summarize(
    q2.5 = quantile(value, 0.025, na.rm = TRUE),
    q50 = quantile(value, 0.50, na.rm = TRUE),
    q97.5 = quantile(value, 0.975, na.rm = TRUE)
  ) %>%
  arrange(parameter)

# Display summaries
betaSummaryLong
corSummaryLong
```