MAST30027 Modern Applied Statistics Assignment 3

Tutorial:Wed 1-2PM, Yidi Deng

James La Fontaine

2023-09-24

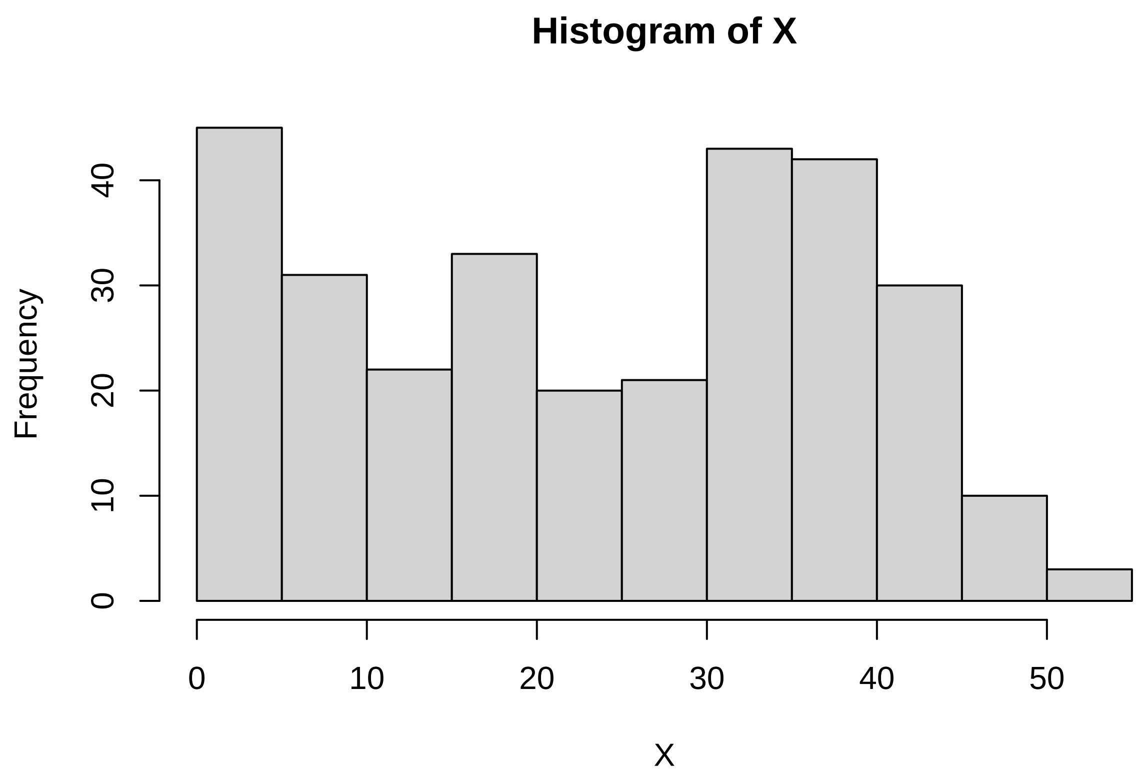
Question 1d

```r
X = scan(file="assignment3_prob1_2023.txt", what=double())

length(X)
```

```
## [1] 300
```

```r
hist(X)
```



Implementation of the EM algorithm

We will assume that the observed data follows a mixture of three Poisson distributions. Specifically, for $i = 1\ldots,n$,

$$Z_i \sim \text{categorical}\ (\pi_1, \pi_2, 1-\pi_1-\pi_2),$$

$$X_i|Z_i = 1 \sim \text{Poisson}(\lambda_1),$$
$$X_i|Z_i = 2 \sim \text{Poisson}(\lambda_2),$$
$$X_i|Z_i = 3 \sim \text{Poisson}(\lambda_3).$$

We aim to obtain MLE of parameters $\theta = (\pi_1, \pi_2, \lambda_1, \lambda_2, \lambda_3)$ using the EM algorithm.

We implement the E and M step in the `EM.iter` function below. The `compute.log.lik` function below computes the incomplete log-likelihood, assuming the parameters are known. We will check that the incomplete log-likelihoods increases at each step by plotting them. The `mixture.EM` function is the main function which runs multiple EM steps and checks for convergence by computing the incomplete log-likelihoods at each step.

```r
# w.init : initial value for pi
# lambda.init : initial value for lambda
# epsilon : If the incomplete log-likelihood has changed by less than epsilon,
# EM will stop.
# max.iter : maximum number of EM-iterations
mixture.EM <- function(X, w.init, lambda.init, epsilon=1e-5, max.iter=100) {

  w.curr = w.init
  lambda.curr = lambda.init

  # store incomplete log-likehoods for each iteration
  log_liks = c()

  # compute incomplete log-likehoods using initial values of parameters.
  log_liks = c(log_liks, compute.log.lik(X, w.curr, lambda.curr)$ill)

  # set the change in incomplete log-likelihood with 1
  delta.ll = 1

  # number of iteration
  n.iter = 1

  # If the log-likelihood has changed by less than epsilon, EM will stop.
  while((delta.ll > epsilon) & (n.iter <= max.iter)){

    # run EM step
    EM.out = EM.iter(X, w.curr, lambda.curr)

    # replace the current value with the new parameter estimate
    w.curr = EM.out$w.new
    lambda.curr = EM.out$lambda.new

    # incomplete log-likehoods with new parameter estimate
    log_liks = c(log_liks, compute.log.lik(X, w.curr, lambda.curr)$ill)

    # compute the change in incomplete log-likelihood
    delta.ll = log_liks[length(log_liks)]  - log_liks[length(log_liks)-1]

    # increase the number of iteration
    n.iter = n.iter + 1
  }
  return(list(w.curr=w.curr, lambda.curr=lambda.curr, log_liks=log_liks))
}
```

```r
EM.iter <- function(X, w.curr, lambda.curr) {

  # E-step: compute E_{Z|X,\theta_0}[I(Z_i = k)]
  # for each sample $X_i$, compute $P(X_i, Z_i=k)$
  prob.x.z = compute.prob.x.z(X, w.curr, lambda.curr)$prob.x.z

  # compute P(Z_i=k | X_i)
  P_ik = prob.x.z / rowSums(prob.x.z)

  # M-step
  w.new = colSums(P_ik)/sum(P_ik)  # sum(P_ik) is equivalent to sample size
  lambda.new = colSums(P_ik*X)/colSums(P_ik)

  return(list(w.new=w.new, lambda.new=lambda.new))
}
```

Now we write a function to compute the incomplete log-likelihood, assuming the parameters are known.

$$\ell(\theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{3} \pi_k f(x_i; \lambda_k) \right)$$

```
# Compute incomplete log-likehoods
compute.log.lik <- function(X, w.curr, lambda.curr) {

  # for each sample $X_i$, compute $P(X_i, Z_i=k)$
  prob.x.z = compute.prob.x.z(X, w.curr, lambda.curr)$prob.x.z

  # incomplete log-likehoods
  ill = sum(log(rowSums(prob.x.z)))

  return(list(ill=ill))
}

# for each sample $X_i$, compute $P(X_i, Z_i=k)$
compute.prob.x.z <- function(X, w.curr, lambda.curr) {

  # for each sample $X_i$, compute $P(X_i, Z_i=k)$. Store these values in the columns of L:
  L = matrix(NA, nrow=length(X), ncol= length(w.curr))
  for(k in seq_len(ncol(L))) {
    L[, k] = dpois(X, lambda=lambda.curr[k])*w.curr[k]
  }

  return(list(prob.x.z=L))
}
```

Apply the EM algorithm

Run EM algorithm with different initial values and check that the incomplete log-likelihoods increases at each step by plotting them.

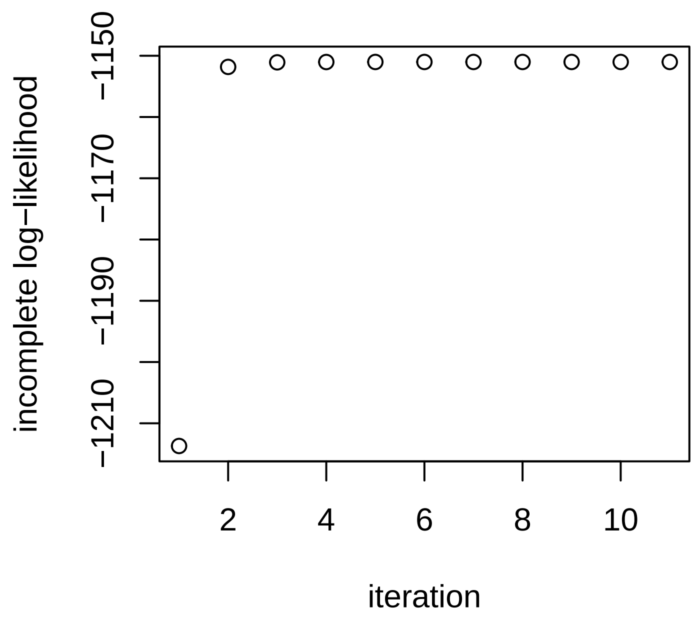
```
EM1 <- mixture.EM(X, w.init=c(0.3,0.3, 0.4), lambda.init=c(3, 20, 35), epsilon=1e-5, max.iter=100)
ee = EM1
print(paste("Estimate pi = (", round(ee$w.curr[1],2), ",",
            round(ee$w.curr[2],2), ",",
            round(ee$w.curr[3],2), ")", sep=""))
```

```
## [1] "Estimate pi = (0.25,0.25,0.5)"
```

```
print(paste("Estimate lambda = (", round(ee$lambda.curr[1],2), ",",
            round(ee$lambda.curr[2],2), ",",
            round(ee$lambda.curr[3],2), ")", sep=""))
```

```
## [1] "Estimate lambda = (5.17,18.09,36.94)"
```

```
plot(ee$log_liks, ylab='incomplete log-likelihood', xlab='iteration')
```



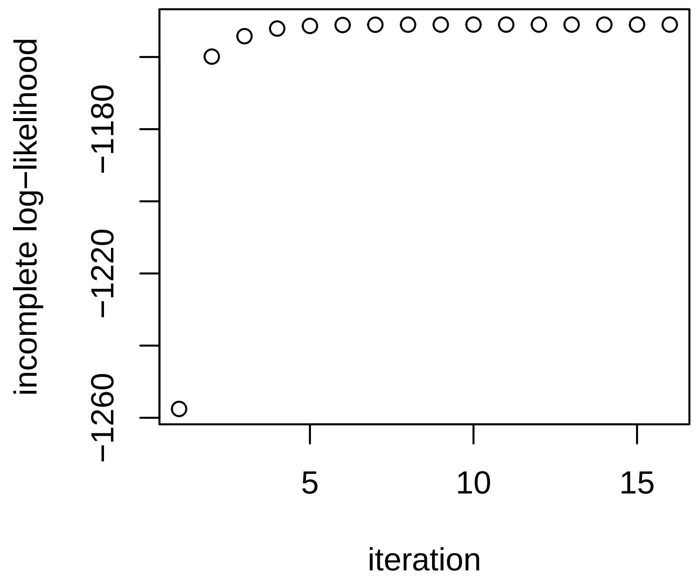
```
EM2 <- mixture.EM(X, w.init=c(0.1,0.2, 0.7), lambda.init=c(5, 25, 40), epsilon=1e-5, max.iter=100)
ee = EM2
print(paste("Estimate pi = (", round(ee$w.curr[1],2), ",",
            round(ee$w.curr[2],2), ",",
            round(ee$w.curr[3],2), ")", sep=""))
```

```
## [1] "Estimate pi = (0.25,0.25,0.5)"
```

```
print(paste("Estimate lambda = (", round(ee$lambda.curr[1],2), ",",
            round(ee$lambda.curr[2],2), ",",
            round(ee$lambda.curr[3],2), ")", sep=""))
```

```
## [1] "Estimate lambda = (5.17,18.09,36.94)"
```

```
plot(ee$log_liks, ylab='incomplete log-likelihood', xlab='iteration')
```



Check which estimators have the highest incomplete log-likelihood.

```
EM1$log_liks[length(EM1$log_liks)]
```

```
## [1] -1151.015
```

```
EM2$log_liks[length(EM2$log_liks)]
```

```
## [1] -1151.015
```

Estimators from the two EM runs have (equally) highest incomplete log-likelihoods. You can see that the estimators from the EM runs are the same, so it doesn't matter which estimators we choose. Lets choose the estimators from the first EM run - $\hat{\pi}_1 = 0.25,\ \hat{\pi}_2 = 0.25,\ \hat{\lambda}_1 = 5.17,\ \hat{\lambda}_2 = 18.09,\ \hat{\lambda}_3 = 36.94$.

Question 2c

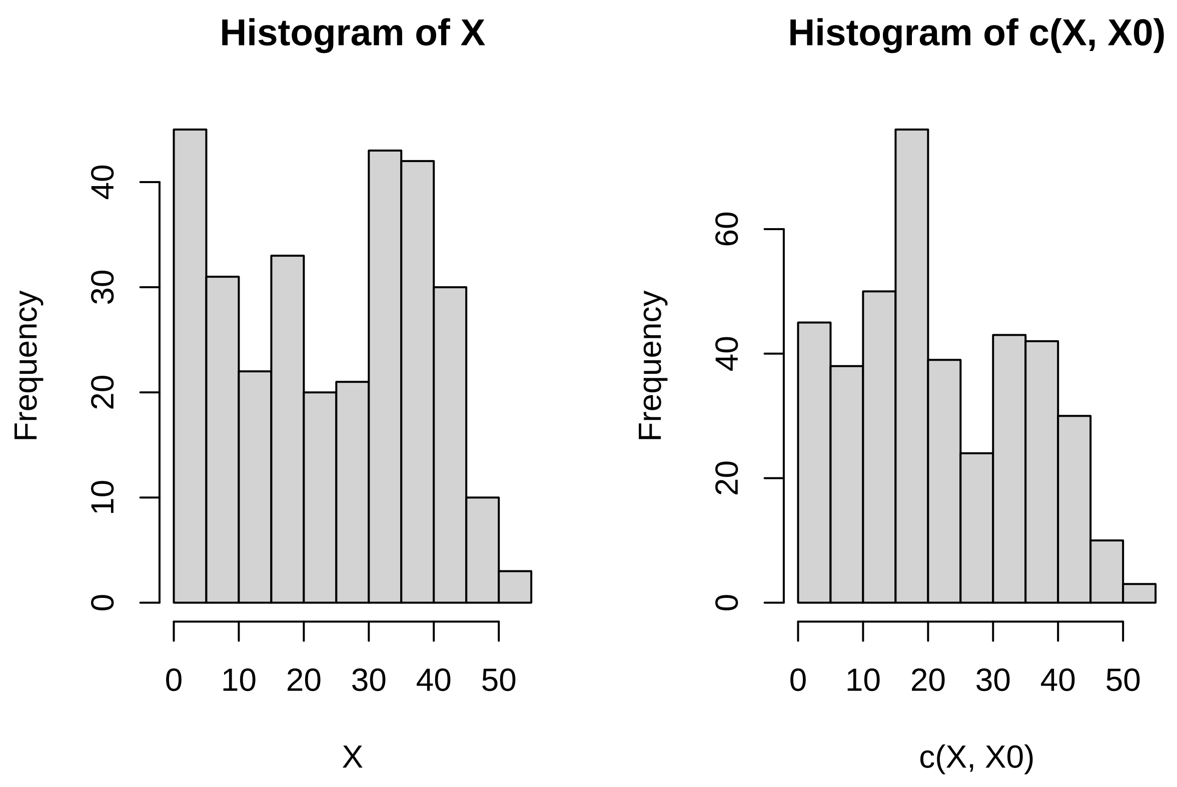
```r
X = scan(file="assignment3_prob1_2023.txt", what=double())
X0 = scan(file="assignment3_prob2_2023.txt", what=double())
length(X)
```

```
## [1] 300
```

```r
length(X0)
```

```
## [1] 100
```

```r
par(mfrow=c(1,2))
hist(X)
hist(c(X,X0))
```



Implementation of the EM algorithm

We will assume that the observed data follows a mixture of three Poisson distributions. Additionally, for samples 301 to 400, we assume that they follow a known Poisson distribution. Specifically, for $i = 1\dots, n$,

$$Z_i \sim \text{categorical }(\pi_1, \pi_2, 1-\pi_1-\pi_2),$$

$$\begin{aligned} X_i|Z_i = 1 &\sim \text{Poisson}(\lambda_1),\\ X_i|Z_i = 2 &\sim \text{Poisson}(\lambda_2),\\ X_i|Z_i = 3 &\sim \text{Poisson}(\lambda_3),\end{aligned}$$

and for i=301, . . . , 400,

$$X_i \sim \text{Poisson}(\lambda_2).$$

We aim to obtain MLE of parameters $\theta = (\pi_1, \pi_2, \lambda_1, \lambda_2, \lambda_3)$ using the EM algorithm.

We implement the E and M step in the `EM.iter` function below. The `compute.log.lik` function below computes the incomplete log-likelihood, assuming the parameters are known. We will check that the incomplete log-likelihoods increases at each step by plotting them. The `mixture.EM` function is the main function which runs multiple EM steps and checks for convergence by computing the incomplete log-likelihoods at each step.

```r
# w.init : initial value for pi
# lambda.init : initial value for lambda
# epsilon : If the incomplete log-likelihood has changed by less than epsilon,
# EM will stop.
# max.iter : maximum number of EM-iterations
mixture.EM <- function(X, w.init, lambda.init, epsilon=1e-5, max.iter=100) {

  w.curr = w.init
  lambda.curr = lambda.init

  # store incomplete log-likehoods for each iteration
  log_liks = c()

  # compute incomplete log-likehoods using initial values of parameters.
  log_liks = c(log_liks, compute.log.lik(X, w.curr, lambda.curr)$ill)

  # set the change in incomplete log-likelihood with 1
  delta.ll = 1

  # number of iteration
  n.iter = 1

  # If the log-likelihood has changed by less than epsilon, EM will stop.
  while((delta.ll > epsilon) & (n.iter <= max.iter)){

    # run EM step
    EM.out = EM.iter(X, w.curr, lambda.curr)

    # replace the current value with the new parameter estimate
    w.curr = EM.out$w.new
    lambda.curr = EM.out$lambda.new

    # incomplete log-likehoods with new parameter estimate
    log_liks = c(log_liks, compute.log.lik(X, w.curr, lambda.curr)$ill)

    # compute the change in incomplete log-likelihood
    delta.ll = log_liks[length(log_liks)]  - log_liks[length(log_liks)-1]

    # increase the number of iteration
    n.iter = n.iter + 1
  }
  return(list(w.curr=w.curr, lambda.curr=lambda.curr, log_liks=log_liks))
}
```

```r
EM.iter <- function(X, w.curr, lambda.curr) {

  # E-step: compute E_{Z|X,\theta_0}[I(Z_i = k)]

  # for each sample $X_i$, compute $P(X_i, Z_i=k)$
  prob.x.z = compute.prob.x.z(X, w.curr, lambda.curr)$prob.x.z

  # compute P(Z_i=k | X_i)
  P_ik = prob.x.z / rowSums(prob.x.z)

  # M-step
  w.new = colSums(P_ik)/sum(P_ik)  # sum(P_ik) is equivalent to sample size
  lambda.new = colSums(P_ik*X)/colSums(P_ik)
  lambda.new[2] = (colSums(P_ik*X)[2] + sum(X0))/(colSums(P_ik)[2] + 100)

  #print(paste("lambda.new = (", lambda.new, ")", sep=""))


  return(list(w.new=w.new, lambda.new=lambda.new))
}
```

Now we write a function to compute the incomplete log-likelihood, assuming the parameters are known.

$$\ell(\theta) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{3} \pi_k f(x_i; \lambda_k)\right)$$

```r
# Compute incomplete log-likehoods
compute.log.lik <- function(X, w.curr, lambda.curr) {

  # for each sample $X_i$, compute $P(X_i, Z_i=k)$
  prob.x.z = compute.prob.x.z(X, w.curr, lambda.curr)$prob.x.z

  # incomplete log-likehoods
  ill = sum(log(rowSums(prob.x.z)))

  return(list(ill=ill))
}

# for each sample $X_i$, compute $P(X_i, Z_i=k)$
compute.prob.x.z <- function(X, w.curr, lambda.curr) {

  # for each sample $X_i$, compute $P(X_i, Z_i=k)$. Store these values in the columns of L:
```

```
  L = matrix(NA, nrow=length(X), ncol= length(w.curr))
  for(k in seq_len(ncol(L))) {
    L[, k] = dpois(X, lambda=lambda.curr[k])*w.curr[k]
  }

  return(list(prob.x.z=L))
}
```

Apply the EM algorithm

Run EM algorithm with different initial values and check that the incomplete log-likelihoods increases at each step by plotting them.

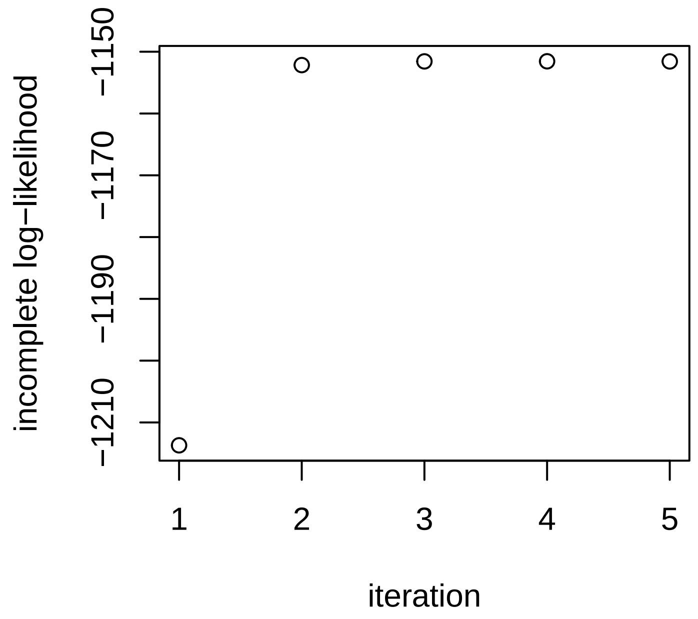
```
EM1 <- mixture.EM(X, w.init=c(0.3,0.3, 0.4), lambda.init=c(3, 20, 35), epsilon=1e-5, max.iter=100)
ee = EM1
print(paste("Estimate pi = (", round(ee$w.curr[1],2), ",",
            round(ee$w.curr[2],2), ",",
            round(ee$w.curr[3],2), ")", sep=""))
```

```
## [1] "Estimate pi = (0.25,0.25,0.51)"
```

```
print(paste("Estimate lambda = (", round(ee$lambda.curr[1],2), ",",
            round(ee$lambda.curr[2],2), ",",
            round(ee$lambda.curr[3],2), ")", sep=""))
```

```
## [1] "Estimate lambda = (5.12,17.38,36.79)"
```

```
plot(ee$log_liks, ylab='incomplete log-likelihood', xlab='iteration')
```



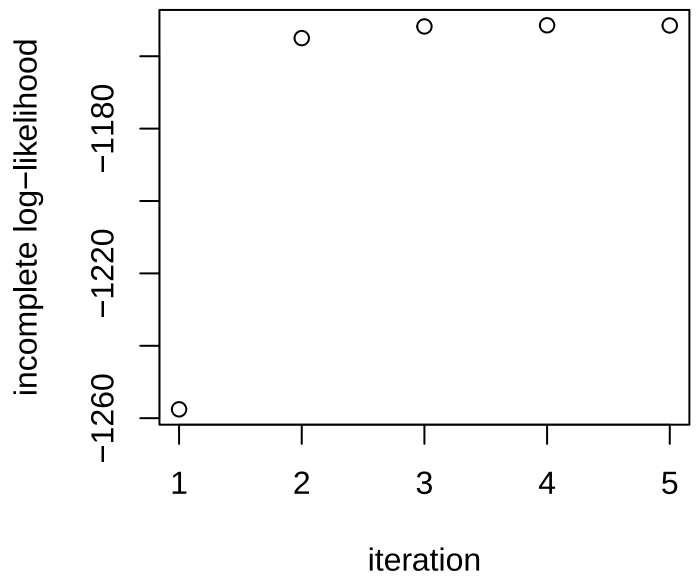
```
EM2 <- mixture.EM(X, w.init=c(0.1,0.2, 0.7), lambda.init=c(5, 25, 40), epsilon=1e-5, max.iter=100)
ee = EM2
print(paste("Estimate pi = (", round(ee$w.curr[1],2), ",",
            round(ee$w.curr[2],2), ",",
            round(ee$w.curr[3],2), ")", sep=""))
```

```
## [1] "Estimate pi = (0.25,0.25,0.5)"
```

```
print(paste("Estimate lambda = (", round(ee$lambda.curr[1],2), ",",
            round(ee$lambda.curr[2],2), ",",
            round(ee$lambda.curr[3],2), ")", sep=""))
```

```
## [1] "Estimate lambda = (5.13,17.43,36.86)"
```

```
plot(ee$log_liks, ylab='incomplete log-likelihood', xlab='iteration')
```



Check which estimators have the highest incomplete log-likelihood.

```
EM1$log_liks[length(EM1$log_liks)]
```

```
## [1] -1151.57
```

```
EM2$log_liks[length(EM2$log_liks)]
```

```
## [1] -1151.507
```

Estimators from the two EM runs nearly have equally highest incomplete log-likelihoods. You can see that the incomplete log-likelihood is slightly higher for the estimators from the second EM run and so we will choose the estimators from the second EM run - $\hat{\pi}_1 = 0.25,\ \hat{\pi}_2 = 0.25,\ \hat{\lambda}_1 = 5.13,\ \hat{\lambda}_2 = 17.43,\ \hat{\lambda}_3 = 36.86.$