Homework 2

Bhargav Ramudu Manam

12/8/2022

This homework is to be sent by mail to aude.sportisse@inria.fr (mailto:aude.sportisse@inria.fr) before January 25 2023.

- The first part consists in comparing imputation methods on a data set.
- The second part is the implementation of the EM algorithm for a logistic regression with missing values in the outcome variable y.

PART-I

Comparison of imputation methods

The goal is to compare imputation methods on the SBS5242 dataset containing missing values, which is a synthetic subset (but realistic) of the Austrian structural business statistics data.

Viewing the first six rows of the dataset:

```
library(VIM)
data(SBS5242)
XNA <- SBS5242
head(XNA)
```

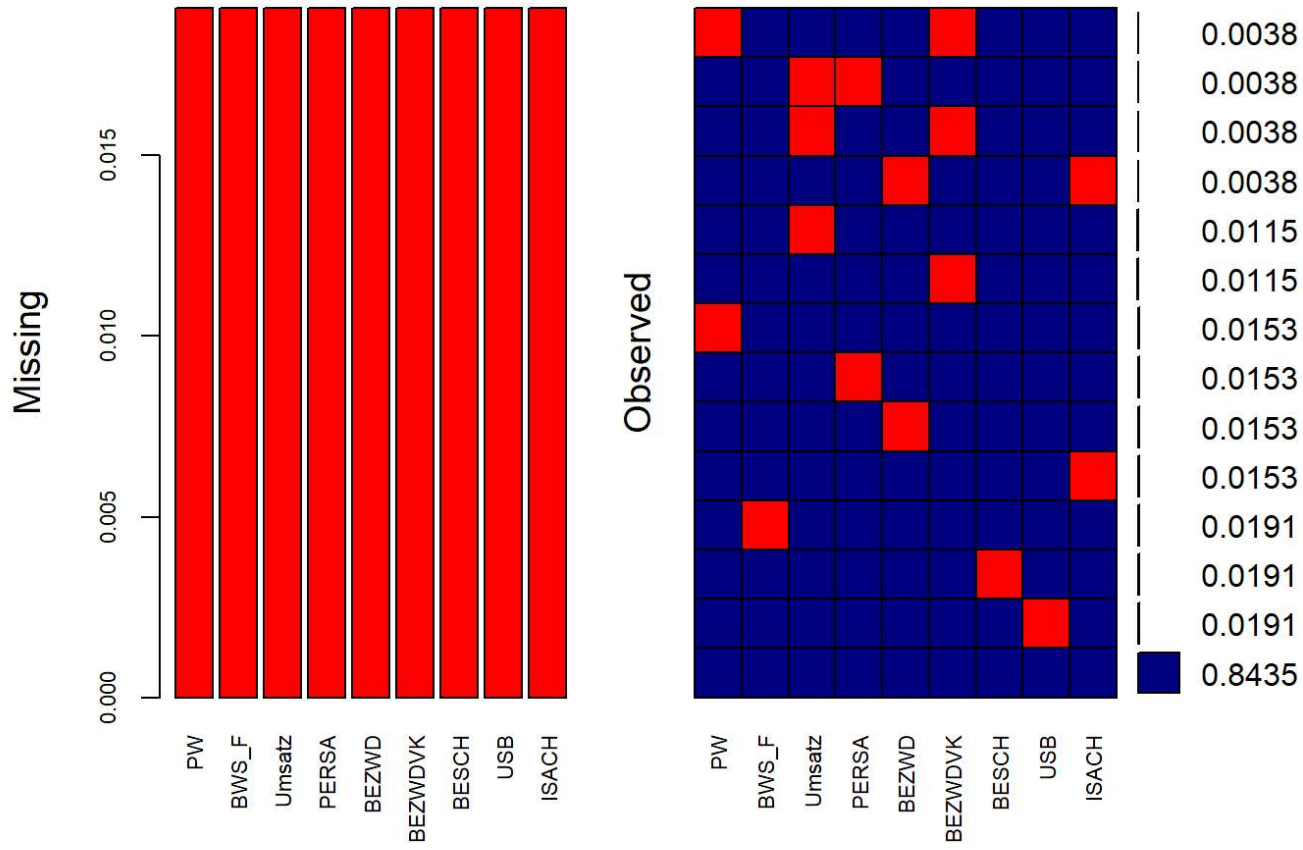
```
##            PW       BWS_F      Umsatz          PERSA       BEZWD     BEZWDVK
## 1 43888.85242 35081.07414 3737.974701 11815.3610831 37577.19455 379.0521485
## 2  1449.40170  1239.13170  120.099200     5.9457950  1251.43080   8.6653528
## 3  1009.61541   944.93722   79.832300   163.2859277   857.75827   4.2969994
## 4  1218.17072   942.29390  104.851729     1.9011026  1063.73052  20.4941952
## 5   517.36342   450.46195   41.326691    18.0060038   448.09227   2.0590045
## 6    67.19103    44.91809    5.003297    -0.6017785    62.71034  -0.2673511
##       BESCH         USB      ISACH
## 1 310.09482 30.48368699 6575.33408
## 2  18.47358 -0.41414050   58.02441
## 3  24.73831  0.35008017   12.33191
## 4  30.44422 -0.37563076  513.91041
## 5  20.18175 -0.09648481   21.44578
## 6  13.85199 -0.68410116   25.90378
```

* Visualize the missing values with the function **aggr** of the **VIM** package and compute the global

percentage of missing values in the dataset.

Solution:

```
aggr(XNA, col=c("navyblue","red"), numbers=TRUE, sortVars=TRUE, labels=names(XNA), cex.axis=.7,
gap=3, ylab=c("Missing","Observed"),legend=TRUE)
```




```
##
##  Variables sorted by number of missings:
##  Variable      Count
##        PW 0.01908397
##     BWS_F 0.01908397
##    Umsatz 0.01908397
##     PERSA 0.01908397
##     BEZWD 0.01908397
##   BEZWDVK 0.01908397
##     BESCH 0.01908397
##       USB 0.01908397
##     ISACH 0.01908397
```

```
MissPct <- function(tab){
  total_data <- length(tab)
  missing_quantity <- sum(is.na(tab))
  pct_na <- missing_quantity/total_data * 100
  return(cat("Percentage of missing values in the dataset: ",pct_na,"%"))
}

MissPct(XNA)
```

```
## Percentage of missing values in the dataset:  1.908397 %
```

Therefore, The percentage of the missing values in the initial dataset is: 1.908397 %

* Introduce 30% synthetic missing values in the dataset. You can use the function **synthetic_MCAR** (defined below) or create your own function. Explain why the synthetic missing values introduced as such are MCAR.

Solution:

Creating the function:

```
###This function takes as input
# XNA: a dataset containing missing values
# percNA: the percentage of missing values we want to introduce in addition.
###This functions returns a list containing
# XNA_new: a dataset containing the old and the new added missing values,
# mask: a vector with the indexes of the new added missing values

synthetic_MCAR <- function(XNA,percNA){
  trueNA <- which(is.na(XNA))
  nbNA <- round(prod(dim(XNA))*percNA)
  syntheticNA <- sample(setdiff(1:prod(dim(XNA)),trueNA),nbNA,replace=FALSE)
  XNA_new <- XNA
  XNA_new[syntheticNA] <- NA
  return(list(XNA_new=XNA_new,mask=syntheticNA))
}
```

Introducing 30% missing values:

```
new_miss <- synthetic_MCAR(XNA,0.3)
XNA_new <- new_miss$XNA_new
mask <- new_miss$mask

MissPct(XNA_new)
```

```
## Percentage of missing values in the dataset:  31.89143 %
```

Therefore, the total no. of missing values in the new dataset is: 31.89143 %

The synthetic missing values introduced are MCAR because:

For the **Missing Data Mechanism** to be **MCAR** we require $\mathbb{P}(M|XNA) = \mathbb{P}(M)$;

Here, the function **sample()** generates the indices vector, **RANDOMLY** where the synthetic **NA's** to be introduced and it **does not depend on our data (XNA).** Hence, they are considered MCAR.

- Compare three imputation methods (mean imputation, another single imputation, multiple imputation) by computing the MSE for the synthetic missing values.

Solution:

Defining MSE function:

```
MSE <- function(X1, X2){ return(norm(as.matrix(X1 - X2),type="F")/norm(as.matrix(X2),type
="F"))}
```

i. Mean Imputation:

```
ImputeMean <- function(tab){
  m <- apply(tab, 2, mean, na.rm = TRUE)
  tab <- sapply(1:ncol(tab), function(x) ifelse(is.na(tab[,x]), m[x], tab[,x]))
  tab <- as.data.frame(tab)
  return(tab)
}

XNA_mean_imp <- ImputeMean(XNA_new)
MSE_Mean <- MSE(XNA[mask], data.matrix(XNA_mean_imp)[mask])
```

ii. Soft Imputation:

```
library(softImpute)
```

```
## Loading required package: Matrix
```

```
## Loaded softImpute 1.4-1
```

```
library(Matrix)
Fx_SoftImpute <- function(tab){
  min_dim <- min(dim(tab))
  min_dim <- min_dim-1
  sft <- softImpute(x = tab, rank.max = min_dim, lambda = 0.0001, type = "als", maxit = 500)
  tab_sft <- sft$u %*% diag(sft$d) %*% t(sft$v) # compute the factorization
  tab_sft[which(!is.na(tab))] <- tab[which(!is.na(tab))]
  return(tab_sft)
}
XNA_soft_imp <- Fx_SoftImpute(XNA_new)

MSE_Soft <- MSE(XNA[mask], data.matrix(XNA_soft_imp)[mask])
```

iii. Multiple Imputation:

```
library(mice)
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:softImpute':
##
##     complete
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
MltImpute <- function(tab, no_imp){
  tab_in <- matrix(tab, nrow = nrow(tab), ncol = ncol(tab))
  capture.output(mice_imputed <- mice(data = tab_in, m=no_imp))
  imp <- 0
  for (i in 1:no_imp) { imp <- imp + mice::complete(mice_imputed, i)}
  imp <- imp/no_imp

  tab[which(is.na(tab))] <- unlist(imp)[which(is.na(tab))]

  return(tab)
}
XNA_mlt_imp <- MltImpute(XNA_new, 10)
MSE_Multi <- MSE(XNA[mask], data.matrix(XNA_mlt_imp)[mask])
```

Comparison of different Imputation Methods:

Mean-Squared Error using different Imputation Methods are:

```
cat("MSE with Mean Imputation is: ",MSE_Mean, "\n")
```

```
## MSE with Mean Imputation is:  3.592128
```

```
cat("MSE with Soft Imputation is: ",MSE_Soft, "\n")
```

```
## MSE with Soft Imputation is:  9.875476
```

```
cat("MSE with Multiple Imputation is: ",MSE_Multi, "\n")
```

```
## MSE with Multiple Imputation is:  2.19909
```

From above, it is clear that Multiple Imputation works best. Mean Imputation performance is better than Soft Imputation, this might be because Soft Imputation needs to be further fine tuned to find optimal hyper-parameters which highly depend on our data we have at hand and inter-dependencies over the samples (i.e, it will easier if the samples are truly i.i.d's).

- To analyse the stochasticity implied by the introduction of missing values, repeat the procedure (introduction of synthetic missing values and imputation) several times and plot the distribution of the MSEs by showing a boxplot.

Solution:

```
Mean_MSE <- c()
Soft_MSE <- c()
Multiple_MSE <- c()

for (i in 1:50){
  new_miss <- synthetic_MCAR(XNA,0.3)
  XNA_new <- new_miss$XNA_new
  mask <- new_miss$mask

  XNA_mean_imp <- ImputeMean(XNA_new)
  MSE_Mean <- MSE(XNA[mask], data.matrix(XNA_mean_imp)[mask])
  Mean_MSE <- append(Mean_MSE, MSE_Mean)

  XNA_soft_imp <- Fx_SoftImpute(XNA_new)
  MSE_Soft <- MSE(XNA[mask], data.matrix(XNA_soft_imp)[mask])
  Soft_MSE <- append(Soft_MSE, MSE_Soft)

  XNA_mlt_imp <- MltImpute(XNA_new, 10)
  MSE_Multi <- MSE(XNA[mask], data.matrix(XNA_mlt_imp)[mask])
  Multiple_MSE <- append(Multiple_MSE, MSE_Multi)
}
```

```
## Warning: Number of logged events: 1
```

```
## Warning: Number of logged events: 2
```

```
## Warning: Number of logged events: 42
```

```
## Warning: Number of logged events: 8
```

```
## Warning: Number of logged events: 4
```

```
## Warning: Number of logged events: 20
```

```
## Warning: Number of logged events: 39
```

```
## Warning: Number of logged events: 16
```

```
## Warning: Number of logged events: 1
```

```
## Warning: Number of logged events: 17

## Warning: Number of logged events: 17
```

```
## Warning: Number of logged events: 28
```

```
## Warning: Number of logged events: 15
```

```
## Warning: Number of logged events: 34
```

```
## Warning: Number of logged events: 21
```

```
## Warning: Number of logged events: 8
```

```
## Warning: Number of logged events: 1
```

```
## Warning: Number of logged events: 50
```

```
## Warning: Number of logged events: 32
```

```
## Warning: Number of logged events: 6
```

```
## Warning: Number of logged events: 34
```

```
## Warning: Number of logged events: 5
```

```
## Warning: Number of logged events: 7
```

```
## Warning: Number of logged events: 47
```

```
## Warning: Number of logged events: 37
```

```
## Warning: Number of logged events: 16
```

```
## Warning: Number of logged events: 45
```

```
## Warning: Number of logged events: 37
```

```
## Warning: Number of logged events: 12
```

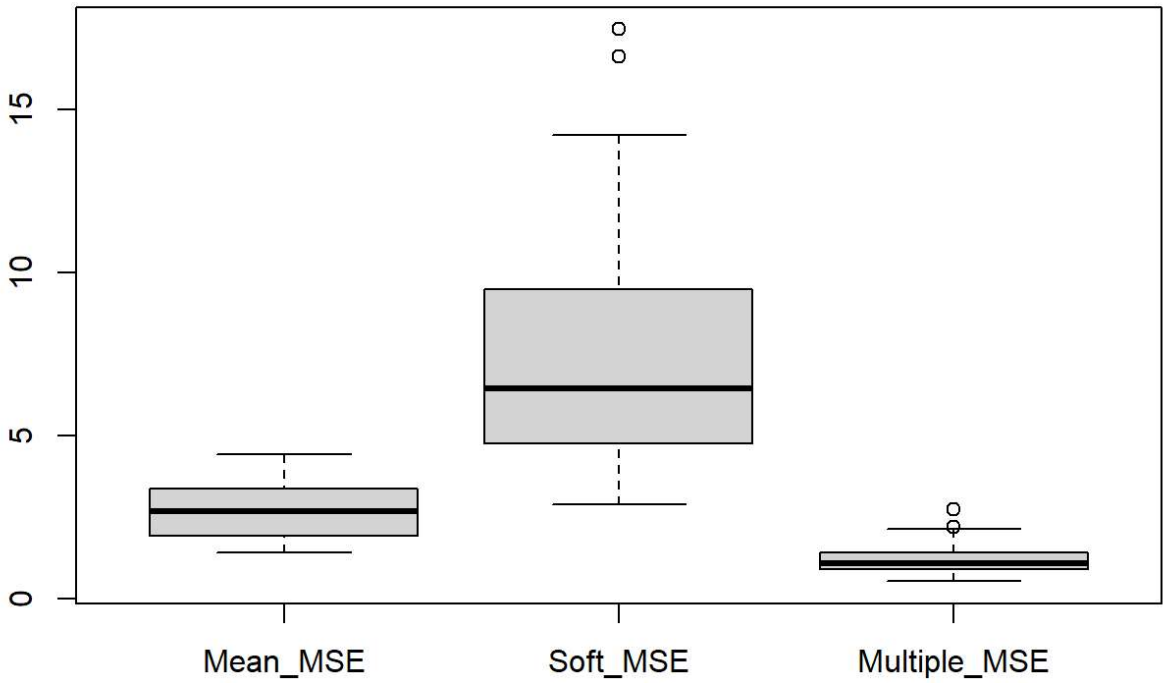
```
## Warning: Number of logged events: 17
```

```
## Warning: Number of logged events: 42
```

```
## Warning: Number of logged events: 21
```

```
## Warning: Number of logged events: 42
```

```
MSE_all <- cbind(Mean_MSE, Soft_MSE, Multiple_MSE)
boxplot(MSE_all)
```



From the blox plot above, it is further evident that Multiple Imputation performs the best among the three Imputation methods chosen followed by Mean Imputation. Soft Imputation has the overall worst MSE and even high variance. Hence, it is recommended to impute by Multiple Imputation, which is more reliable among the three methods, as it has lowest MSE and Variance.

PART-II

EM algorithm for logistic regression

Let us consider an i.i.d. sample $(y_i, X_i)_{i=1,\dots,n}$.

$y_i \in \{0, 1\}$ is the outcome variable, $X_i \in \mathbb{R}^d$ are the covariates, $\beta \in \mathbb{R}^d$ is the regression parameter.

We assume the logistic regression:

$$\mathbb{P}(y_i = 1|X_i;\beta) = \frac{1}{1+\exp(-X_i^T\beta)}$$

In the sequel, we consider that y contains some MCAR values and we derive an EM algorithm to estimate β.

Solution:

$$\mathbf{Data = (y_i, X_i)_{i=1,\dots,n}}$$

Where, $y_i \in \{0,1\}$, $X_i \in \mathbb{R}^d$, $\forall i = 1,\dots,n$.

Likelihood:

Likelihood is defined by:

$$L(Data, M) = \mathbb{P}(Data) * \mathbb{P}(M|Data)$$

Where M is the **Missing data pattern** and $\mathbb{P}(M|Data)$ is the **Missing data mechanism**.

Log- Likelihood is given by:

$$\begin{aligned}\mathcal{L}(Data, M) &= log(L(Data, M)) \\ &= log(\mathbb{P}(Data) * \mathbb{P}(M|Data)) \\ &= log(\mathbb{P}(Data)) + log(\mathbb{P}(M|Data)) \end{aligned} \tag{1}$$

$$Data = y, X = (y_i, X_i)_{i=1,\dots,n} = ((y_1, X_1), \dots, (y_n, X_n)) \implies \mathbb{P}(Data) = \mathbb{P}((y_1, X_1), \dots, (y_n, X_n))$$

Since the samples are **i.i.d**, we can write:

$$\begin{aligned}\mathbb{P}(Data) &= \prod_{i=1}^{n} \mathbb{P}(y_i, X_i) \\ &= \prod_{i=1}^{n} \mathbb{P}(y_i|X_i;\beta) * \mathbb{P}(X_i).\end{aligned}$$

And,

$$\begin{aligned}log(\mathbb{P}(Data)) &= log(\prod_{i=1}^{n} \mathbb{P}(y_i|X_i;\beta) * \mathbb{P}(X_i)) \\ &= \sum_{i}^{n} log(\mathbb{P}(y_i|X_i;\beta) * \mathbb{P}(X_i)) \\ &= \sum_{i}^{n} log(\mathbb{P}(y_i|X_i;\beta)) + \sum_{i}^{n} log(\mathbb{P}(X_i)) \end{aligned} \tag{2}$$

Since the data is **MCAR**,

$$\mathbb{P}(M|Data) = \mathbb{P}(M) \implies log(\mathbb{P}(M|Data)) = log(\mathbb{P}(M)) \tag{3}$$

and by substituting equations (2) and (3) in (1), we get:

$$\mathcal{L}(y, X, M;\beta) = \sum_{i}^{n} log(\mathbb{P}(y_i|X_i;\beta)) + \sum_{i}^{n} log(\mathbb{P}(X_i)) + log(\mathbb{P}(M)) \tag{4}$$

The last two terms in the above equation (4) can be excluded from the Likelihood estimation, since they are parameter free and act as only noise (Max. Likelihood Estimation involves only the parameter dependent terms).

Therefore,

$\mathcal{L}(y, X; \beta) \propto \sum_i^n log(\mathbb{P}(y_i|X_i; \beta)) \implies$

$$\begin{aligned}\mathcal{L}(y|X; \beta) &\approx \sum_i^n log(\mathbb{P}(y_i|X_i; \beta)) \\ &= \sum_i^n \mathcal{L}_i(y_i|X_i; \beta) \end{aligned} \tag{5}$$

Where, $\mathcal{L}_i(y_i|X_i; \beta) := log(\mathbb{P}(y_i|X_i; \beta))$

To calculate the likelihood, we need to evaluate (in equation(5)) the quantity $\mathbb{P}(y_i|X_i; \beta)$, for each given X_i.

We know that:

$$\mathbb{P}(y_i = 1|X_i; \beta) = \frac{1}{1 + \exp(-X_i^T \beta)} \implies \mathbb{P}(y_i = 0|X_i; \beta) = 1 - \mathbb{P}(y_i = 1|X_i; \beta)$$

Hence, $ (y_i| X_i;)= (y_i=1|X_i;)^{}*(y_i=0|X_i;)^{}$

Where, $\mathbf{1}$ is the Indicator function.

Then,

$$\begin{aligned}\mathcal{L}_i(y_i|X_i; \beta) &= log(\mathbb{P}(y_i|X_i; \beta)) \\ &= log(\mathbb{P}(y_i = 1|X_i; \beta)^{\mathbf{1_{y_i=1}}} * \mathbb{P}(y_i = 0|X_i; \beta)^{\mathbf{1_{y_i=0}}}) \\ &= \mathbf{1_{y_i=1}} * log(\mathbb{P}(y_i = 1|X_i; \beta) + \mathbf{1_{y_i=0}} * log(\mathbb{P}(y_i = 0|X_i; \beta) \\ &= \mathbf{1_{y_i=1}} * log(\frac{1}{1 + \exp(-X_i^T \beta)}) + \mathbf{1_{y_i=0}} * log(1 - \frac{1}{1 + \exp(-X_i^T \beta)}) \\ &= \mathbf{1_{y_i=1}} * log(\pi_i) + \mathbf{1_{y_i=0}} * log(1 - \pi_i) \end{aligned} \tag{6}$$

Where $\pi_i = \mathbb{P}(y_i = 1|X_i; \beta) = \frac{1}{1+\exp(-X_i^T \beta)}$

- Write the observed log-likelihood. Note that in this case, we can maximize it with numerical methods (for example using the R function **glm** and argument **family=binomial**).

Solution:

Observed Log- Likelihood:

Suppose, we observe n_{Obs} , y_i's $\forall i = \{1, 2, \ldots, n_{Obs}\}$, out of n. i.e., $n = n_{Obs} + n_{Miss}$.

Where, n_{Obs} : No. of observed y_i's and n_{Miss} : No. of missing y_i's.

$$\begin{aligned}\mathcal{L}_{OBS}(y|X;\beta) &= \sum_{i}^{n_{Obs}} \mathcal{L}_i(y_i|X_i;\beta) \\ &= \sum_{i=1}^{n_{Obs}} (1_{\mathbf{y_i=1}} * log(\pi_i) + 1_{\mathbf{y_i=0}} * log(1-\pi_i)) \\ &= \sum_{i=1}^{n_{Obs}} log(\mathbb{P}(y=y_i|X_i;\beta)) \end{aligned} \tag{7}$$

Where $\pi_i = \mathbb{P}(y_i = 1|X_i;\beta) = \frac{1}{1+\exp(-X_i^T\beta)}$

- Although maximizing the observed likelihood can be done easily, we will also derive an EM algorithm to maximize it. Write the full log-likelihood.

Solution:

Full (Complete Data) Log- Likelihood:

From equations (6) and (7), we can write the full log- likelihood as follows:

$$\begin{aligned}\mathcal{L}_{FULL}(y|X;\beta) &= \sum_{i}^{n} \mathcal{L}_i(y_i|X_i;\beta) \\ &= \mathcal{L}_{OBS}(y|X;\beta) + \mathcal{L}_{MISS}(y|X;\beta) \\ &= \sum_{i=1}^{n_{Obs}} log(P(y=y_i|X_i;\beta)) + \sum_{i \in n_{Miss}} \sum_{y_i \in \{0,1\}} 1_{y_i} \log(p(y_i|X_i;\beta)) \end{aligned} \tag{8}$$

In essence, the complete data log- likelihood can be written as:

$$\mathcal{L}_{FULL}(y|X;\beta) = \sum_{i}^{n} \mathcal{L}_i(y_i|X_i;\beta)$$

where

$$\mathcal{L}_i(y_i|X_i;\beta) = \begin{cases} \sum_{y_i \in \{0,1\}} 1_{y_i} \log p(y_i|x_i;\beta) & \text{if } y_i \text{ is missing} \\ \log p(y_i|x_i;\beta) & \text{otherwise.} \end{cases} \tag{9}$$

- Show that the E-step can be written as follows

$$Q(\beta;\beta^{(r)}) = \sum_{i=1}^{n} Q_i(\beta;\beta^{(r)})$$

where

$$Q_i(\beta;\beta^{(r)}) = \begin{cases} \sum_{y_i \in \{0,1\}} p(y_i|x_i;\beta^{(r)}) \log p(y_i|x_i;\beta) & \text{if } y_i \text{ is missing} \\ \log p(y_i|x_i;\beta) & \text{otherwise.} \end{cases}$$

Solution:

E- Step Formulation:

By definition:

$$
\begin{aligned}
Q(\beta;\beta^{(r)}) &= \mathbb{E}[\mathcal{L}_{FULL}(y|X;\beta)|y^{Obs},X^{Obs};\beta^{(r)}] \\
&= \mathbb{E}[\sum_{i}^{n}\mathcal{L}_i(y_i|X_i;\beta);\beta^{(r)}] \\
&= \sum_{i}^{n}\mathbb{E}[\mathcal{L}_i(y_i|X_i;\beta);\beta^{(r)}] \\
&= \sum_{i=1}^{n} Q_i(\beta;\beta^{(r)}) \qquad (10)
\end{aligned}
$$

Where $Q_i(\beta;\beta^{(r)}) := \mathbb{E}[\mathcal{L}_i(y_i|X_i;\beta);\beta^{(r)}]$

From equation (8), we can write:

$$
Q_i(\beta;\beta^{(r)}) = \begin{cases} \sum_{y_i\in\{0,1\}} \mathbb{E}[\mathbf{1}_{y_i}]\log p(y_i|x_i;\beta) & \text{if } y_i \text{ is missing} \\ \log p(y_i|x_i;\beta) & \text{otherwise.} \end{cases}
$$

And, $\mathbb{E}[\mathbf{1}_{y_i}] = p(y_i|x_i;\beta^{(r)})\square$.

Therefore, E-step can be written as shown above in equation (10).

- Code the EM algorithm.

Hint 1: remark that $Q(\beta;\beta^{(r)})$ can be seen as a weighted complete data log-likelihood ($\sum_{i=1}^{n}\sum_{k=1}^{n_i}\omega_{y_k}\log p(y_k|x_k;\beta)$) based on a sample size $N=\sum_{i=1}^{n}n_i$, where $n_i=2$ if y_i is missing and $n_i=1$ otherwise. The weights are $\omega_{y_k}=p(y_k|x_k;\beta^{(r)})$ if y_k is missing and $\omega_{y_k}=1$ if y_k is observed.

*Hint 2: For the M-step, you can simply use the function **glm** with the argument **weights**.*

Solution:

EM Algorithm:

```r
# Function to calculate probability y= 1 given X and beta

prob_y_xbeta <- function(x, beta, intercept=FALSE){
  if(intercept){
    return(1/(1+exp(-x%*%beta[-1] -beta[1])))
  }
  else{
    return(1/(1+exp(-x%*%beta)))
  }
}

# Function to return Observed Likelihood

obs_lik <- function(y, x, beta, intercept=FALSE){
  return(y*prob_y_xbeta(x, beta, intercept)+(1-y)*(1-prob_y_xbeta(x, beta, intercept)))
}

# Function to perform EM Algorithm

em_log_reg <- function (x, y, iter=25, draw_plot=FALSE, intercept=FALSE, title=''){

  # Random Initialization
  n_vars <- ncol(x)
  beta <- if(intercept){rnorm(n_vars+1)}
  else(rnorm(n_vars))

  # Creating new vectors where y is missing (for y=0 and y=1)
  mask <- is.na(y)
  idx_miss <- which(mask)
  idx_obs <- which(!mask)

  y_obs <- y[idx_obs]
  count_miss <- length(idx_miss)

  x_miss <- x[idx_miss,]
  x_obs <- x[idx_obs,]

  y_new <- c(y_obs, rep(1, count_miss), rep(0, count_miss))
  mask <- is.na(c(y_obs, rep(NA, 2*count_miss)))
  x_new <- rbind(x_obs, x_miss, x_miss)

  df_x <- data.frame(x_new)
  x_vars <- colnames(df_x)

  data <- data.frame(y_new, df_x)
  formula <- as.formula(paste("y_new ~ ", paste(x_vars, collapse= "+")))

  q_vec <- c()

  for (i in 1:iter) {

    # E-STEP
    prob_obs <- obs_lik(y_obs, x_obs, beta, intercept)
    prob_y_miss_1 <- prob_y_xbeta(x_miss, beta, intercept) #P(y=1|X,beta)
    prob_y_miss_0 <- 1-prob_y_miss_1 #P(y=0|X,beta)
    prob <- c(prob_obs, prob_y_miss_1, prob_y_miss_0)
```

```
    weights <- c(rep(1,length(prob_obs)),prob_y_miss_1,prob_y_miss_0)

    # Expected Full Data Log Likelihood
    q <- sum(weights*log(prob))
    q_vec <- append(q_vec, q)

    # M-STEP
    glm_result <- glm(formula, family = binomial, data = data, weights = weights)
    beta <- if(intercept){
      coefficients(glm_result)
    }
    else{coefficients(glm_result)[-1]}
  }

  if (draw_plot){
    plot(1:iter, q_vec, xlab="Iteration",ylab="Expected Full Log-Likelihood",
         main = paste("Expectation Maximization for Logistic Regression", title, sep=' '))
  }
  return(coefficients(glm_result))
}
```

- Apply the EM algorithm for the synthetic data defined below. Compare the following estimators for β by computing the MSE: (i) without missing values (y, X), (ii) with missing values (y_{NA}, X) by using only the rows which do not contain missing values, (iii) with missing values (y_{NA}, X) by using the EM algorithm. Note that in (i) and (ii), you just have to use the function **glm**. Here, you can consider that the intercept is null. What do we notice for estimators (ii) and (iii) ?

```
library(mvtnorm)
set.seed(1)

d <- 3
beta_true <- c(0.1,0.5,0.9)

n <- 1000
mu <- rep(0,d)
Sigma <- diag(1,d)+matrix(0.5,d,d)

X <- rmvnorm(n, mean=mu, sigma=Sigma) #multivariate Gaussian variable
logit_link <- 1/(1+exp(-X%*%beta_true))
y <- (runif(n)<logit_link)*1

#### Introduction of MCAR values

nb_missingvalues <- 0.2*n*d
missing_idx <- sample(1:n,nb_missingvalues)
yna <- y
yna[missing_idx] <- NA
```

Solution:

```
idx_obs <- which(!is.na(yna))
y_no_miss_data <- yna[idx_obs]
x_no_miss_data <- X[idx_obs,]

# Case (i): without missing values (y,X)
glm_case_i <- glm(y ~ X, family = binomial, data = data.frame(y,X))
pred_case_i <- coefficients(glm_case_i)[-1]

## Case (ii): with missing values by using only the rows which do not contain missing values
glm_case_ii <- glm(y_no_miss_data ~ x_no_miss_data, family = binomial, data = data.frame(y_no_m
iss_data,x_no_miss_data))
pred_case_ii <- coefficients(glm_case_ii)[-1]

## Case (iii): with missing values by using the EM algorithm.
pred_case_iii <- em_log_reg(X, yna, draw_plot=TRUE, title='Case (iii)')[-1]
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

Comparison of Estimators and MSE:

```
cat("Estimators of Beta's for case i", pred_case_i, "\n")
```

```
## Estimators of Beta's for case i 0.1441317 0.4606494 1.037282
```

```
cat("Estimators of Beta's for case ii", pred_case_ii, "\n")
```

```
## Estimators of Beta's for case ii -0.0443445 0.5443778 1.099247
```

```
cat("Estimators of Beta's for case iii", pred_case_iii, "\n")
```

```
## Estimators of Beta's for case iii -0.04915227 0.5452184 1.087039
```

```
cat('\n\n')
```

```
MSE_log_reg <- function(beta_pred, intercept=FALSE){
  y_pred_probs <- prob_y_xbeta(X, beta_pred, intercept)
  y_pred <- ifelse(y_pred_probs>0.5,1,0)
  diff_squared <- (y - y_pred)^2
  mse <- sum(diff_squared)/length(diff_squared)
  return(mse)
}

MSE_log_reg_i <- MSE_log_reg(pred_case_i)
MSE_log_reg_ii <- MSE_log_reg(pred_case_ii)
MSE_log_reg_iii <- MSE_log_reg(pred_case_iii)

cat("MSE for case i: ", MSE_log_reg_i, "\n")
```

```
## MSE for case i:  0.235
```

```
cat("MSE for case ii: ", MSE_log_reg_ii, "\n")
```

```
## MSE for case ii:  0.245
```

```
cat("MSE for case iii: ", MSE_log_reg_iii, "\n")
```

```
## MSE for case iii:  0.245
```

Inference:

- The MSE is slightly higher for missing value cases: **ii** and **iii**.
- The estimators of β are exactly same for cases **ii** and **iii** and also they are close to those β's of Case **i**. This shows the effectiveness of **EM Algorithm** in estimating the parameters.
- Ideally the difference between MSE's when there is no missing data (Case i) and with missing data (Cases ii and iii) should be very high due to loss of information. But this phenomenon is not present very clearly. This is because the data is missing at random **(MCAR)**. Hence, if the missing data mechanism is **M(C)AR**, we can accurately estimate the model parameters and impute the missing values confidently with techniques like EM Algorithm.
- Apply the EM algorithm for the cancer prostate dataset (only quantitative variables) and compare the different estimators for β. Here, consider that the intercept is **not null**.

```
set.seed(1)

canpros <- read.table(file = 'cancerprostate.csv', header = T, sep = ";")
#head(canpros)
quanti_var <- c(1,2,6,7)
canpros <- canpros[,quanti_var] # we use only the quantitative variables.

n <- dim(canpros)[1]
y <- canpros$Y
#X <- cbind(rep(1,n),canpros$age,canpros$acide,canpros$log.acid)
# Removed the vector of 1 s since, glm function takes care of this. Since the intercept is impl
emented by glm by default.
X <- cbind(canpros$age,canpros$acide,canpros$log.acid)
d <- dim(X)[2]

#### Introduction of MCAR values

nb_missingvalues <- 0.2*n*d
missing_idx <- sample(1:n,nb_missingvalues)

yna <- y
yna[missing_idx] <- NA
```

Solution:

```
# Remove rows with NA values for case ii
idx_obs <- which(!is.na(yna))
y_no_miss_data <- yna[idx_obs]
x_no_miss_data <- X[idx_obs,]

# Predict beta's
# Case i
cancer_glm_case_i <- glm(y ~ X, family = binomial, data = data.frame(y,X))
result_cancer_i <- coefficients(cancer_glm_case_i)
beta_cancer_i <- result_cancer_i[-1]
intercept_cancer_i <- result_cancer_i[1]

# Case ii
cancer_glm_case_ii <- glm(y_no_miss_data ~ x_no_miss_data, family = binomial, data = data.frame
(y_no_miss_data,x_no_miss_data))
result_cancer_ii <- coefficients(cancer_glm_case_ii)
beta_cancer_ii <- result_cancer_ii[-1]
intercept_cancer_ii <- result_cancer_ii[1]

# Case iii
result_cancer_iii <- em_log_reg(X, yna, draw_plot= TRUE, intercept = TRUE, title='Case (iii)')
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```


```
beta_cancer_iii <- result_cancer_iii[-1]
intercept_cancer_iii <- result_cancer_iii[1]
```

```
cat("Beta's and intercept for Case i   ", beta_cancer_i, "\t", intercept_cancer_i, "\n")
```

```
## Beta's and intercept for Case i    -0.02805178 -9.964988 10.54332     12.347
```

```
cat("Beta's and intercept for Case ii  ", beta_cancer_ii, "\t", intercept_cancer_ii, "\n")
```

```
## Beta's and intercept for Case ii   -0.05941504 13.68663 -4.531646     -8.6132
```

```
cat("Beta's and intercept for Case iii ", beta_cancer_iii, "\t", intercept_cancer_iii, "\n")
```

```
## Beta's and intercept for Case iii  -0.05940975 13.68564 -4.531123     -8.612616
```

```
cat('\n\n')
```

```
MSE_log_reg_i <- MSE_log_reg(result_cancer_i, intercept = TRUE)
MSE_log_reg_ii <- MSE_log_reg(result_cancer_ii, intercept = TRUE)
MSE_log_reg_iii <- MSE_log_reg(result_cancer_iii, intercept = TRUE)

cat("MSE for case i: ", MSE_log_reg_i, "\n")
```

```
## MSE for case i:  0.2830189
```

```
cat("MSE for case ii: ", MSE_log_reg_ii, "\n")
```

```
## MSE for case ii:  0.3584906
```

```
cat("MSE for case iii: ", MSE_log_reg_iii, "\n")
```

```
## MSE for case iii:  0.3584906
```

Inference: (Same as before exercise)

- The MSE is slightly higher for missing value cases: **ii** and **iii**.
- The estimators of β are exactly same for cases **ii** and **iii** and also they are close to those β's of Case **i**. This shows the effectiveness of **EM Algorithm** in estimating the parameters.
- Ideally the difference between MSE's when there is no missing data (Case i) and with missing data (Cases ii and iii) should be very high due to loss of information. But this phenomenon is not present very clearly. This is because the data is missing at random **(MCAR)**. Hence, if the missing data mechanism is **M(C)AR**, we can accurately estimate the model parameters and impute the missing values confidently with techniques like EM Algorithm.
- Briefly discuss the link with semi-supervised learning.

Solution:

Missing data problem and semi-supervised learning are related in the sense that both involve the use of incomplete data sets for learning.

In semi-supervised learning, a model is trained on a dataset that contains both labeled and unlabeled data, where the goal is to use the information from the labeled data to make predictions for the unlabeled data. The performance of a semi-supervised learning model is severely if the missing data mechanism is: not missing at random (NMAR).

Missing data can also be viewed as a form of weak supervision, where the missing values can be thought of as unlabeled data. Techniques such as Expectation-Maximization (EM) algorithm and multiple imputation can be used to fill in the missing data and improve the performance of a semi-supervised learning model.

Assumptions:

- All the classes are balanced.
- The marginal distributions of the features and the labels are identical in the labeled and unlabeled dataset.

Classical Approach:

- Learning with only labeled data.
  - The estimators are unbiased iff the missing data pattern is MCAR.
- Learning with both labeled and unlabled data.
  - The estimators are biased even if the missing data pattern is MCAR.