# Project 2 EM Algorithm

In this problem, you will implement the EM algorithm for the coin toss problem in R.

Below we provide you with a skeleton of the algorithm. You can either fill this skeleton with the required functions or write your own version of the EM algorithm. If you choose to do the latter, please also present your results using Rmarkdown in a clear fashion.

```r
suppressPackageStartupMessages({
  library(data.table)
  library(pheatmap)
  library(viridis)
})
set.seed(2023)
```

## (a) Load data

We first read the data stored in the file "coinflip.csv".

```r
# read the data into D
D <- read.csv("coinflip.csv")
# check the dimension of D
all(dim(D) == c(200, 100))
```

```
## [1] TRUE
```

## (b) Initialize parameters

Next, we will need to initialize the mixture weights and the probabilities of obtaining heads. You can choose your own values as long as they make sense.

```r
# Number of coins
k <- 2
# Mixture weights (a vector of length k)
lambda <- c(0.5, 0.5)
# Probabilities of obtaining heads (a vector of length k)
theta <- runif(2)
```

## (c) The EM algorithm

Now we try to implement the EM algorithm. Please write your code in the indicated blocks.

```r
##' This function implements the EM algorithm for the coin toss problem
##' @param D Data matrix of dimensions 100-by-N, where N is the number of observations
##' @param k Number of coins
##' @param lambda Vector of mixture weights
##' @param theta Vector of probabilities of obtaining heads
##' @param tolerance A threshold used to check convergence
coin_EM <- function(D, k, lambda, theta, tolerance = 1e-2) {

  # expected complete-data (hidden) log-likelihood
  ll_hid <- -Inf
  # observed log-likelihood
  ll_obs <- -Inf
  # difference between two iterations
  diff <- Inf
  # number of observations
  N <- nrow(D)
  # responsibilities
  gamma <- matrix(0, nrow = k, ncol = N)

  # run the E-step and M-step until convergence
  while (diff > tolerance) {

    # store old likelihood
    ll_obs_old <- ll_obs

    ############# E-step #############

    ### YOUR CODE STARTS ###

    # Compute the responsibilities
    # Define the probability of data given theta
    prob_x_theta <- rbind(theta[1] ** (rowSums(D == 1)) + (1 - theta[1]) ** (rowSums(D==0)),
                          theta[2] ** (rowSums(D == 1)) + (1 - theta[2]) ** (rowSums(D==0)))

    # Find the respective responsibility vector
    lambda_prob_prod <- data.table(rbind(lambda[1] * prob_x_theta[1,],
                                         lambda[2] * prob_x_theta[2,]))
    gamma <- lambda_prob_prod[, lapply(.SD, function(x) x/sum(x))]

    # Update expected complete-data (hidden) log-likelihood
    # For binary cases the log likelihood is also following the same
    ll_hid <- sum(gamma * log(lambda_prob_prod))

    # Update observed log-likelihood
    ll_obs <- ll_hid

    # Recompute difference between two iterations
    diff <- ll_obs - ll_obs_old

    ### YOUR CODE ENDS ###

    ############# M-step #############
    ### YOUR CODE STARTS ###

    # Recompute priors (mixture weights)
    lambda <- rowMeans(gamma)

    # Recompute probability of heads for each coin
    theta <- c(sum(gamma[1,] * rowSums(D == 1))/(100*sum(gamma[1,])),
               sum(gamma[2,] * rowSums(D == 1))/(100*sum(gamma[2,])))

    ### YOUR CODE ENDS ###

  }

  return(list(ll_hid = ll_hid, ll_obs = ll_obs, lambda = lambda, theta = theta, gamma = gamma))

}
```

Run the EM algorithm:

```
res <- coin_EM(D, k, lambda, theta)
```

## (d) Results

Probability of heads:

```
## YOUR CODE ##
cat(sprintf(
  "The probability of heads are:
  coin 1: %.3f
  coin 2: %.3f",
  res$theta[1], res$theta[2]))
```

```
## The probability of heads are:
##   coin 1: 0.482
##   coin 2: 0.379
```

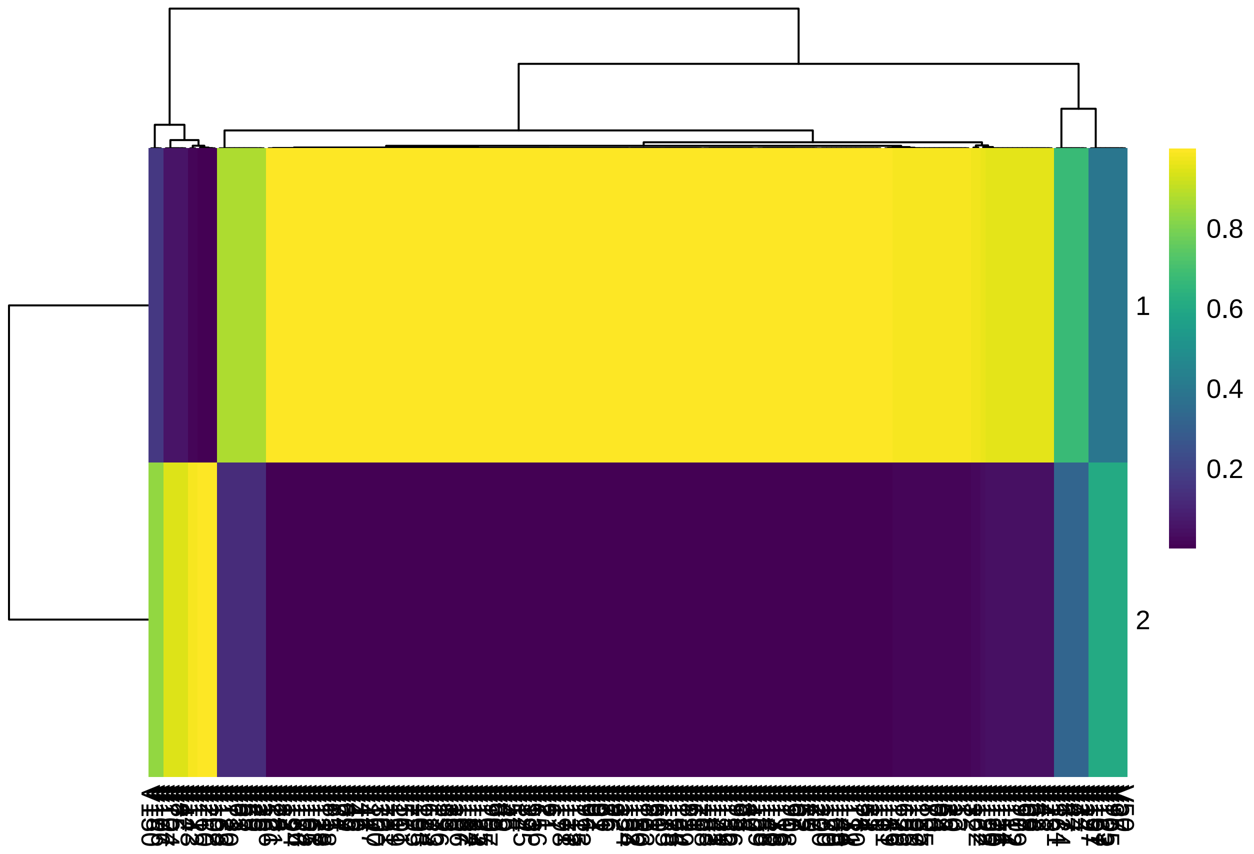
Mixture weights:

```
## YOUR CODE ##
cat(sprintf(
  "The mixture weights are:
  coin 1 : %.3f
  coin 2 : %.3f ",
  res$lambda[1], res$lambda[2]
))
```

```
## The mixture weights are:
##   coin 1 : 0.886
##   coin 2 : 0.114
```

Heatmap of responsibilities:

```r
## YOUR CODE ##
pheatmap::pheatmap(res$gamma, color = viridis_pal(option = "D")(100))
```



How many observations belong to each coin?

```r
## YOUR CODE ##
cat("The number of observation for each coin is predicted below: \n")
```

```
## The number of observation for each coin is predicted below:
```

```r
knitr::kable(table(as.numeric(res$gamma[, lapply(.SD, which.max)])))
```

| Var1 | Freq |
|---|---|
| 1 | 178 |
| 2 | 22 |