1.1

Project 2 EM Algorithm

Exercise 1

Bayes Theorem

$$P(C_i|D_j) = \frac{P(D_j|C_i)P(C_i)}{P(D_j)}$$

Conditional $P(D_j|C_i)$:

$$P(D_1|C_A) = 0.7^8 \cdot 0.3^2$$

$$P(D_2|C_A) = 0.3^7 \cdot 0.7^3$$

$$P(D_1|C_B) = 0.4^8 \cdot 0.6^2$$

$$P(D_2|C_B) = 0.6^7 \cdot 0.4^3$$

Marginal of D_j:

$$P(D_j) = \sum_{i \in A,B} P(D_j|C_i) \cdot P(C_i)$$

$$P(D_1) = \sum_{i \in A,B} P(D_1|C_i) \cdot P(C_i) = P(D_1|C_A) \cdot P(C_A) + P(D_1|C_B) \cdot P(C_B)$$

$$P(D_2) = \sum_{i \in A,B} P(D_2|C_i) \cdot P(C_i) = P(D_2|C_A) \cdot P(C_A) + P(D_2|C_B) \cdot P(C_B)$$

```
p_D1 <- 0.7^8 * 0.3^2 * 0.6 + 0.4^8 * 0.6^2 * 0.4
p_D2 <- 0.3^7 * 0.7^3 * 0.6 + 0.6^7 * 0.4^3 * 0.4

p_D1
```

```
## [1] 0.003207364
```

```
p_D2
```

```
## [1] 0.0007616446
```

```
p_CA_D1 <- 0.7^8 * 0.3^2 * 0.6 / p_D1
p_CB_D1 <- 0.4^8 * 0.6^2 * 0.4 / p_D1

p_CA_D2 <- 0.3^7 * 0.7^3 * 0.6 / p_D2
p_CB_D2 <- 0.6^7 * 0.4^3 * 0.4 / p_D2
```

```
p_CA_D1
```

```
## [1] 0.9705765
```

```
p_CB_D1
```

```
## [1] 0.02942349
```

```
p_CA_D2
```

```
## [1] 0.05909378
```

```
p_CB_D2
```

```
## [1] 0.9409062
```

The mixture weights are the averaged posteriors:

```
lambda_A <- (p_CA_D1 + p_CA_D2)/2
lambda_B <- (p_CB_D1 + p_CB_D2)/2

lambda_A
```

```
## [1] 0.5148351
```

```
lambda_B
```

```
## [1] 0.4851649
```

Exercise 2

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.

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
set.seed(42)
```

(a) Load data

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

```
# read the data into D
D <- read.csv("coinflip.csv")
D <- as.matrix(D)
```

```r
# 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 <- rep(1, k) / k
# Probabilities of obtaining heads (a vector of length k)
theta <- runif(k)
```

(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)
  # keep track of lambda and theta during the optimisation
  lambda_all <- lambda
  theta_all <- theta
  # iteration number
  t <- 1

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

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

    ### YOUR CODE STARTS ###

    # Compute the probabilities

    # Compute the likelihoods
```

```r
  P_D_given_C <- apply(D, 1,
                function(x) theta ** sum(x==1) * (1-theta)** sum(x==0))

  # Compute the responsibilities
  gamma <- lambda * P_D_given_C
  gamma <- apply(gamma, 2, function (x) x / sum(x))

  # Update expected complete-data (hidden) log-likelihood
  log_P_D_given_C <- log(P_D_given_C)
  ll_hid_new <-sum(gamma * log(lambda) + gamma * log_P_D_given_C)

  ll_hid <- c(ll_hid, ll_hid_new) # keep track of this quantity

  # Update observed log-likelihood
  #
  ll_obs_new <- sum(log(colSums(lambda * P_D_given_C)))
  ll_obs <- c(ll_obs, ll_obs_new) # keep track of this quantity

  # Recompute difference between two iterations
  diff <- abs(ll_obs[t] - ll_obs[t+1])

  ### YOUR CODE ENDS ###

  ############# M-step ############

  ### YOUR CODE STARTS ###

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

  lambda_all <- rbind(lambda_all, lambda) # keep track of this quantity

  # Recompute probability of heads for each coin
      # Expected count of coin for heads and tails
        # A
  c_A_h <- sum(gamma[1, ] %*% D)
  c_A_t <- sum(gamma[1, ] %*% !D)

        # B
  c_B_h <- sum(gamma[2, ] %*% D)
  c_B_t <- sum(gamma[2, ] %*% !D)

  theta_A <- c_A_h / (c_A_h + c_A_t)
  theta_B <- c_B_h / (c_B_h + c_B_t)

  theta <- c(theta_A, theta_B)

  theta_all <- rbind(theta_all, theta) # keep track of this quantity

  ### YOUR CODE ENDS ###
  t <- t+1

}
```

```
  print(t-1)

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

}
```

Run the EM algorithm:

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

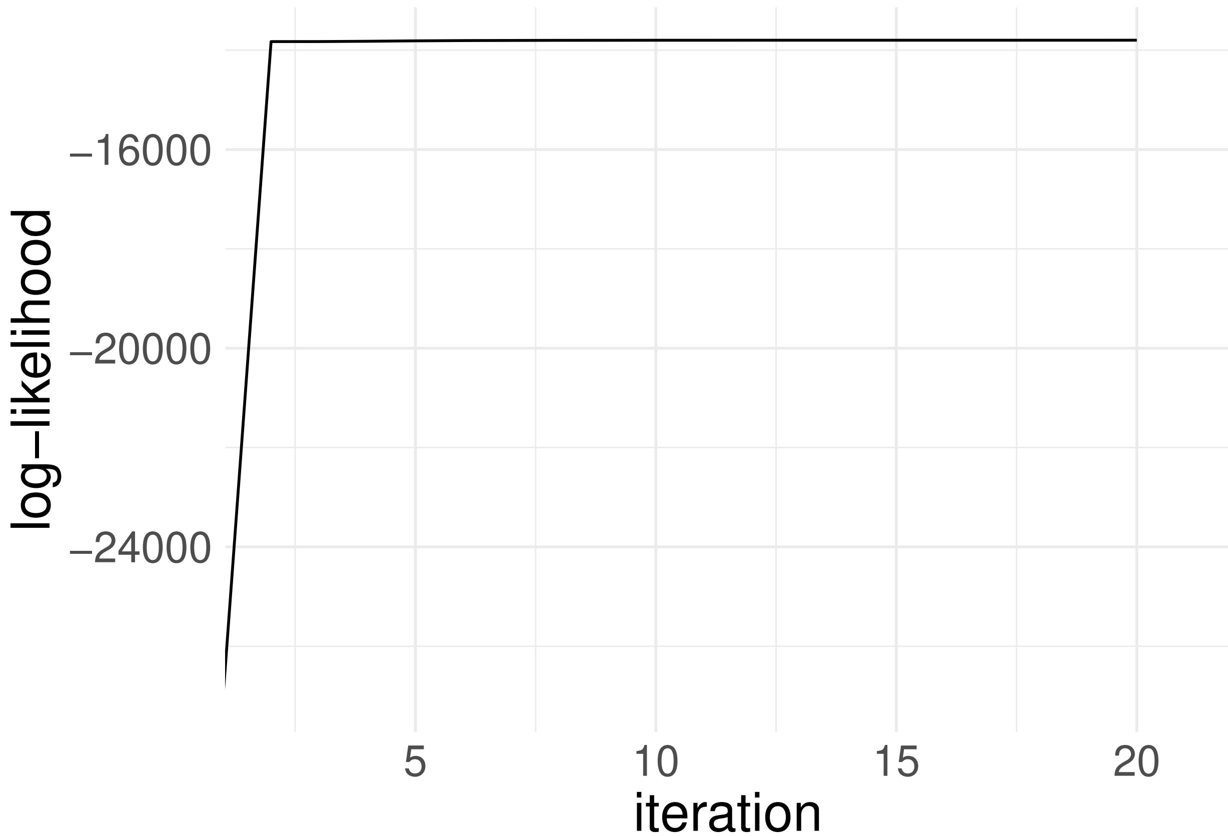
```
## [1] 20
```

Log-likelihood through iterations

Log-likelihood rapidly increased after the first iteration. Over the remaining iterations it converged to the value of around -13800.

```
data <- data.frame(iterations=1:(length(res$ll_obs)-1),
  ll_obs=res$ll_obs[2:length(res$ll_obs)]
)

ggplot(data, aes(x=iterations, y=ll_obs)) +
  geom_line() +
  labs(x="iteration", y="log-likelihood") +
  theme_minimal() +
  coord_cartesian(xlim=c(2, length(res$ll_obs))) +
  theme(text = element_text(size=20))
```



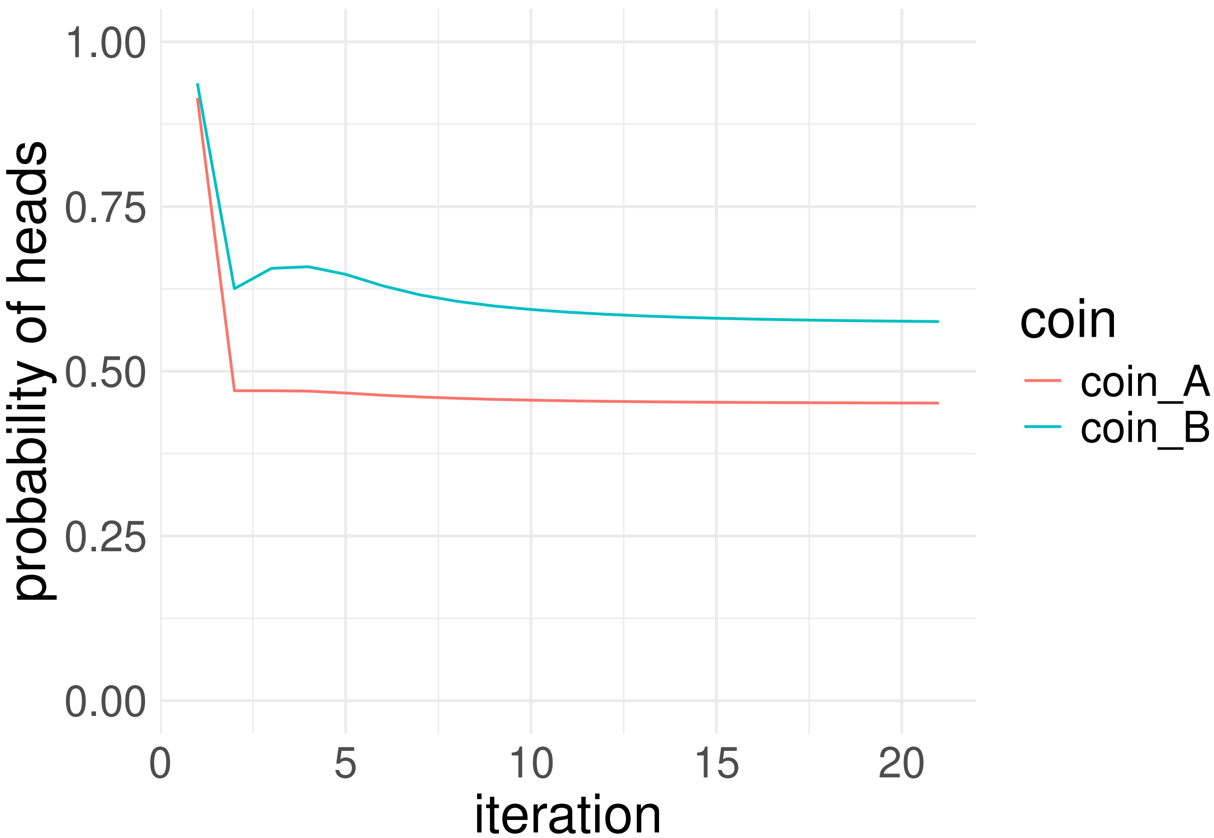
Probability of heads through iterations

The probability of heads for both coins fastly decreased over the first iterations (up to around 10). After 10th iteration probability of coin A to turn heads converged to be around 0.45 and of coin B - 0.58.

```
theta_data <- data.frame(iterations=1:(length(res$theta_all[,1])),
  coin_A=res$theta_all[,1], coin_B=res$theta_all[,2]
)

theta_data <- theta_data %>% pivot_longer(cols=c('coin_A', 'coin_B'),
                    names_to='coin',
                    values_to='theta')

ggplot(theta_data, aes(x=iterations, y=theta, color=coin)) +
  geom_line(linetype="solid") +
  labs(x="iteration", y="probability of heads") +
  theme_minimal() +
  theme(text=element_text(size=20)) +
  ylim(0, 1.0)
```

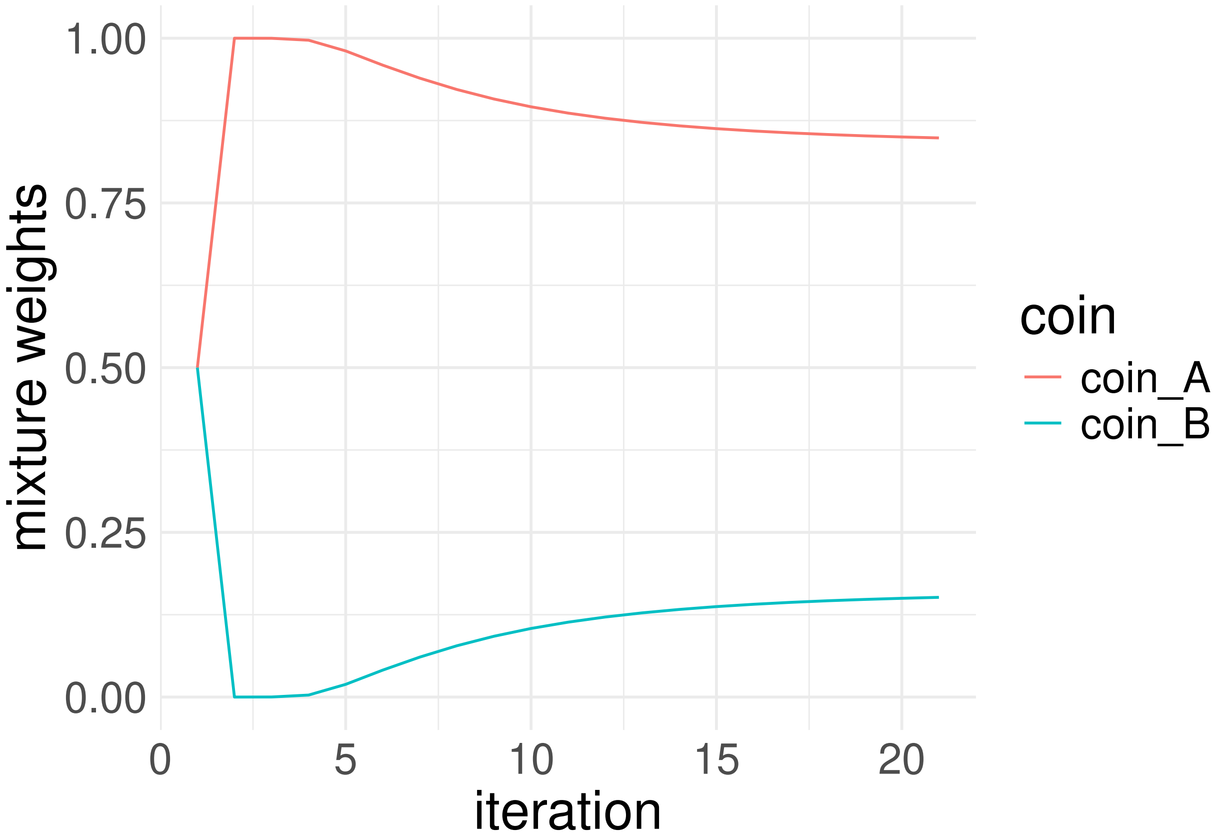


Mixture weights through iterations

The weight of coin A increased in the first iterations to be close to 1. In the next 10 iterations the weight of coin A converged to around 0.85 (weight of coin B converged symmetricaly to around 0.15).

```
lambda_data <- data.frame(iterations=1:(length(res$lambda_all[,1])),
  coin_A=res$lambda_all[,1], coin_B=res$lambda_all[,2]
)
```

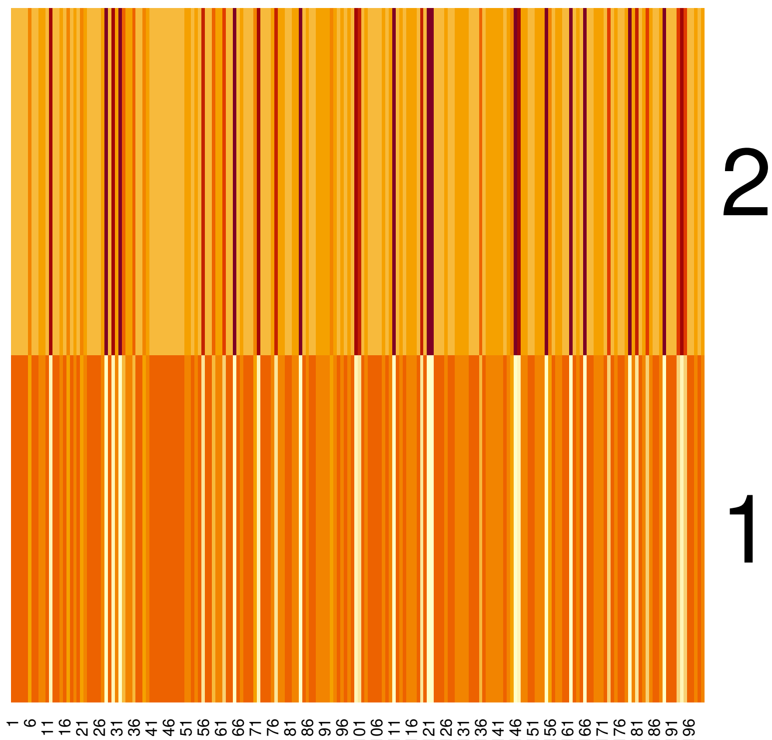
```r
lambda_data <- lambda_data %>% pivot_longer(cols=c('coin_A', 'coin_B'),
                    names_to='coin',
                    values_to='lambda')
ggplot(lambda_data, aes(x=iterations, y=lambda, color=coin)) +
  geom_line(linetype="solid") +
  labs(x="iteration", y="mixture weights") +
  theme_minimal() +
  theme(text=element_text(size=20))
```



Overall, it took up to 20 iterations for the algorithm to converge to the end state.

Heatmap of responsibilities at final iteration:

```r
heatmap(res$gamma, Colv=NA, Rowv=NA)
```



(d) Results

How many observations belong to each coin? There are two ways to calculate - we can decide for each toss round if the coin is A or B (corresponds to the `num_obs` variable).

```
num_obs <- rowSums(round(res$gamma))
print(paste("Number of observations belonging to coin A: ", num_obs[1]))
```

```
## [1] "Number of observations belonging to coin A:  171"
```

```
print(paste("Number of observations belonging to coin B: ", num_obs[2]))
```

```
## [1] "Number of observations belonging to coin B:  29"
```

Index of comments

1.1 very nice!