HW2

Philipp Ross

2017-01-17

Last updated: 2017-01-19

Code version: 06a095c

A. Likelihood Inference

Exercise 1

For this problem, my intuition says that I should believe he rolled the blue die. To state this formally we have our data represented by which numbers were actually rolled and our models represented by the probability of rolling those numbers given the blue or green die.

We can represent our data as a vector of frequencies for having rolled the numbers one through six and our models as probability vectors for rolling those numbers based on the sides of each die.

```
x <- c(1, 2, 6, 1, 0, 0)  # data
fB <- c(1/6, 1/6, 1/2, 1/6, 0, 0)  # blue dice model
fG <- c(1/6, 1/3, 1/6, 1/3, 0, 0)  # green dice model
L <- function(f, x) {
    prod(f^x * (1 - f)^(1 - x))
}  # likelihood function
LR <- L(fB, x)/L(fG, x)  # likelihood ratio
print(LR)
```

```
[1] 937.5
```

We can see that our intuition was right! The chances of having rolled the blue die are 937.5 times more likely than having rolled the green die to generate those data.

Exercise 2

(a)

Below is the code for running simulations:

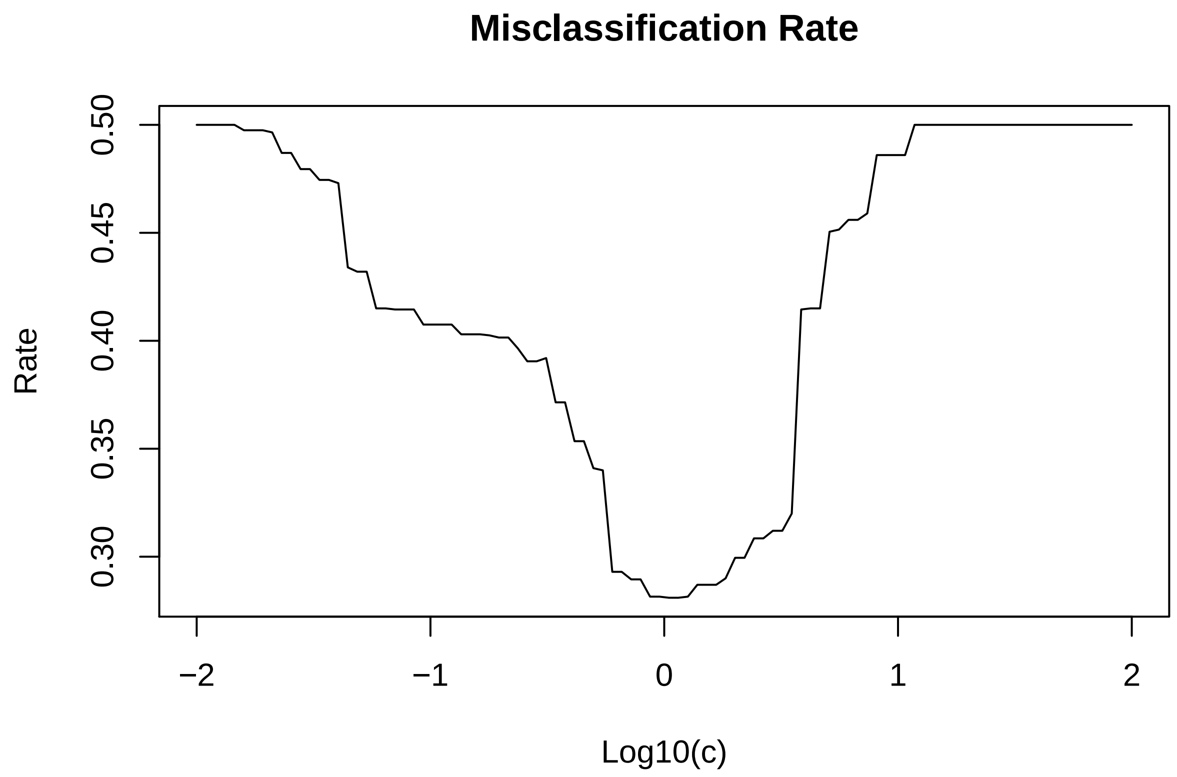
```
#' Simluate tusks
#'
#' Simulates genotype data of elephant tusks using a multinomial model
#'
#' @param nS number of savanna tusks to use
#' @param nF number of forest tusks to use
#' @param fS allele frequencies of savanna elephants
#' @param fF allele frequencies of forest elephants
#'
#' @return list of simulated tusks
#'
```

```r
#' @example simulate_tusks(1000, 1000, c(0.40,0.12,0.21,0.12,0.02,0.32), c(0.8,0.2,0.11,0.17,0.23,0.25)
#'
#'
simulate_tusks <- function(nS, nF, fS, fF) {

    sS <- replicate(nS, sapply(fS, function(x) {
        sample(c(1, 0), 1, prob = c(x, 1 - x))
    }))
    sF <- replicate(nF, sapply(fF, function(x) {
        sample(c(1, 0), 1, prob = c(x, 1 - x))
    }))

    return(list(S = sS, F = sF))

}

#' Compute misclassification rates
#'
#' Using a vector of log10 threshold values, calculate the rate of misclassification
#'
#' @param lr likelihood ratios computed for tusks
#' @param lc vector of log10(c) values of LR cutoff to us
#'
#' @return list of lists of simulated tusks
#'
#' @example compute_misclassification_rate(data, seq(-2,2,length=100))
#'
#'
compute_misclassification_rate <- function(lr, lc) {

    l1 <- unlist(lapply(lc, function(c) {
        sum(log10(lr$S) <= c)
    }))  # incorrectly classified as Savanna elephants
    l2 <- unlist(lapply(lc, function(c) {
        sum(log10(lr$F) > c)
    }))  # incorrectly classified as Forest elephants

    rates <- (l1 + l2)/(length(lr$S) + length(lr$F))

    return(rates)

}

#' Run Simulation
#'
#' Simulates data from forest and savanna tusks from prespecified frequencies, and uses LR to classify
#'
#' @param nS number of savanna tusks to use
#' @param nF number of forest tusks to use
#' @param fS allele frequencies of savanna elephants
#' @param fF allele frequencies of forest elephants
#' @param lc vector of log(c) values of LR cutoff to use
#'
```

```r
#' @return Miscalculation rates based on cutoffs
#'
#' @example run_simulation(1000, 1000, c(0.40, 0.12,0.21,0.12,0.02,0.32), c(0.8,0.2,0.11,0.17,0.23,0.25
#'
#'
run_simulation <- function(nS, nF, fS, fF, lc) {

    # simulate tusks
    data <- simulate_tusks(nS, nF, fS, fF)

    # compute likelihood ratios
    LRS <- apply(data$S, 2, function(x) {
        L(fS, x)
    })/apply(data$S, 2, function(x) {
        L(fF, x)
    })
    LRF <- apply(data$F, 2, function(x) {
        L(fS, x)
    })/apply(data$F, 2, function(x) {
        L(fF, x)
    })

    # compute misclassification rate
    rates <- compute_misclassification_rate(lr = list(S = LRS, F = LRF), lc)

    return(rates)
}
```

When we run the simulation for 1000 Savanna elephant tusks and 1000 Forest elephant tusks:

```r
nS <- 1000
nF <- 1000
fS <- c(0.4, 0.12, 0.21, 0.12, 0.02, 0.32)
fF <- c(0.8, 0.2, 0.11, 0.17, 0.23, 0.25)
lc <- seq(-2, 2, length = 100)

results <- run_simulation(nS, nF, fS, fF, lc)
plot(seq(-2, 2, length = 100), results, type = "l", main = "Misclassification Rate",
    xlab = "Log10(c)", ylab = "Rate")
```

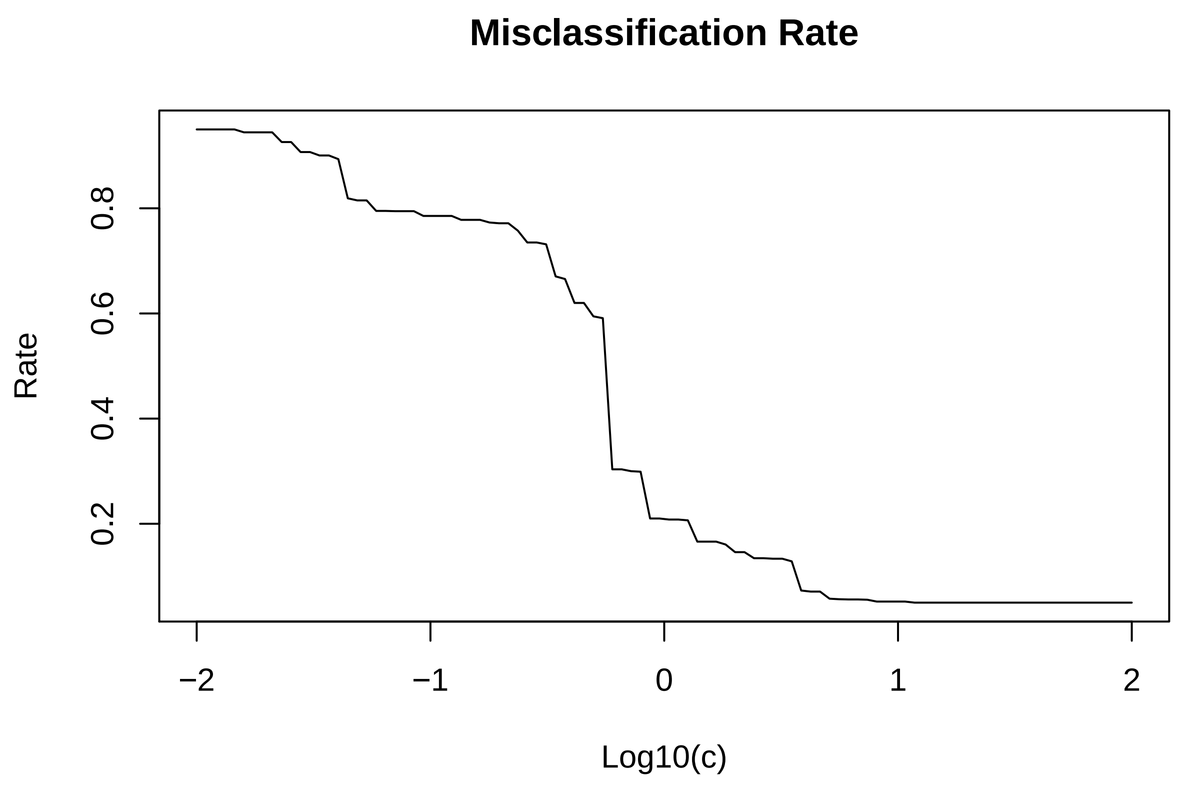


We can see that the value of c that minimizes our misclassification rate is 1.0476158.

(b)

When we run the simulation for 100 Savanna elephant tusks and 1900 Forest elephant tusks:

```r
nS <- 100
nF <- 1900
fS <- c(0.4, 0.12, 0.21, 0.12, 0.02, 0.32)
fF <- c(0.8, 0.2, 0.11, 0.17, 0.23, 0.25)
lc <- seq(-2, 2, length = 100)

results <- run_simulation(nS, nF, fS, fF, lc)
plot(seq(-2, 2, length = 100), results, type = "l", main = "Misclassification Rate",
    xlab = "Log10(c)", ylab = "Rate")
```



We can see that the value of c that minimizes our misclassification rate is 11.7681195.

Exercise 3

We can modify the tusk example by including a bernoulli random variable with a chance of success of 0.98. This can be added to the code as it is below using the `rbinom()` function.

```
x <- c(1, 0, 1, 0, 0, 1)
x <- sapply(x, function(x) {
    ifelse(rbinom(1, 1, 0.98) == 1, x, 1 - x)
})
fS <- c(0.4, 0.12, 0.21, 0.12, 0.02, 0.32)
fF <- c(0.8, 0.2, 0.11, 0.17, 0.23, 0.25)
L <- function(f, x) {
    prod(f^x * (1 - f)^(1 - x))
}
LR <- L(fS, x)/L(fF, x)
print(L)
```

```
function(f, x) {
    prod(f^x * (1 - f)^(1 - x))
}
```

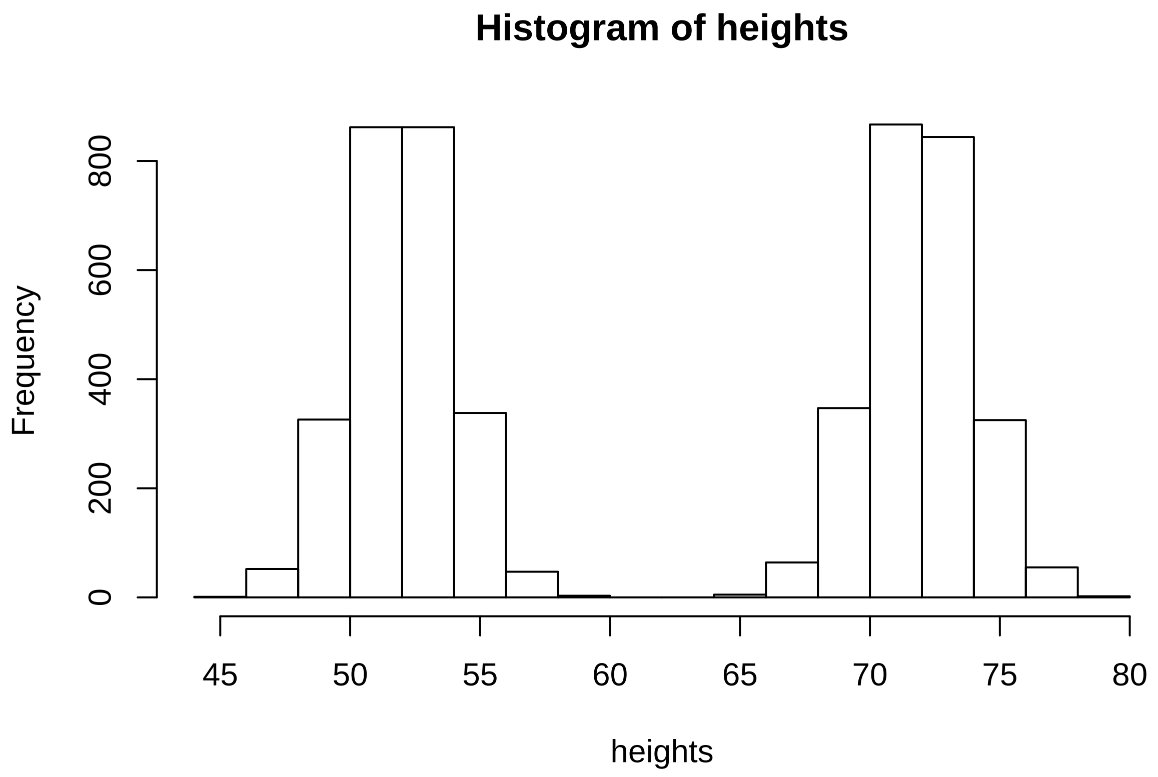
```
print(LR)
```

```
[1] 1.81359
```

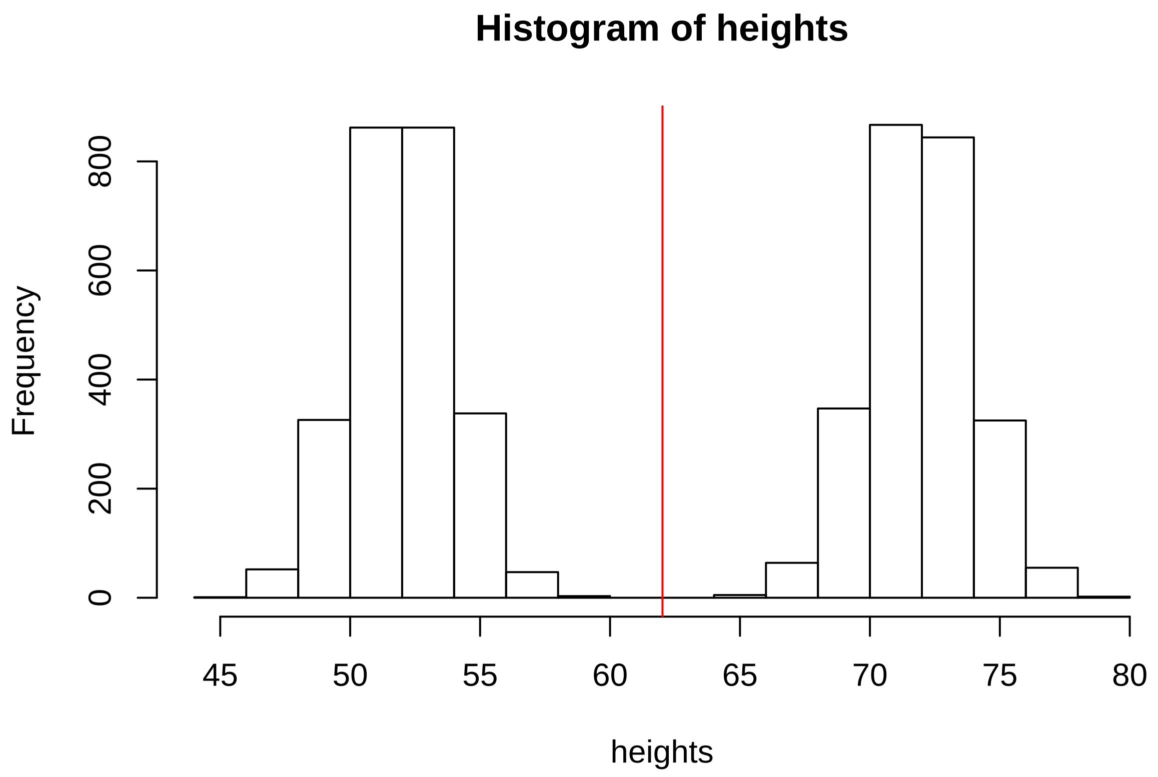
B. Mixture Models

Introductino to Mixture Models

Suppose that we receive a data set of adult, human heights. We don't know what kinds of humans, exactly, but we know that they are adults. The first thing we might want to do is explore and describe our data. Now suppose our data looks like this:



Based on what we know about the distribution of adult, human heights we might have expected it to look like a gaussian distribution, but here we clearly see what looks like two normal distributions next to one another. What would be the expected height if we simply calculated the mean of all of our data? Let's create another plot with the mean highlighted in red:



Does this make sense?

Calculating the mean in this way assumes that the data comes from a unimodal distribution, but we can clearly see here that we have what's called a **multimodal** distribution. We can imagine how this might be the case if we consider the fact that women are, on average, shorter than men. Thus, it could be that the data we received contained heights that weren't gender biased. That is, the data contains the heights recorded from both men and women. In other words, our the data was actually sampled from a population that can more accurately be described by separating it into a distinct number of subpopulations. In this case, 2; one for males and one for females.

In order to model this, we use what's called a **mixture model** where we assume that the to modes within our data represent data that was sampled from two independent distributions.

We can represent the probability of heights using the following equation:

$$P(X_i = x) = \sum_{k=1}^{K} P(X_i = x|Z_i = k)P(Z_i = k) = \sum_{k=1}^{K} P(X_i|Z_i = k)\pi_k$$

Where Z_i is the latent variable that provides the category (male or female here) for our recorded data point and π_k is the proportion of the population that's either one category or the other.

Therefore the likelihood function for our model would be:

$$L(\pi) = \prod_{i=1}^{n} P(X_i|\pi) = \prod_{i=1}^{n}\sum_{k=1}^{K} P(X_i|Z_i = k)\pi_k$$

Simulating from mixtures and computing log-likelihoods

(a)

An R function to simulate genetic data for tusks:

```r
#' Simulate tusks from a mixture distribution
#'
#' @param N number of tusks to simulate
#' @param w vector of mixture proportions of length K that sums to 1
#' @param F K by M matrix of allele frequencies, where M is the number of markers
#'
#' @return list containing an N by M matrix of data and an N vector of component memberships
#'
#' @example sim_mixture_tusks(10,c(0.4,0.6),matrix(c(0.40,0.12,0.21,0.12,0.02,0.32,0.8,0.2,0.11,0.17,0.
#'
#'
sim_mixture_tusks <- function(N, w, F) {

    memberships <- numeric(N)
    data <- matrix(0, nrow = N, ncol = nrow(F))

    for (i in 1:N) {

        # which category are we randomly choosing based on our mixture proportions
        z <- which.max(rmultinom(n = 1, size = 1, prob = w))
        memberships[i] <- z

        # generate identified for individual markers
        data[i, ] <- sapply(F[, z], function(x) {
            sample(c(1, 0), 1, prob = c(x, 1 - x))
        })

    }

    return(list(X = data, Z = memberships))
}
```

(b)

An R function to compute the log-likelihood for the simulated data set:

```r
#' Calculate the log-likelihood of a data matrix
#'
#' @param X N by M matrix of N samples by M genetic markers
#' @param w vector of mixture proportions of length K that sums to 1
#' @param F K by M matrix of allele frequencies, where M is the number of markers
#'
#' @return log-likelihood under the model for our data set
#'
#' @example calculate_log_likelihood(X,c(0.4,0.6),matrix(c(0.40,0.12,0.21,0.12,0.02,0.32,0.8,0.2,0.11,0
#'
#'
calculate_log_likelihood <- function(X, w, F) {

    s <- apply(X, 1, function(x) {

        v <- numeric(length(w))
        for (k in length(w)) {

            v[k] <- L(F[, k], x) * w[k]

        }

        sum(v)

    })

    return(sum(log(s)))

}
```

(c)

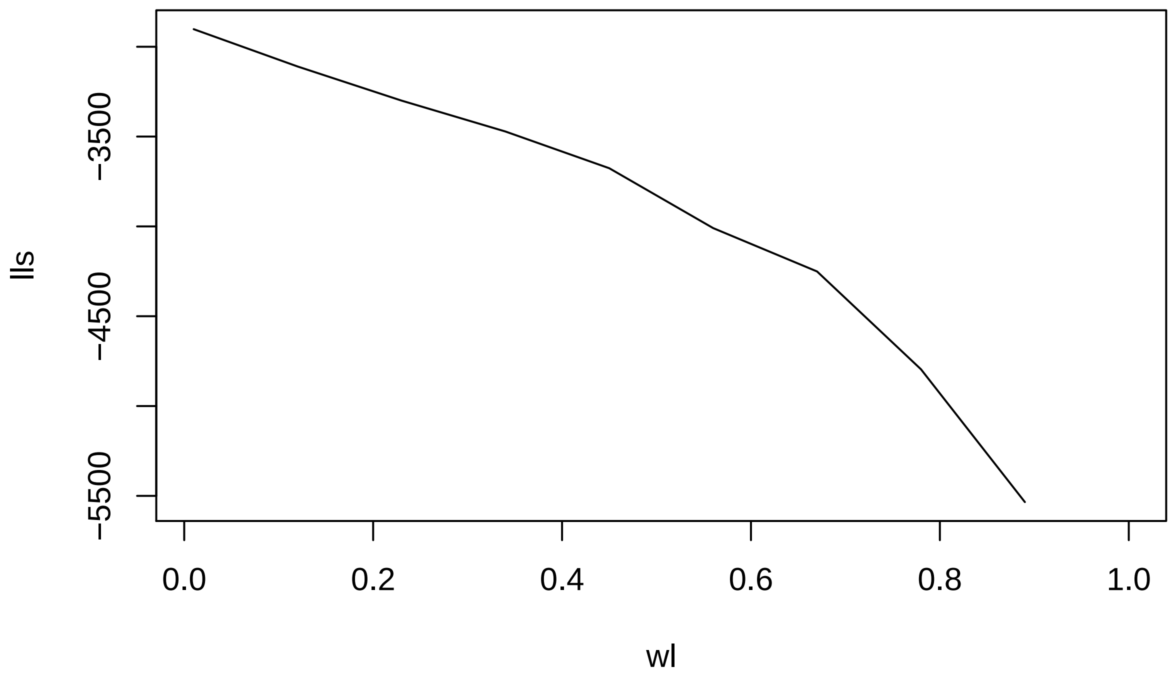
Now to test the functions:

```
N <- 1000
w <- c(0.25, 0.75)
F <- matrix(c(0.4, 0.12, 0.21, 0.12, 0.02, 0.32, 0.8, 0.2, 0.11, 0.17, 0.23,
    0.25), ncol = 2)

data <- sim_mixture_tusks(N, w, F)
calculate_log_likelihood(data$X, w, F)
```

```
[1] -3251.249
```

Now we can sweep through different values of our mixture proportions to see where we see the highest likelihood:

```
steps <- 10

wl <- seq(0.01, 1, length = steps)

lls <- sapply(wl, function(x) {

    w <- c(x, 1 - x)
    data <- sim_mixture_tusks(N, w, F)
    calculate_log_likelihood(data$X, w, F)

})

plot(wl, lls, type = "l")
```



EM Algorithm

First we need to write a function that calculates and returns the log likelihoods:

```
mixture.EM <- function(w.init, L, X) {

    w.curr <- w.init

    # store log-likehoods for each iteration
    log_liks <- c()
    ll <- calculate_log_likelihood(X, w.curr, L)
    log_liks <- c(log_liks, ll)
    delta.ll <- 1

    while (delta.ll > 1e-05) {
        w.curr <- EM.iter(w.curr, L)
        ll <- compute.log.lik(X, L, w.curr)
        log_liks <- c(log_liks, ll)
        delta.ll <- log_liks[length(log_liks)] - log_liks[length(log_liks) -
            1]
    }
    return(list(w.curr, log_liks))
}

EM.iter <- function(w.curr, L, ...) {

    # E-step: compute E_{Z|X,w0}[I(Z_i = k)]
    z_ik <- L
    for (i in seq_len(ncol(L))) {
        z_ik[, i] <- w.curr[i] * z_ik[, i]
    }
    z_ik <- z_ik/rowSums(z_ik)

    # M-step
    w.next <- colSums(z_ik)/sum(z_ik)
    return(w.next)
}
```

```
## # perform EM
## ee <- mixture.EM(w.init = c(0.5, 0.5), L, X)
## print(paste("Estimate = (", round(ee[[1]][1], 2), ",", round(ee[[1]][2], 2),
##     ")", sep = ""))
```

Session Information

```
sessionInfo()
```

```
R version 3.3.1 (2016-06-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.1 LTS

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] backports_1.0.4 magrittr_1.5    rprojroot_1.1   formatR_1.4
 [5] tools_3.3.1     htmltools_0.3.5 yaml_2.1.14     Rcpp_0.12.6
 [9] stringi_1.1.1   rmarkdown_1.3   knitr_1.15.1    git2r_0.18.0
[13] stringr_1.1.0   digest_0.6.11   workflowr_0.2.0 evaluate_0.10
```