Project 2 EM Algorithm

Team C - Minghang Li, Xiaocheng Yang, Xinyi Chen

March 04, 2024

```
library(ggplot2)
library(reshape2)
```

Problem 4: Responsibilities and prior for biased coins

In this problem, we are basically performing EM algorithm *manually*.

The responsibility of component i (coin i) can be calculated by:

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

Given mixture weights λ and model parameter (the probability of obtaining heads for coins) θ, the expected hidden log likelihood is:

$$E(\ell_{\text{hid}}(\theta, \lambda)) = \sum_{i,k} \gamma_{ik} \log \lambda_k + \sum_{i,k} \gamma_{ik} \log P(X^{(i)}|\theta_k)$$

Since we're only updating λ (according to the description), we only need to maximize the "left sum". And the optimal model mixture weights $\hat{\lambda}_k$ is given by

$$\hat{\lambda}_k = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}$$

Re-write into the notation of this problem, we should update the mixture weights by:

$$\hat{P}(C_A) = \frac{\gamma_{A1} + \gamma_{A2}}{2}$$

and

$$\hat{P}(C_B) = \frac{\gamma_{B1} + \gamma_{B2}}{2}$$

```
P_head_CA <- 0.7
P_head_CB <- 0.4
prior_CA <- 0.6
prior_CB <- 0.4
```

```r
D1 <- c(1, 1, 0, 1, 1, 1, 0, 1, 1, 1)
D2 <- c(0, 0, 1, 0, 0, 0, 1, 1, 0, 0)

# get num of heads
num_heads_D1 <- sum(D1)
num_tails_D1 <- length(D1) - num_heads_D1
num_heads_D2 <- sum(D2)
num_tails_D2 <- length(D2) - num_heads_D2
```

The responsibilites are calcualted below:

```r
jointP_CA_D1 <- prior_CA * P_head_CA^num_heads_D1 * (1 - P_head_CA)^num_tails_D1
jointP_CB_D1 <- prior_CB * P_head_CB^num_heads_D1 * (1 - P_head_CB)^num_tails_D1
```

```r
gamma_A1 <- jointP_CA_D1 / (jointP_CA_D1 + jointP_CB_D1)
gamma_A1
```

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

$$\begin{aligned}\gamma_{A1} &= \frac{P(C_A)P(D_1|C_A)}{P(C_A)P(D_1|C_A) + P(C_B)P(D_1|C_B)} \\ &= \frac{0.6 \cdot (0.7^8 \cdot 0.3^2)}{0.6 \cdot (0.7^8 \cdot 0.3^2) + 0.4 \cdot (0.4^8 \cdot 0.6^2)} \\ &= 0.9705765\end{aligned}$$

```r
gamma_B1 <- jointP_CB_D1 / (jointP_CA_D1 + jointP_CB_D1)
gamma_B1
```

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

$$\gamma_{B1} = 1 - \gamma_{A1} = 0.02942349$$

```r
jointP_CA_D2 <- prior_CA * P_head_CA^num_heads_D2 * (1 - P_head_CA)^num_tails_D2
jointP_CB_D2 <- prior_CB * P_head_CB^num_heads_D2 * (1 - P_head_CB)^num_tails_D2
```

```r
gamma_A2 <- jointP_CA_D2 / (jointP_CA_D2 + jointP_CB_D2)
gamma_A2
```

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

$$\begin{aligned}\gamma_{A2} &= \frac{P(C_A)P(D_2|C_A)}{P(C_A)P(D_2|C_A) + P(C_B)P(D_2|C_B)} \\ &= \frac{0.6 \cdot (0.7^3 \cdot 0.3^7)}{0.6 \cdot (0.7^3 \cdot 0.3^7) + 0.4 \cdot (0.4^3 \cdot 0.6^7)} \\ &= 0.05909378\end{aligned}$$

```r
gamma_B2 <- jointP_CB_D2 / (jointP_CA_D2 + jointP_CB_D2)
gamma_B2
```

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

$$\gamma_{B2} = 1 - \gamma_{A2} = 0.9409062$$

```
(gamma_A1 + gamma_A2) / 2
```

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

The updated $P(C_A)$ would be

$$\begin{aligned}\hat{P}(C_A) &= \frac{\gamma_{A1} + \gamma_{A2}}{2} \\ &\approx 0.515\end{aligned}$$

```
(gamma_B1 + gamma_B2) / 2
```

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

$$\begin{aligned}\hat{P}(C_B) &= \frac{\gamma_{B1} + \gamma_{B2}}{2} \\ &\approx 0.485\end{aligned}$$

Problem 5: Learning a mixture model for two biased coins

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.

```
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)
# 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.

```
# 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)
```

```
theta
```

```
## [1] 0.9148060 0.9370754
```

(c) The EM algorithm

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

```
##' 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 responsibilities
    L <- dim(D)[2]

    likelihoods <- rbind(theta[1]^rowSums(D) * (1-theta[1])^(L - rowSums(D)),
                         theta[2]^rowSums(D) * (1-theta[2])^(L - rowSums(D)))

    joint_1 <- lambda[1] * likelihoods[1, ]
    joint_2 <- lambda[2] * likelihoods[2, ]

    gamma[1, ] <- joint_1 / (joint_1 + joint_2)
    gamma[2, ] <- joint_2 / (joint_1 + joint_2)


    # Update expected complete-data (hidden) log-likelihood
    ll_hid_new <- sum(gamma * log(lambda * likelihoods))
    ll_hid <- c(ll_hid, ll_hid_new) # keep track of this quantity

    # Update observed log-likelihood
    ll_obs_new <- sum(log(colSums(lambda * likelihoods)))
    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 <- rowSums(gamma) / N

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

    # Recompute probability of heads for each coin
    heads <- rowSums(D)
    count_heads <- gamma %*% heads
    count_tot <- rowSums(gamma) * L

    theta <- t(count_heads / count_tot)

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

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

  }

  return(list(ll_hid = ll_hid,
              ll_obs = ll_obs,
              lambda = lambda,
              theta = theta,
              gamma = gamma,
              lambda_all = lambda_all,
              theta_all = theta_all))
}
```

Run the EM algorithm:

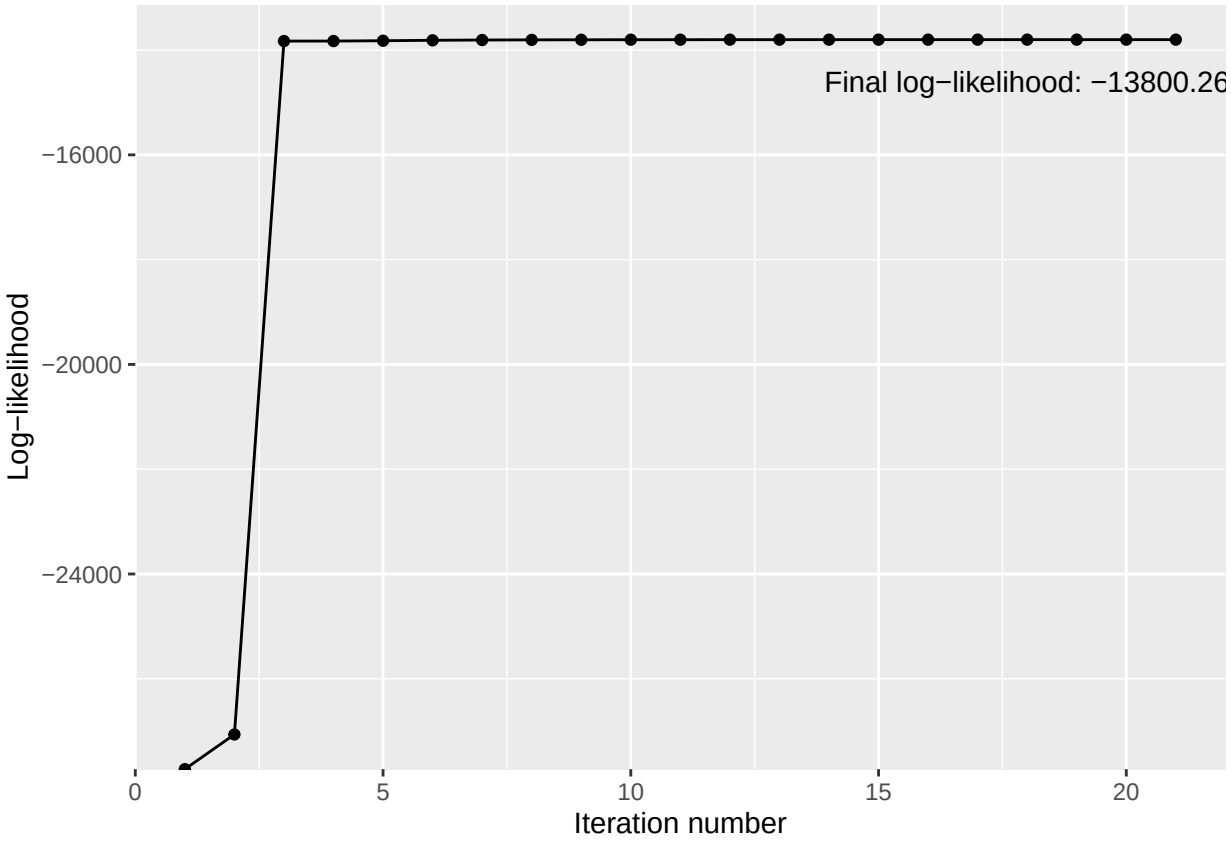
```
res <- coin_EM(D, k, lambda, theta)
```

(d) Results

Log-likelihood through iterations:

```
## YOUR CODE ##
ggplot(data = data.frame(iter = 1:length(res$ll_obs),
                         ll = res$ll_obs), aes(x = iter, y = ll)) +
  geom_point() +
  geom_line() +
  xlab("Iteration number") +
  ylab("Log-likelihood") +
  annotate("text",
           x = length(res$ll_obs) - 3,
```

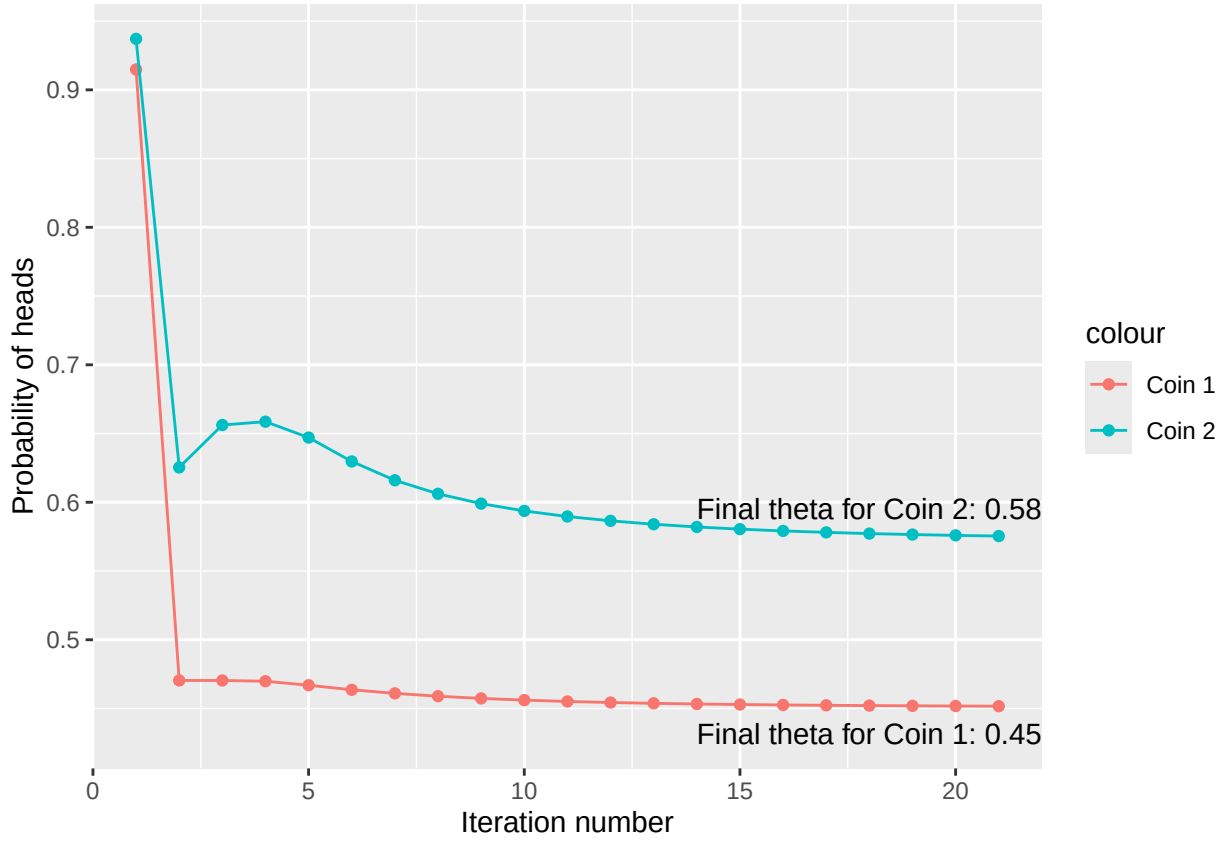
```
          y = res$ll_obs[length(res$ll_obs)] - 800,
          label = paste("Final log-likelihood:",
                        round(res$ll_obs[length(res$ll_obs)], 2)))
```



Probability of heads through iterations:

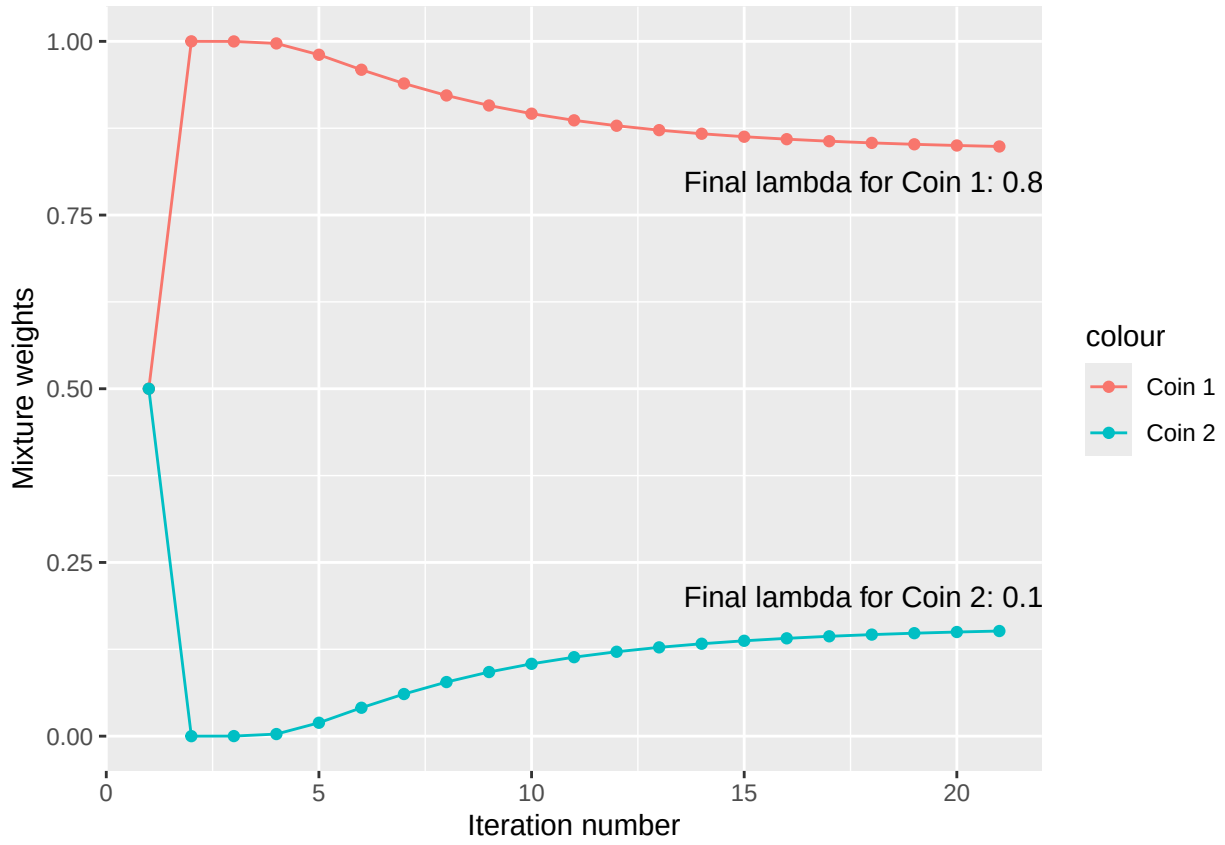
```
## YOUR CODE ##
# plot res$theta_all as a function of iteration number
ggplot(data = data.frame(iter = 1:length(res$theta_all[,1]),
                         theta1 = res$theta_all[, 1],
                         theta2 = res$theta_all[, 2]), aes(x = iter)) +
  geom_point(aes(y = theta1, color = "Coin 1")) +
  geom_line(aes(y = theta1, color = "Coin 1")) +
  geom_point(aes(y = theta2, color = "Coin 2")) +
  geom_line(aes(y = theta2, color = "Coin 2")) +
  xlab("Iteration number") +
  ylab("Probability of heads") +
  annotate("text",
           x = length(res$theta_all[,1]) - 3,
           y = res$theta_all[length(res$theta_all[,1]), 1] - 0.02,
           label = paste("Final theta for Coin 1:",
                         round(res$theta_all[length(res$theta_all[,1]), 1], 2))) +
  annotate("text",
           x = length(res$theta_all[,1]) - 3,
           y = res$theta_all[length(res$theta_all[,1]), 2] + 0.02,
           label = paste("Final theta for Coin 2:",
```

```
                        round(res$theta_all[length(res$theta_all[,1]), 2], 2)))
```



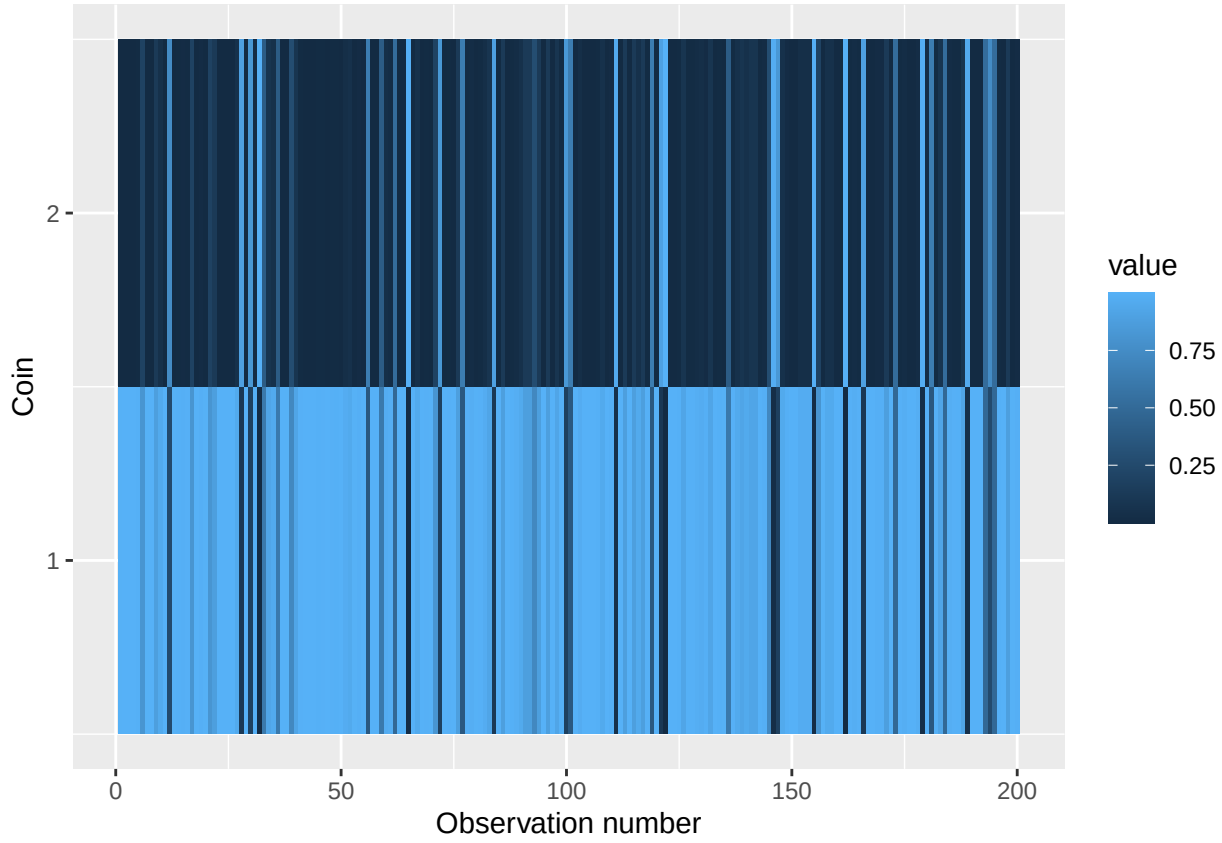
Mixture weights through iterations:

```
## YOUR CODE ##
# plot res$lambda_all as a function of iteration number
ggplot(data = data.frame(iter = 1:length(res$lambda_all[,1]),
                         lambda1 = res$lambda_all[, 1],
                         lambda2 = res$lambda_all[, 2]), aes(x = iter)) +
  geom_point(aes(y = lambda1, color = "Coin 1")) +
  geom_line(aes(y = lambda1, color = "Coin 1")) +
  geom_point(aes(y = lambda2, color = "Coin 2")) +
  geom_line(aes(y = lambda2, color = "Coin 2")) +
  xlab("Iteration number") +
  ylab("Mixture weights") +
  annotate("text",
           x = length(res$lambda_all[,1]) - 3,
           y = res$lambda_all[length(res$lambda_all[,1]), 1] - 0.05,
           label = paste("Final lambda for Coin 1:",
                         round(res$lambda_all[length(res$lambda_all[,1]), 1], 2))) +
  annotate("text",
           x = length(res$lambda_all[,1]) - 3,
           y = res$lambda_all[length(res$lambda_all[,1]), 2] + 0.05,
           label = paste("Final lambda for Coin 2:",
                         round(res$lambda_all[length(res$lambda_all[,1]), 2], 2)))
```



Heatmap of responsibilities at final iteration:

```
## YOUR CODE ##
df_gamma <- melt(res$gamma)
ggplot(data = df_gamma, aes(x = Var2, y = Var1, fill = value)) +
  geom_tile() +
  xlab("Observation number") +
  ylab("Coin") +
  scale_y_continuous(breaks = seq(0, 2, 1))
```



How many observations belong to each coin?

Coin 1:

```
## YOUR CODE ##
coin_1 <- sum(res$gamma[1, ] > res$gamma[2, ])
coin_1
```

```
## [1] 171
```

Coin 2:

```
coin_2 <- sum(res$gamma[2, ] > res$gamma[1, ])
coin_2
```

```
## [1] 29
```