University of Edinburgh, School of Mathematics
Incomplete Data Analysis, 2023/2024
Assessment—My Solution

Billy Ding

Question 1

Solution:

Suppose H is a random variable following a Pareto distribution with parameter $\gamma > 0$. Given that its CDF is $F_H(h;\gamma) = 1 - \frac{1}{h^\gamma}$, for $h > 1$, the PDF for H is

$$f_H(h;\gamma) = \frac{d}{dh}F_H(h;\gamma) = \frac{d}{dh}(1 - \frac{1}{h^\gamma}) = \gamma h^{-(\gamma+1)},\ \ h > 1.$$

(a) The following process includes deducing the probability density (mass) function of the two given random variables.

- Since $Z = \min\{X, Y\}$, we have

$$\begin{aligned}F_Z(z) &\equiv \mathbb{P}(Z \leq z)\\ &= 1 - \mathbb{P}(Z > z)\\ &= 1 - \mathbb{P}(X > z,\ Y > z)\\ &= 1 - \mathbb{P}(X > z)\mathbb{P}(Y > z) \quad \text{(since } X \text{ and } Y \text{ are independent)}\\ &= 1 - (1 - F_X(z))(1 - F_Y(z))\\ &= 1 - z^{-\lambda}z^{-\mu}\\ &= 1 - \frac{1}{z^{\lambda+\mu}}.\end{aligned}$$

  Thus, Z also follows a Pareto distribution with parameter $\lambda + \mu$, with density function

$$f_Z(z) = (\lambda + \mu)z^{-(\lambda+\mu+1)},\ \ z > 1.$$

- Since $\delta = \mathbb{1}_{\{X<Y\}}$, δ is a Bernoulli random variable with parameter $p \equiv \mathbb{P}(\delta = 1) = \mathbb{P}(X < Y)$, where

$$\begin{aligned}p \equiv \mathbb{P}(X < Y) &= \int_1^\infty \mathbb{P}(Y > X|X = a)f_X(a)da\\ &= \int_1^\infty (1 - F_Y(a))f_X(a)da\\ &= \int_1^\infty a^{-\mu}\lambda a^{-(\lambda+1)}\\ &= -\frac{\lambda}{\mu+\lambda}a^{-(\mu+\lambda)}\Bigg|_{a=1}^{a=\infty} \quad \text{(since } \mu + \lambda > 0\text{)}\\ &= \frac{\lambda}{\mu+\lambda}.\end{aligned}$$

Thus, the frequency function of $delta$ is

$$f_\delta(\delta) = \begin{cases} \frac{\lambda}{\mu+\lambda}, & \text{if } \delta = 1 \\ \frac{\mu}{\mu+\lambda}, & \text{if } \delta = 0 \end{cases}$$

(b) Next, we calculate the maximum likelihood estimators of the corresponding parameters.

- The likelihood function for θ is

$$L(\theta|\vec{z}) = \prod_{i=1}^n f_{Z_i}(z_i;\theta) = \prod_{i=1}^n \theta z_i^{-(\theta+1)}.$$

Thus, the corresponding log-likelihood function is

$$\log L(\theta|\vec{z}) = n\log(\theta) - (\theta+1)\sum_{i=1}^n \log(z_i).$$

Solving the first order condition (FOC), i.e. setting

$$\frac{\partial}{\partial\theta}\log L(\theta|\vec{z}) = \frac{\partial}{\partial\theta}(n\log(\theta) - (\theta+1)\sum_{i=1}^n \log(z_i)) = \frac{n}{\theta} - \sum_{i=1}^n \log(z_i) = 0,$$

we have the MLE of θ

$$\hat{\theta}_M = \frac{n}{\sum_{i=1}^n \log(z_i)} \quad (z_i > 1).$$

Note that

$$\frac{\partial^2}{\partial\theta^2}\log L(\theta|\vec{z}) = \frac{-n}{\theta^2} < 0.$$

Hence, $\hat{\theta}_M$ is indeed a global maxima of the likelihood function.

- The likelihood function for p is

$$L(p|\vec{\delta}) = \prod_{i=1}^n p^{\delta_i}(1-p)^{1-\delta_i}.$$

Thus, the corresponding log-likelihood function is

$$\begin{aligned}\log L(p|\vec{\delta}) &= \sum_{i=1}^n \delta_i\log(p) + (1-\delta_i)\log(1-p) \\ &= \log(p)\sum_{i=1}^n \delta_i + \log(1-p)\sum_{i=1}^n (1-\delta_i).\end{aligned}$$

Solving the first order condition (FOC), i.e. setting

$$\frac{\partial}{\partial p}\log L(p|\vec{\delta}) = \frac{\partial}{\partial p}(\log(p)\sum_{i=1}^n \delta_i + \log(1-p)\sum_{i=1}^n(1-\delta_i)) = \frac{\sum_{i=1}^n \delta_i}{p} - \frac{n-\sum_{i=1}^n \delta_i}{1-p} = 0,$$

we have the MLE of p

$$\hat{p}_M = \frac{\sum_{i=1}^n \delta_i}{n}.$$

Note that

$$\frac{\partial^2}{\partial p^2}\log L(p|\vec{\delta}) = -\frac{\sum_{i=1}^n \delta_i}{p^2} - \frac{n-\sum_{i=1}^n \delta_i}{(1-p)^2} < 0.$$

Hence, $\widehat{p}_M$ is indeed a global maxima of the likelihood function.

(c) We now derive 95% asymptotic confidence intervals for the desired parameters. It follows that the MLE for a 1-d parameter has approximately, in large samples, a normal distribution with mean equal to the true parameter and variance given by the reciprocal of the expected Fisher information.

- For θ, we have
$$\frac{\partial^2}{\partial\theta^2}\log L(\theta|\vec{z}) = \frac{-n}{\theta^2} \Rightarrow E(\frac{\partial^2}{\partial\theta^2}\log L(\theta|\vec{z})) = \frac{-n}{\theta^2}.$$
Thus, the expected information is
$$I(\theta) = -E(\frac{\partial^2}{\partial\theta^2}\log L(\theta|\vec{z})) = \frac{n}{\theta^2},$$
and so
$$Var(\widehat{\theta}_M) \approx \frac{\widehat{\theta}_M^2}{n}.$$
Therefore, a 95% confidence interval for θ is
$$\widehat{\theta}_M \pm 1.96 \times \sqrt{\frac{\widehat{\theta}_M^2}{n}} = \frac{n}{\sum_{i=1}^n \log(z_i)} \pm 1.96 \times \frac{\sqrt{n}}{\sum_{i=1}^n \log(z_i)},$$
where the number 1.96 is the 97.5th quantile of the standard normal distribution.

- For p, we have
$$\frac{\partial^2}{\partial p^2}\log L(p|\vec{\delta}) = -\frac{\sum_{i=1}^n \delta_i}{p^2} - \frac{n - \sum_{i=1}^n \delta_i}{(1-p)^2}.$$
Thus,
$$\begin{aligned} E(\frac{\partial^2}{\partial p^2}\log L(p|\vec{\delta})) &= \frac{-np}{p^2} - \frac{n-np}{(1-p)^2} \\ &= -\frac{n}{p} - \frac{n}{1-p} \\ &= -\frac{n}{p(1-p)}. \end{aligned}$$
Thus, the expected information is
$$I(p) = -E(\frac{\partial^2}{\partial p^2}\log L(p|\vec{\delta})) = \frac{p(1-p)}{n},$$
and so
$$Var(\widehat{p}_M) \approx \frac{\widehat{p}_M(1-\widehat{p}_M)}{n}.$$
Therefore, a 95% confidence interval for p is
$$\widehat{p}_M \pm 1.96 \times \sqrt{\frac{\widehat{p}_M(1-\widehat{p}_M)}{n}} = \frac{\sum_{i=1}^n \delta_i}{n} \pm 1.96 \times \sqrt{\frac{\sum_{i=1}^n \delta_i(n - \sum_{i=1}^n \delta_i)}{n^3}},$$
where the number 1.96 is the 97.5th quantile of the standard normal distribution.

Question 2

Solution: In this question, we implement stochastic regression imputation and the corresponding bootstrap based version to compute the empirical coverage probability of the 95% confidence intervals for β_1.

- First, we apply stochastic regression imputation method as follows. Note that stochastic regression imputation utilises the following model to impute the missing responses:

$$\dot{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i,\text{mis}} + \dot{\epsilon}_i,$$

where $\hat{\beta}_0$, $\hat{\beta}_1$ are the least square estimates from the observed data, and $\dot{\epsilon}_i \sim N(0, \hat{\sigma}^2)$ is randomly drawn from a zero-mean normal distribution with variance equal to the estimated variance of the residual.

```r
# Question 2 -- Stochastic regression imputation

library(mice)
load("dataex2.Rdata")

# Loop over each dataset
beta1_in_CI = rep(0,100)
#beta1_CI = matrix(0,nrow = 100, ncol = 2)
for (i in 1:100) {
  data <- dataex2[, , i]

  # Stochastic regression imputation with bootstrapping
  imps0 <- mice(data, maxit = 0)
  meth = imps0$method["Y"]
  meth <- "norm.nob" # norm.nob stands for stochastic regression imputation
  imps_boot <- mice(data, method = meth, m = 20, seed = 1, printFlag = FALSE)

  # Fit the model to each imputed dataset and extract confidence intervals for
  beta1
  fit_boot <- with(imps_boot, lm(Y ~ X))
  summary_boot <- summary(pool(fit_boot), conf.int = TRUE)
  ci_boot <- summary_boot[2, 7:8]  # beta1 is the second parameter

  # Check if true beta1 is within the CI
  beta1_in_CI[i] <- ci_boot[1] <= 3 && 3 <= ci_boot[2]
  #beta1_CI[i,] <- c(ci_boot[[1]],ci_boot[[2]])
}

# Calculate empirical coverage probability
emp_coverage <- mean(beta1_in_CI)

# Output the result for the bootstrap-based approach
cat("Empirical Coverage Probability:", emp_coverage, "\n")
```

```
## Empirical Coverage Probability: 0.88
```

- Then, we apply the bootstrapped-based stochastic regression imputation method as follows. This method utilises a different model to impute the missing responses:

$$\dot{y}_i = \dot{\beta}_0 + \dot{\beta}_1 x_{i,\text{mis}} + \dot{\epsilon}_i,$$

where $\dot{\beta}_0$, $\dot{\beta}_1$ are the least square estimates from a bootstrap sample taken from the observed data, and $\dot{\epsilon}_i \sim N(0, \dot{\sigma}^2)$ is randomly drawn from a zero-mean normal distribution with variance equal to the estimated variance of the residual under bootstrapping. Note that this bootstrap method introduces parameters uncertainty.

```r
# Question 2 -- Stochastic regression imputation with bootstrapping

# Loop over each dataset
beta1_in_CI_boot = rep(0,100)
beta1_CI_boot = matrix(0,nrow = 100, ncol = 2)
for (i in 1:100) {
  data <- dataex2[, , i]

  # Stochastic regression imputation with bootstrapping
  imps0 <- mice(data, maxit = 0)
  meth = imps0$method["Y"]
  meth <- "norm.boot" # norm.nob stands for Bootstrap multiple imputation
  imps_boot <- mice(data, method = meth, m = 20, seed = 1, printFlag = FALSE)

  # Fit the model to each imputed dataset and extract confidence intervals for
  beta1
  fit_boot <- with(imps_boot, lm(Y ~ X))
  summary_boot <- summary(pool(fit_boot), conf.int = TRUE)
  ci_boot <- summary_boot[2, 7:8]  # Assuming beta1 is the second parameter

  # Check if true beta1 is within the CI
  beta1_in_CI_boot[i] <- ci_boot[1] <= 3 && 3 <= ci_boot[2]
  beta1_CI_boot[i,] <- c(ci_boot[[1]],ci_boot[[2]])
}

# Calculate empirical coverage probability for the bootstrap-based approach
emp_coverage_boot <- mean(beta1_in_CI_boot)

# Output the result for the bootstrap-based approach
cat("Empirical Coverage Probability (Bootstrap):", emp_coverage_boot, "\n")
```

```
## Empirical Coverage Probability (Bootstrap): 0.95
```

- It follows that the empirical coverage probabilities of the 95% confidence intervals for β_1 under the stochastic regression imputation approaches and the corresponding bootstrap based version approach are 0.88 and 0.95 respectively.
- Note that the empirical coverage probability using bootstrap based version is exactly the nominal value of 95%, while the empirical coverage probability using the other method is slightly smaller. This is probably because the stochastic regression imputation method might underestimate the variance of the parameter and thus it could lead to narrower confidence intervals that do not capture the true parameter value as frequently as they should, resulting in a smaller empirical coverage probability.
- Different from the stochastic imputation method which might not fully capture the uncertainty of parameters due to missing data, the bootstrap-based approach tends to be more robust in capturing the variability in the estimators.

Question 3

Solution:

(a) We compute the likelihood of the given censored dataset as follows.

- The contribution to the likelihood from a non-censored observation is

$$\phi(y;\mu,\sigma^2) = \phi(x;\mu,\sigma^2),$$

  since $X = Y$ when $Y \geq D$.

- For the censorred obs, we have $Y < D$ and so its contribution to the likelihood is

$$\mathbb{P}(Y_i < D;\mu,\sigma^2) = \Phi(D;\mu,\sigma^2) = \Phi(x_i;\mu,\sigma^2),$$

  since $X = D$ when $Y < D$.

- Moreover, since $Y_i \sim_{i.i.d} N(\mu,\sigma^2)$, all observations $(X_i = x_i, R_i = r_i)$ are independent, and we have

$$L(\mu,\sigma^2|\vec{x},\vec{r}) = \prod_{i=1}^{n}(\phi(x_i;\mu,\sigma^2))^{r_i}(\Phi(x_i;\mu,\sigma^2))^{1-r_i},$$

  and hence

$$\log L(\mu,\sigma^2|\vec{x},\vec{r}) = \sum_{i=1}^{n}\{r_i\log(\phi(x_i;\mu,\sigma^2)) + (1-r_i)\log(\Phi(x_i;\mu,\sigma^2))\}$$

(b) We implement the method for computing the MLE of μ as follows.

```
# Question 3 -- MLE of mu

load("dataex3.Rdata")

# Define the negative log-likelihood function
neg_log_lik_cen <- function(mu) {
  x = dataex3$X
  r = dataex3$R
  sigma = 1.5
  n <- length(x)
  sum <- sum(r * log(dnorm(x, mean = mu, sd = sigma)) + (1 - r) * log(pnorm(x, mean
= mu, sd = sigma)))
  sum <- -sum
  return(sum)
}

mle_mu_result <- optim(par = 0, fn = neg_log_lik_cen)

# Estimated mu
mu_est <- mle_mu_result$par

cat("MLE-mu:", mu_est, "\n")
```

```
## MLE-mu: 5.53125
```

- The maximum likelihood estimate of μ given $\sigma^2 = 1.5^2$ is

$$\mu_{MLE} = 5.53125.$$

Question 4

Solution: We justify that missing data mechanisms (a) and (c) are ignorable for likelihood-based estimation, while (b) cannot be ignored.

A general criterion for justification is stated as follows.

> A missing data mechanism is ignorable for like-lihood inference if i) the missing data are MAR or MCAR; and ii) the parameter of the missing mechanism and the parameter of the data model are distint/disjoint.

- (a): The missing data mechanism is a function of ψ_1, ψ_2 and y_1, which indicates an MAR scenario. Note also that y_1 is fully observed and ψ_1, ψ_2 are distinct from θ. Therefore, we could conclude that (a) is ignorable for likelihood-based estimation.
- (c): The missing data mechanism is a function of ψ, μ_1 and y_1, which also indicates an MAR scenario. While μ_1 appears in the missing data mechanism, it actually does not have any effect on ψ, i.e. μ_1 and ψ are independent. Note also ψ are distinct from θ. Therefore, we could conclude that (c) is also ignorable for likelihood-based estimation.
- (b): The missing data mechanism is a function of ψ_1, ψ_2 and y_2, which indicates an MNAR scenario, since y_2 has some missing values while the missing data mechanism depends on y_2. Therefore, we could conclude that (a) is ignorable for likelihood-based estimation.

Question 5

Solution: In this question, we derive and implement an EM algorithm to compute the maximum likelihood estimate of $\vec{\beta} = (\beta_0, \beta_1)^T$.

- WLOG, assume the first m values of Y are observed and the remaining $n - m$ are missing, i.e. $\vec{y}_{\text{mis}} = (y_{m+1}, y_{m+2}, \cdots, y_n)^T \in \{0,1\}^{n-m}$, and $\vec{y}_{\text{obs}} = (y_1, y_2, \cdots, y_m)^T \in \{0,1\}^m$.
- Denote $\vec{x}_i \equiv (1, x_i)^T$. Then $\vec{x}_i^T \vec{\beta} = \beta_0 + \beta_1 x_i$. And so

$$\begin{aligned}
p_i(\vec{\beta}) &= \frac{\exp\left(\vec{x}_i^T \vec{\beta}\right)}{1 + \exp\left(\vec{x}_i^T \vec{\beta}\right)} \\
\Rightarrow \log p_i(\vec{\beta}) &= \vec{x}_i^T \vec{\beta} - \log\left(1 + \exp\left(\vec{x}_i^T \vec{\beta}\right)\right) \\
\log\left(1 - p_i(\vec{\beta})\right) &= -\log\left(1 + \exp\left(\vec{x}_i^T \vec{\beta}\right)\right).
\end{aligned}$$

  Since Y_i's are independent Bernoulli random variables with different success probability $p(\vec{\beta})$, the likelihood of the complete data $(\vec{y}_{\text{obs}}, \vec{y}_{\text{mis}})$ is

$$L(\vec{\beta}; \vec{y}_{\text{obs}}, \vec{y}_{\text{mis}}) = \prod_{i=1}^{n} (p_i(\vec{\beta}))^{y_i} (1 - p_i(\vec{\beta}))^{1-y_i} \quad ,$$

  with the corresponding log-likelihood being

$$\begin{aligned}
\log L(\vec{\beta}; \vec{y}_{\text{obs}}, \vec{y}_{\text{mis}}) &= \sum_{i=1}^{n} y_i (\log p_i(\vec{\beta})) + (1 - y_i)(\log\left(1 - p_i(\vec{\beta})\right)) \\
&= \sum_{i=1}^{n} y_i \vec{x}_i^T \vec{\beta} - \log\left(1 + \exp\left(\vec{x}_i^T \vec{\beta}\right)\right).
\end{aligned}$$

- For the E-step, we need to calculate

$$\begin{aligned} Q(\vec{\beta}|\vec{\beta}_{(k)}) &= E_{\vec{Y}_{\text{mis}}}\left[\log L(\vec{\beta};\vec{y}_{\text{obs}},\vec{y}_{\text{mis}})\Big|\vec{y}_{\text{obs}},\vec{\beta}_{(k)}\right] \\ &= \sum_{i=1}^{m} y_i\vec{x}_i^T\vec{\beta} - \log\Big(1+\exp\big(\vec{x}_i^T\vec{\beta}\big)\Big) \\ &+ \sum_{i=m+1}^{n} E_{\vec{Y}_{\text{mis}}}\left[Y_i\vec{x}_i^T\vec{\beta} - \log\Big(1+\exp\big(\vec{x}_i^T\vec{\beta}\big)\Big)\Big|\vec{y}_{\text{obs}},\vec{\beta}_{(k)}\right]. \end{aligned}$$

  Calculating the remaining expectation term, we have

$$E_{\vec{Y}_{\text{mis}}}\left[Y_i\vec{x}_i^T\vec{\beta} - \log\Big(1+\exp\big(\vec{x}_i^T\vec{\beta}\big)\Big)\Big|\vec{y}_{\text{obs}},\vec{\beta}_{(k)}\right] = \sum_{i=m+1}^{n} \frac{\vec{x}_i^T\vec{\beta}_{(k)}}{1+exp(-\vec{x}_i^T\vec{\beta}_{(k)})} - \log\Big(1+\exp\big(\vec{x}_i^T\vec{\beta}_{(k)}\big)\Big).$$

  The above expectation follows from the Bernoulli distribution of Y_i, with parameter $p(\vec{\beta}) = 1 + \exp\big(-\vec{x}_i^T\vec{\beta}\big)$. Thus,

$$\begin{aligned} Q(\vec{\beta}|\vec{\beta}_{(k)}) &= \sum_{i=1}^{m} y_i\vec{x}_i^T\vec{\beta} - \log\Big(1+\exp\big(\vec{x}_i^T\vec{\beta}\big)\Big) \\ &+ \sum_{i=m+1}^{n} \frac{\vec{x}_i^T\vec{\beta}_{(k)}}{1+exp(-\vec{x}_i^T\vec{\beta}_{(k)})} - \log\Big(1+\exp\big(\vec{x}_i^T\vec{\beta}_{(k)}\big)\Big). \end{aligned}$$

- Now we can proceed the M-steps by minimising the negative of the Q-function with the helop of the `optim` package, implemented as follows.

```
# Method 2:

load("dataex5.Rdata")

# Reorder the dataset so that the first m rows are complete cases
complete_rows_ex5 <- complete.cases(dataex5)
complete_data_ex5 <- dataex5[complete_rows_ex5, ]
incomplete_data_ex5 <- dataex5[!complete_rows_ex5, ]
r_data_ex5 <- rbind(complete_data_ex5, incomplete_data_ex5)


x_yobs = complete_data_ex5$X
y_obs = complete_data_ex5$Y
x_ymis = incomplete_data_ex5$X

# Function for EM algorithms
haha1 = function(beta_0_EM, beta_1_EM, tol = 1e-6, max_iter = 1000){
  # beta_0_EM, beta_1_EM: initial guessing

  # implement the negative of the Q-function
  objective_function <- function(beta) {

      beta_0 <- beta[1]
      beta_1 <- beta[2]
```

```r
      K_yobs <- beta_0 + beta_1 * x_yobs
      K_ymis <- beta_0_EM + beta_1_EM * x_ymis

      ssd <- - sum(y_obs * K_yobs) + sum(log(1 + exp(K_yobs))) - sum(K_ymis / (1 +
exp(K_ymis))) + sum(log(1 + exp(K_ymis)))
      return(ssd)
    }

  # Looping algorithm
  for (i in 1:max_iter) {
    # Previous step's beta values, for convergence check
    beta_prev = c(beta_0_EM, beta_1_EM)

    # Define the objective function for optimization
    # Want to derive beta given the previous value beta_0_EM, beta_1_EM
    objective_function <- function(beta) {

      beta_0 <- beta[1]
      beta_1 <- beta[2]

      K_yobs <- beta_0 + beta_1 * x_yobs
      K_ymis <- beta_prev[1] + beta_prev[2] * x_ymis

      ssd <- - sum(y_obs * K_yobs) + sum(log(1 + exp(K_yobs))) - sum(K_ymis / (1 +
exp(K_ymis))) + sum(log(1 + exp(K_ymis)))
      return(ssd)
    }



    # Use optim to minimise the objective function
    optim_result <- optim(par = beta_prev, fn = objective_function)

    # Extract the optimised beta values
    beta_0_EM <- optim_result$par[1]
    beta_1_EM <- optim_result$par[2]

    if (sqrt(sum((c(beta_0_EM, beta_1_EM) - beta_prev)^2)) < tol) {
      cat("Convergence achieved after", i, "iterations.\n")
      break
    }

    if (i == max_iter) {
      cat("Maximum iterations reached without convergence.\n")
    }

  }

  # Final beta values
  cat("Final beta_0:", beta_0_EM, "\n")
  cat("Final beta_1:", beta_1_EM, "\n")
}
```

```r
haha1(beta_0_EM = 3, beta_1_EM = 3)
```

```
## Convergence achieved after 3 iterations.
## Final beta_0: 0.9756532
## Final beta_1: -2.480037
```

- It follows that
$$\hat{\beta}_0 = 0.976, \hat{\beta}_1 = -2.480.$$

5.1 Alternative implementation

Alternatively, one might process the M-step by solving the FOC of the Q-function.

1. It follows from the complete log-likelihood that
$$\begin{aligned}\log L(\vec{\beta}; \vec{y}_{\text{obs}}, \vec{y}_{\text{mis}}) &= \sum_{i=1}^{n} y_i \vec{x}_i^T \vec{\beta} - \log\Big(1 + \exp\big(\vec{x}_i^T \vec{\beta}\big)\Big) \\ &= \vec{\beta}^T \vec{t}(\vec{Y}_{\text{c}}) - b(\vec{\beta}) + c(\vec{Y}_{\text{c}}).\end{aligned},$$
where
$$\vec{t}(\vec{Y}_{\text{c}}) = \sum_{i=1}^{n} y_i \vec{x}_i, b(\vec{\beta}) = \sum_{i=1}^{n} \log\Big(1 + \exp\big(\vec{x}_i^T \vec{\beta}\big)\Big), c(\vec{Y}_{\text{c}}) = 0,$$
with $\vec{Y}_{\text{c}}^T = (\vec{Y}_{\text{obs}}^T, \vec{Y}_{\text{mis}}^T)$ One can observe that the complete data belongs to a exponential family (in canonical form). Therefore, it follows that
$$E(t(\vec{Y}_{\text{c}})) = \frac{\partial b(\vec{\beta})}{\partial \vec{\beta}}.$$

2. Note also that the Q-function can be written as
$$\begin{aligned}Q(\vec{\beta}|\vec{\beta}_{(k)}) &= E_{\vec{Y}_{\text{mis}}}\Big[\log L(\vec{\beta}; \vec{y}_{\text{obs}}, \vec{y}_{\text{mis}}) \Big| \vec{y}_{\text{obs}}, \vec{\beta}_{(k)}\Big] \\ &= E_{\vec{Y}_{\text{mis}}}\Big[\vec{\beta}^T \vec{t}(\vec{Y}_{\text{c}}) - b(\vec{\beta}) \Big| \vec{y}_{\text{obs}}, \vec{\beta}_{(k)}\Big] \\ &= \vec{\beta}^T E_{\vec{Y}_{\text{mis}}}\Big[\vec{t}(\vec{Y}_{\text{c}}) \Big| \vec{y}_{\text{obs}}, \vec{\beta}_{(k)}\Big] - b(\vec{\beta}).\end{aligned}$$
Therefore, the corresponding FOC is
$$\frac{\partial Q(\vec{\beta}|\vec{\beta}_{(k)})}{\partial \vec{\beta}} = E_{\vec{Y}_{\text{mis}}}\Big[\vec{t}(\vec{Y}_{\text{c}}) \Big| \vec{y}_{\text{obs}}, \vec{\beta}_{(k)}\Big] - E(t(\vec{Y}_{\text{c}})|\vec{\beta}) = 0.$$
Hence, an alternative M-step is to find $\vec{\beta}_{(k+1)}$ s.t.
$$E_{\vec{Y}_{\text{mis}}}\Big[\vec{t}(\vec{Y}_{\text{c}}) \Big| \vec{y}_{\text{obs}}, \vec{\beta}_{(k)}\Big] = E(t(\vec{Y}_{\text{c}})|\vec{\beta}_{(k+1)}).$$
Note that the expectation on the left is actually the E-step.

3. We first process the E-step:

$$
\begin{aligned}
E_{\vec{Y}_{\text{mis}}}\left[\vec{t}(\vec{Y}_{\text{c}})\middle|\vec{y}_{\text{obs}},\vec{\beta}_{(k)}\right] &= E_{\vec{Y}_{\text{mis}}}\left[\left[\begin{matrix}\sum_{i=1}^m y_i\\ \sum_{i=1}^m y_i x_i\end{matrix}\right]+\left[\begin{matrix}\sum_{i=m+1}^n Y_i\\ \sum_{i=m+1}^n Y_i x_i\end{matrix}\right]\middle|\vec{y}_{\text{obs}},\vec{\beta}_{(k)}\right]\\
&= \left[\begin{matrix}\sum_{i=1}^m y_i\\ \sum_{i=1}^m y_i x_i\end{matrix}\right]+E_{\vec{Y}_{\text{mis}}}\left[\left[\begin{matrix}\sum_{i=m+1}^n Y_i\\ \sum_{i=m+1}^n Y_i x_i\end{matrix}\right]\middle|\vec{y}_{\text{obs}},\vec{\beta}_{(k)}\right]\\
&= \left[\begin{matrix}\sum_{i=1}^m y_i\\ \sum_{i=1}^m y_i x_i\end{matrix}\right]+\sum_{i=m+1}^n\left[\begin{matrix}1\\ x_i\end{matrix}\right]\cdot\frac{1}{1+\exp\big(-\beta_{0(k)}-\beta_{1(k)}x_i\big)}.
\end{aligned}
$$

4. Then, we write the M-step as a system of non-linear equations:

$$
\begin{cases}
\sum_{i=1}^n \frac{1}{1+\exp(-\beta_0-\beta_1 x_i)} = \sum_{i=1}^m y_i + \sum_{i=m+1}^n \frac{1}{1+\exp\big(-\beta_{0(k)}-\beta_{1(k)}x_i\big)}\\
\sum_{i=1}^n \frac{x_i}{1+\exp(-\beta_0-\beta_1 x_i)} = \sum_{i=1}^m x_i y_i + \sum_{i=m+1}^n \frac{x_i}{1+\exp\big(-\beta_{0(k)}-\beta_{1(k)}x_i\big)}
\end{cases}
$$

5. To solve this system, we transfer it to an unconstraint optimisation problem. Note that the RHS of the two equations are all fixed. We denote the values of them to be A, B respectively. Also it follows that solving the system is equivalent to find the minima of the following problem:

$$
\min_{\beta_0,\beta_1}\ \left(\sum_{i=1}^n \frac{1}{1+\exp(-\beta_0-\beta_1 x_i)}-A\right)^2+\left(\sum_{i=1}^n \frac{x_i}{1+\exp(-\beta_0-\beta_1 x_i)}-B\right)^2
$$

6. We can now implement the above process in the following code chunk.

```r
load("dataex5.Rdata")

complete_rows_ex5 <- complete.cases(dataex5)
complete_data_ex5 <- dataex5[complete_rows_ex5, ]
incomplete_data_ex5 <- dataex5[!complete_rows_ex5, ]
r_data_ex5 <- rbind(complete_data_ex5, incomplete_data_ex5)



x_yobs = complete_data_ex5$X
y_obs = complete_data_ex5$Y
x_ymis = incomplete_data_ex5$X



haha = function(beta_0_EM, beta_1_EM, tol = 1e-6, max_iter = 1000){
  solve_beta <- function(x, A, B) {
    # Define the objective function for optimization
    objective_function <- function(beta) {

      beta_0 <- beta[1]
      beta_1 <- beta[2]

      # Calculate the sums for the given beta values
      sum1 <- sum(1 / (1 + exp(-beta_0 - beta_1 * x)))
      sum2 <- sum(x / (1 + exp(-beta_0 - beta_1 * x)))

      # Calculate the sum of squared differences from A and B
      ssd <- (sum1 - A)^2 + (sum2 - B)^2
      return(ssd)
    }

    # Initial guesses for beta_0 and beta_1
    init_guess <- c(beta_0_EM, beta_1_EM)  # Based on prior knowledge

    # Use optim to minimise the objective function
    optim_result <- optim(par = init_guess, fn = objective_function, method =
"BFGS",control = list(reltol = 1e-6))

    optimised_beta <- optim_result$par

    return(list(beta_0 = optimised_beta[1], beta_1 = optimised_beta[2]))
  }

  # Looping algorithm
  for (i in 1:max_iter) {
    # Previous step's beta, for convergence check
    beta_prev = c(beta_0_EM, beta_1_EM)

    # Update A and B based on new beta
    A <- sum(y_obs) + sum( 1 / (1 + exp(-beta_0_EM - beta_1_EM * x_ymis)))  # Update
A using beta_0 and beta_1
    B <- sum(y_obs * x_yobs) + sum(x_ymis / (1 + exp(-beta_0_EM - beta_1_EM *
x_ymis)))  # Update B using beta_0 and beta_1

    # Step 3: Solve for beta using the solve_beta function
    beta_sol <- solve_beta(r_data_ex5$X, A, B)
    beta_0_EM <- beta_sol$beta_0
    beta_1_EM <- beta_sol$beta_1


    # Check for convergence
    if (sqrt(sum((c(beta_0_EM, beta_1_EM) - beta_prev)^2)) < tol) {
      cat("Convergence achieved after", i, "iterations.\n")
      break
    }

    #Break the loop if max_iterations reached without convergence
    if (i == max_iter) {
      cat("Maximum iterations reached without convergence.\n")
    }
  }

  # Final beta values
  cat("Final beta_0:", beta_0_EM, "\n")
  cat("Final beta_1:", beta_1_EM, "\n")
}


haha(beta_0_EM = 1, beta_1_EM = 1)
```

```
## Convergence achieved after 30 iterations.
## Final beta_0: 0.9755264
## Final beta_1: -2.480383
```