Project Stat 6115

Akshay Patil (801034919)

12/14/2020

Gaussian Mixture Model

Generating Samples from Mixture of Normal Distrbution

```
N <- Number of samples to be generated

k <-  Number of gaussian distribution

mu <- vector of μ_k values

std_dev <- vector of corresponding std deviation values
```

Following function generates data from mixture of 'k' normal distribution with probability

```
gen_data_n_gaussian <- function(N,k,mu.true,std_dev.true, mix_prob.true){
        #Function for any number of gaussians with desired mixing proportions

        index <- sample(k, size=N, replace = T, mix_prob.true)
        samples <- rnorm(n=N, mean=mu.true[index], sd=std_dev.true[index])
        return(samples)
}
```

Following function plots the density of samples generated from mixture of 'k' normal distribution. It also plots the true density

```
plot_GM_data <- function(N, k, mu.true, std_dev.true, mix_prob.true, both=TRUE){
        samples = gen_data_n_gaussian(N, k, mu.true, std_dev.true, mix_prob.true)
        plot(density(samples),main="Density of generated data"
             ,col="red",ylim=c(0,max(density(samples)$y)+(0.5*(max(density(samples)$y)))),lty=1,lwd = 3

        # True_density
        res_density=0
        x=seq(-20,20,0.1)
        for ( i in 1:k){
```

```
                res_density= res_density +
                  mix_prob.true[i]*dnorm(x,mu.true[i],std_dev.true[i])
        }
        lines(x,(res_density),col="black",lty=1,lwd=2)
        legend("topright",
               legend=c("Density of randomly generated data ",
                        "True density of finite mixture model"),
               col=c("red","black"),lty=1:1,lwd = 3:2)
}
```

Implementing the E step

```
E.step <- function(samples, mu, std_dev, mix_prob, K){
        #function to calculate the responsibilities

        for(i in 1:K){
          assign(paste0("prod_", i), (mix_prob[[i]]* dnorm(samples, mean = mu[[i]],sd = std_dev[[i]])))
        }
        prod_list <- mget(ls(pattern = "prod_"))                                    # LIST

        prod_df = as.data.frame(prod_list)

        total_prob = as.data.frame(rowSums(prod_df))

        for(i in 1:K){
          assign(paste0("posterior_clus", i), (prod_df[i] / total_prob))
        }

        posterior_clust_list <- mget(ls(pattern = "posterior_clus"))                # LIST

        responsibilities <- as.data.frame(posterior_clust_list)
        log.likelihood = sum(log(total_prob))

        mylist =list(responsibilities, log.likelihood)

        return(mylist)

}
```

Implementing M step

```
M.step <- function(samples, responsibilities, K){
        #function to update parameters

        N = length(samples)
        for(i in 1:K){
                assign(paste0("clust.mu_",i), sum(responsibilities[,i]*samples) / sum(responsibilities[

                assign(paste0("mixing_prob_", i), sum(responsibilities[,i]) / N)
```

```
        }

        clust_mu_list <- mget(ls(pattern = "clust.mu_"))                          # LIST
        clust_mixprob_list <- mget(ls(pattern = "mixing_prob_"))                  # LIST

        clust_mu_df <- as.data.frame(clust_mu_list)
        clust_mixprob_df <- as.data.frame(clust_mixprob_list)


        for(i in 1:K){
                assign(paste0("clust.var_",i), sum(responsibilities[,i]*((samples - clust_mu_list[[i]])
                                        sum(responsibilities[,i]))
        }

        clust_var_list <- mget(ls(pattern = "clust.var_"))
        clust_var_df <- as.data.frame(clust_var_list)



        mylist = list(clust_mu_df, clust_var_df, clust_mixprob_df)


        return(mylist)

}
```

Iterative EM algorithm

```
EM_algorithm <- function(samples, N, iteration = 500, K){

        prob_pi = rep(0.5,K)
        mu = sample(samples,K)
        std_dev = rep((sqrt(sum((samples-mean(samples))^2)/N)), K)

        log_likeli = c()

        for (i in 1:iteration){
                if (i==1){
                        #initial parameters to calculate responsibilties
                        list_1= E.step(samples, mu=mu, std_dev=std_dev, mix_prob = prob_pi, K)
                        list_2= M.step(samples, responsibilities=list_1[[1]], K)
                        log_likeli =c(list_1[[2]])
                }
                else{
                        list_1 = E.step(samples, mu=list_2[[1]], std_dev=sqrt(list_2[[2]]),
                                        mix_prob=list_2[[3]], K)
                        list_2= M.step(samples, responsibilities=list_1[[1]], K)

                        log_likeli = c(log_likeli, list_1[[2]])

                }

                if (i>2){
```

```
                        if(log_likeli[i]-log_likeli[i-1] < 1e-5){
                                break
                        }
                }
        }

        final_list <- list(list_2, log_likeli, list_1[[1]])
        return(final_list)

}
```

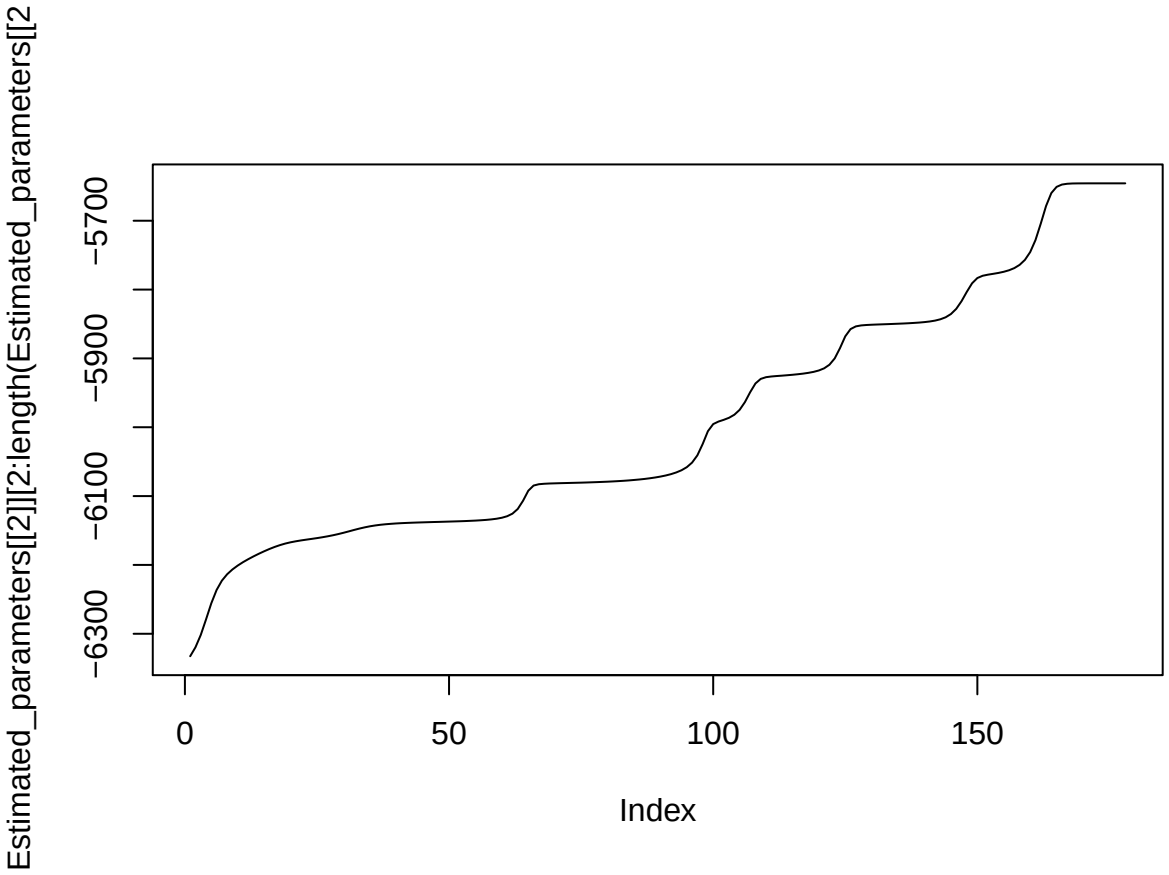
Genreating samples and plotting the densities

```
N = 2000
mu.true = c(-9,-6,-3,0,3,6,9)
std_dev.true = c(0.6,0.6,0.6,0.6,0.6,0.6,0.6)
mix_prob.true = c(0.15,0.15,0.15,0.15,0.15,0.15,0.10)
k=length(mu.true)

samples = gen_data_n_gaussian(N, k, mu.true, std_dev.true,
                              mix_prob.true)

#par(mfrow = c(1,1))

#plot_GM_data(N, k, mu.true, std_dev.true,mix_prob.true)
```

```
K = 7
samples = samples
Estimated_parameters = EM_algorithm(samples, N=length(samples), K =7,iteration = 1000)
Estimated_parameters[[1]]
```

```
## [[1]]
##   clust.mu_1 clust.mu_2 clust.mu_3 clust.mu_4 clust.mu_5 clust.mu_6 clust.mu_7
## 1  -9.038152  -5.999613  -2.998675   9.041701   2.961799 0.07905847   6.022384
##
## [[2]]
##   clust.var_1 clust.var_2 clust.var_3 clust.var_4 clust.var_5 clust.var_6
## 1   0.3959868   0.4040368   0.3329943   0.3757124   0.3543564   0.3382872
##   clust.var_7
## 1   0.3612425
##
## [[3]]
##   mixing_prob_1 mixing_prob_2 mixing_prob_3 mixing_prob_4 mixing_prob_5
## 1      0.137349     0.1550775     0.1392638     0.1061725     0.1647368
##   mixing_prob_6 mixing_prob_7
## 1     0.1510846     0.1463157
```

```
plot(Estimated_parameters[[2]][2:length(Estimated_parameters[[2]])], type = 'l')
```

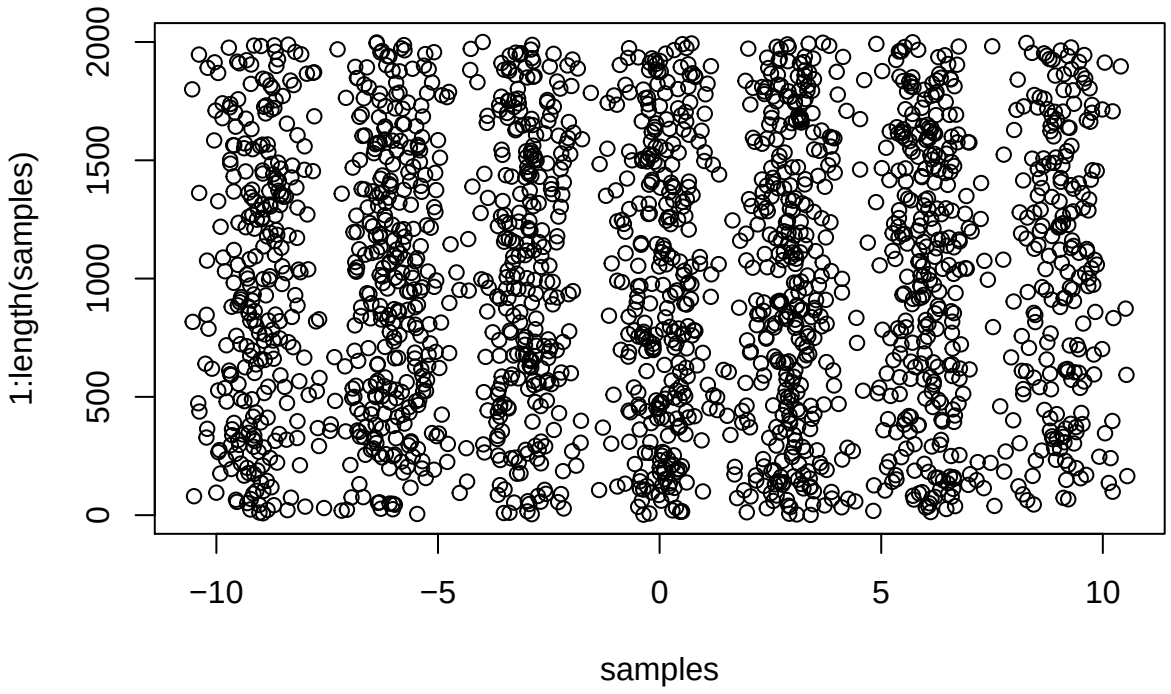


Soft Probability

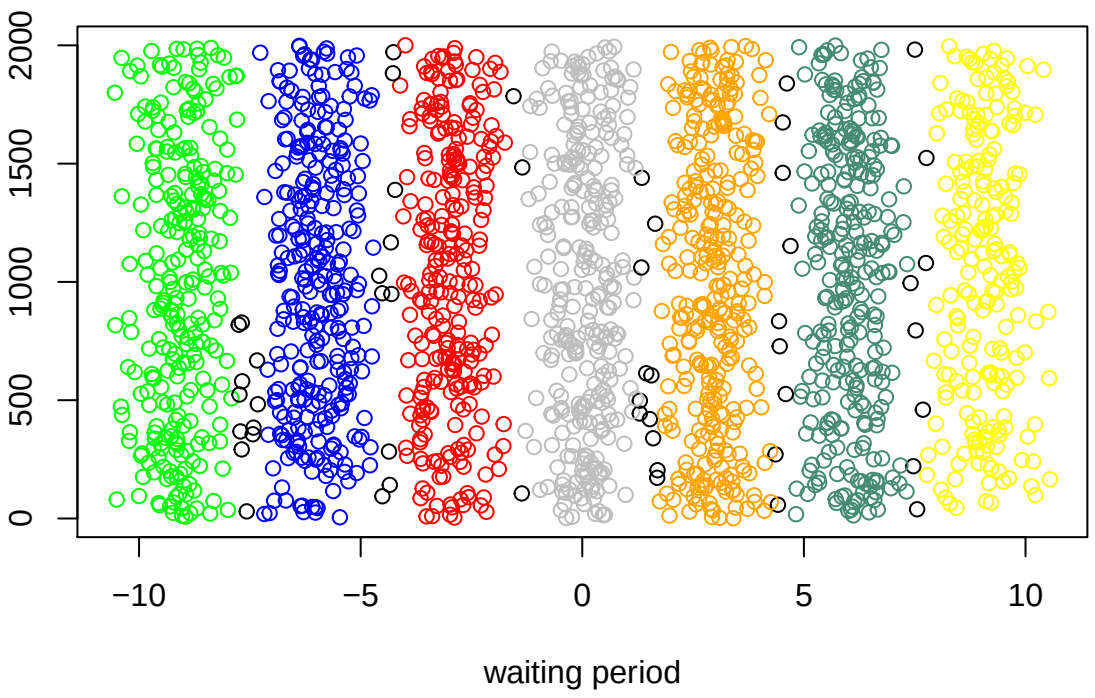
```
Estim_respons = round(Estimated_parameters[[3]]*100,2)
Estim_respons = cbind(samples, Estim_respons)
head(Estim_respons)
```

```
##      samples prod_1 prod_2 prod_3 prod_4 prod_5 prod_6 prod_7
## 1  3.4081506      0   0.00  0e+00      0  99.99      0   0.01
## 2 -0.3635889      0   0.00  0e+00      0   0.00    100   0.00
## 3  3.0950490      0   0.00  0e+00      0 100.00      0   0.00
## 4 -2.8857447      0   0.00  1e+02      0   0.00      0   0.00
## 5 -5.4688589      0  99.99  1e-02      0   0.00      0   0.00
## 6  2.9257628      0   0.00  0e+00      0 100.00      0   0.00
```

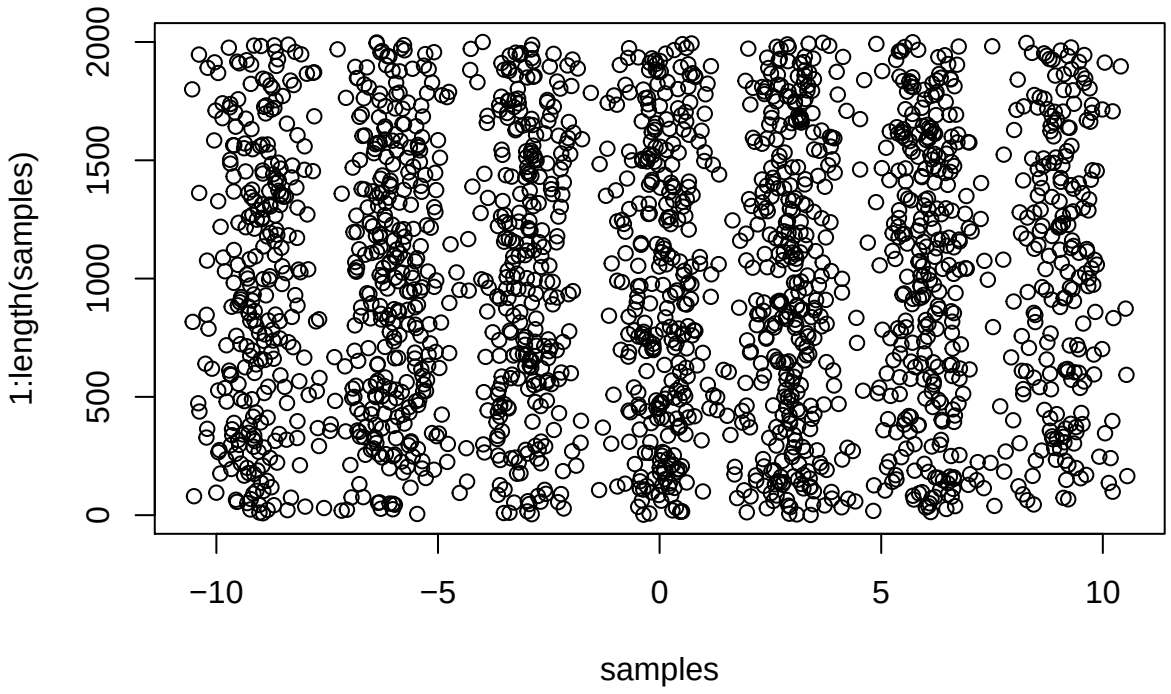
```
boolean1 = Estim_respons[,2] > 85
boolean2 = Estim_respons[,3] > 85
boolean3 = Estim_respons[,4] > 85
boolean4 = Estim_respons[,5] > 85
boolean5 = Estim_respons[,6] > 85
boolean6 = Estim_respons[,7] > 85
boolean7 = Estim_respons[,8] > 85



X = data.frame(samples)
plot(samples,1:length(samples))
```



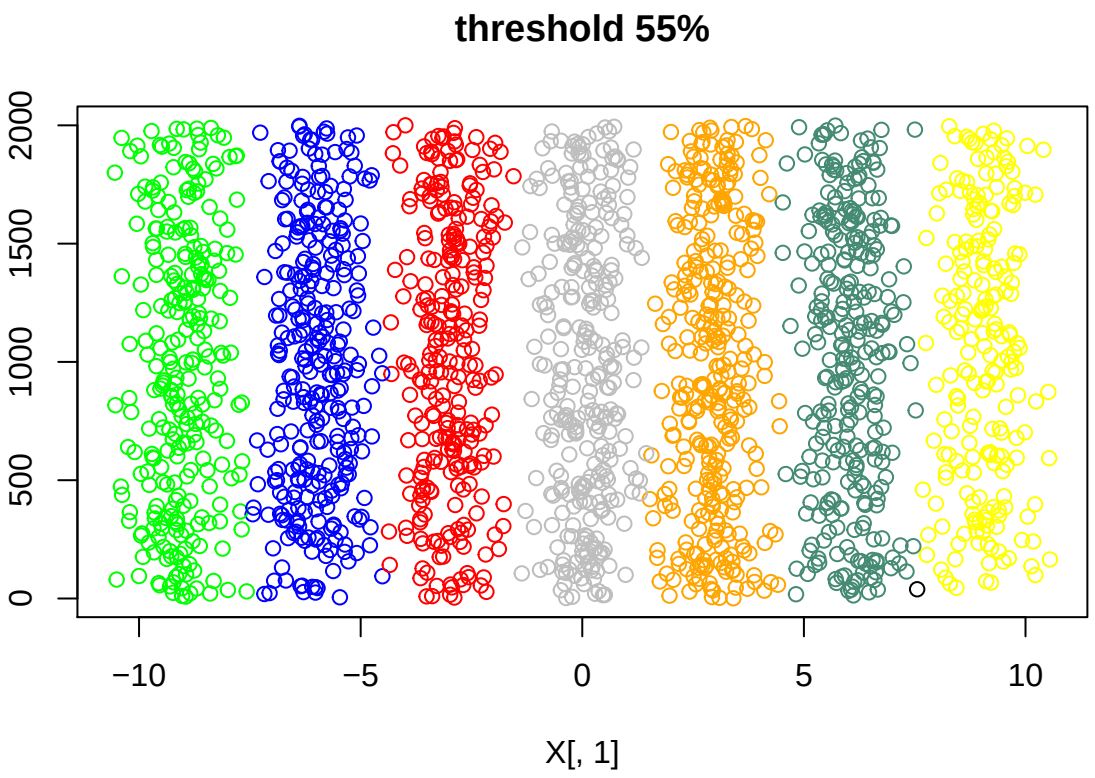

```
X$Colour = 'black'
X$Colour[boolean1] = 'green'
X$Colour[boolean2] = 'blue'
X$Colour[boolean3] = 'red'
X$Colour[boolean4] = 'yellow'
X$Colour[boolean5] = 'orange'
X$Colour[boolean6] = 'gray'
X$Colour[boolean7] = 'aquamarine4'

plot(X[,1], 1:length(X[,1]), col = X[,2],xlab ='waiting period',ylab="", main = 'threshold 85%')
```



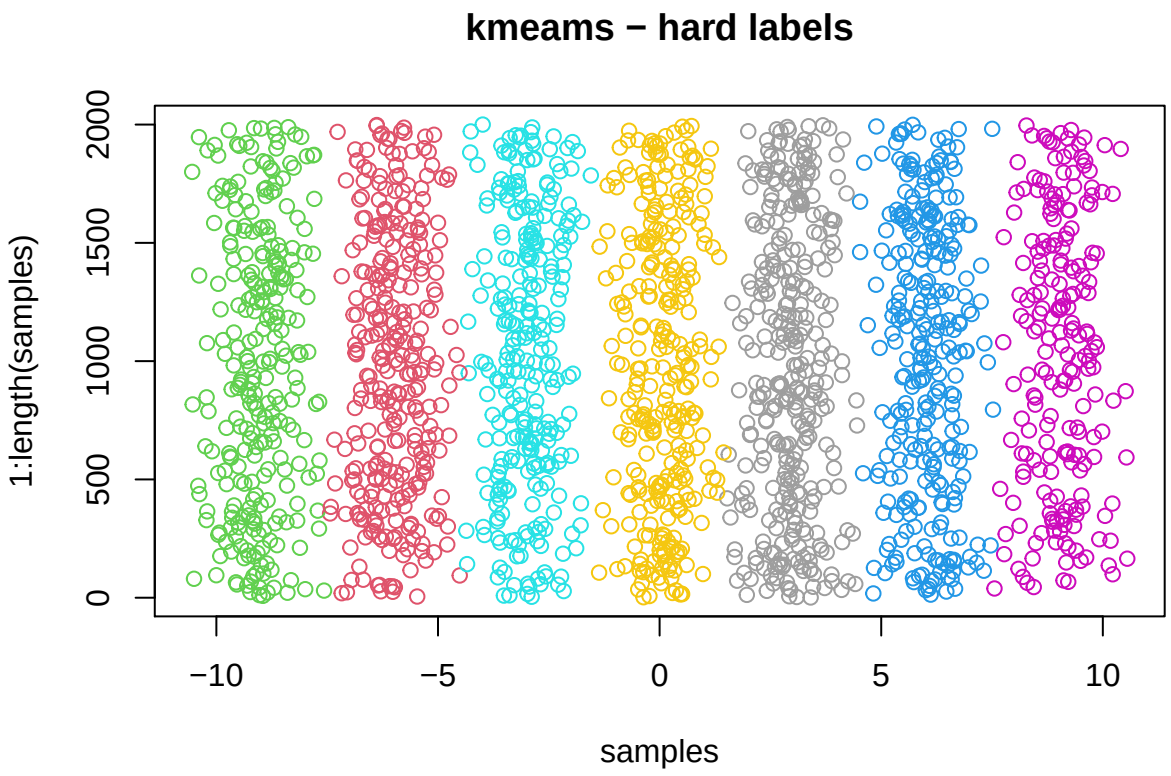

```
boolean1 = Estim_respons[,2] > 55
boolean2 = Estim_respons[,3] > 55
boolean3 = Estim_respons[,4] > 55
boolean4 = Estim_respons[,5] > 55
boolean5 = Estim_respons[,6] > 55
boolean6 = Estim_respons[,7] > 55
boolean7 = Estim_respons[,8] > 55

X = data.frame(samples)
plot(samples,1:length(samples))
```



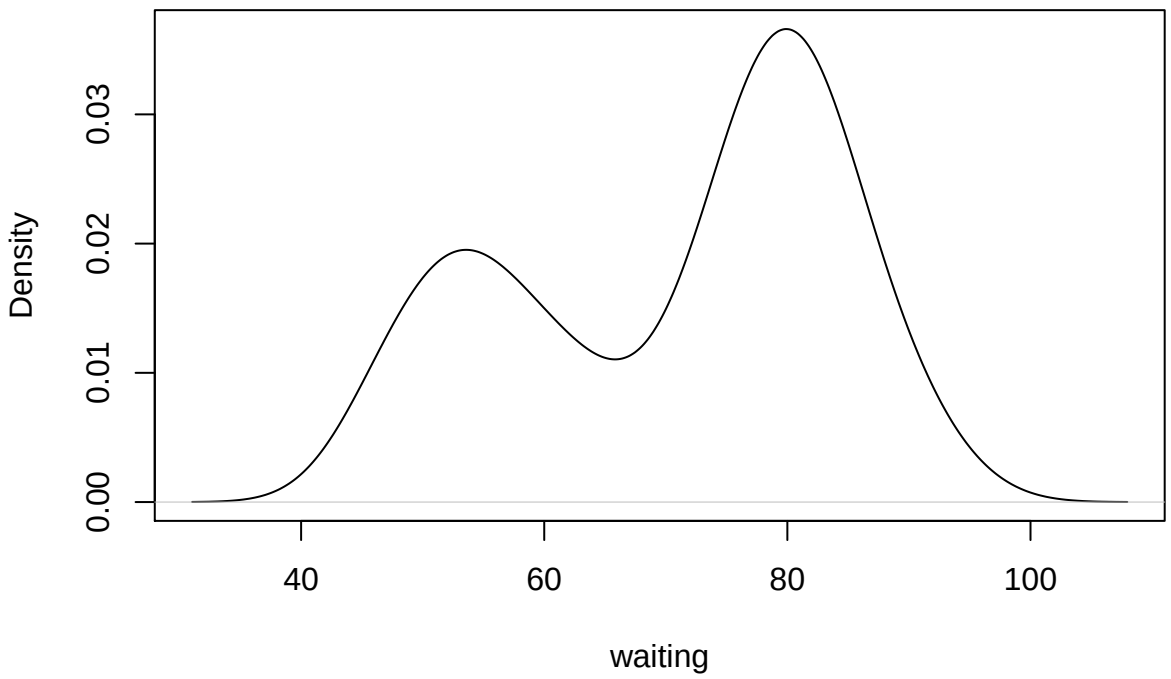

```
X$Colour = 'black'
X$Colour[boolean1] = 'green'
X$Colour[boolean2] = 'blue'
X$Colour[boolean3] = 'red'
X$Colour[boolean4] = 'yellow'
X$Colour[boolean5] = 'orange'
X$Colour[boolean6] = 'gray'
X$Colour[boolean7] = 'aquamarine4'

plot(X[,1], 1:length(X[,1]), col = X[,2],ylab="",main = 'threshold 55%')
```




```
km.out = kmeans(samples, 7, nstart = 2)
plot(samples,1:length(samples),col = (km.out$cluster+1), pch = 1, cex = 1, main = 'kmeams - hard labels
```



```
waiting_period = faithful$waiting
plot(density(waiting_period),xlab = "waiting",main = " ")
```




```
K = 2
samples = waiting_period
Estimated_parameters = EM_algorithm(samples, N=length(samples), K =2)
(Estim_mean = Estimated_parameters[[1]][[1]])
```

```
##   clust.mu_1 clust.mu_2
## 1   54.61583   80.09169
```

```
(Estim_sd = sqrt(Estimated_parameters[[1]][[2]]))
```

```
##   clust.var_1 clust.var_2
## 1    5.872052    5.867118
```

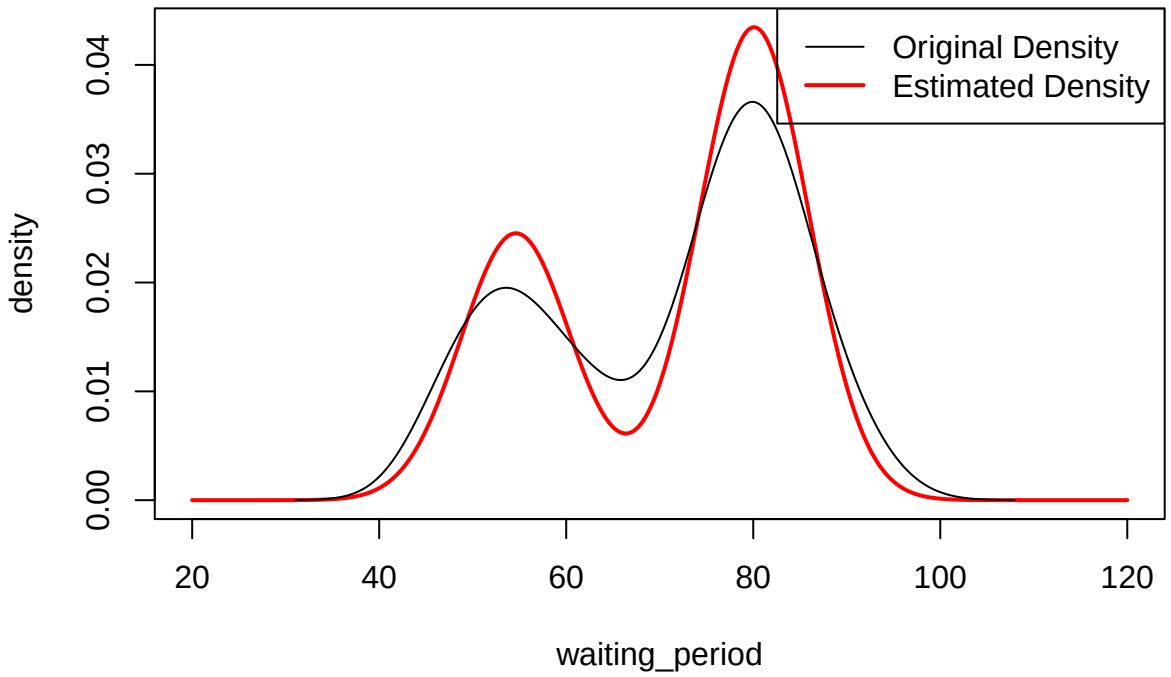
```
(Estim_mix_prop = Estimated_parameters[[1]][[3]])
```

```
##   mixing_prob_1 mixing_prob_2
## 1     0.3609153     0.6390847
```

Plotting the clusters

```
res_density=0
x=seq(20,120,0.1)
for ( i in 1:K){
```

```
        res_density= res_density +
        Estim_mix_prop[[i]]*dnorm(x,Estim_mean[[i]],Estim_sd[[i]])
}
plot(x,(res_density),col="red",typ='l',lwd=2,
        xlab = 'waiting_period', ylab = 'density')
lines(density(waiting_period))
legend("topright",
             legend=c("Original Density ",
                "Estimated Density"),
             col=c("black","red"),lty=1:1,lwd = 1:2)
```

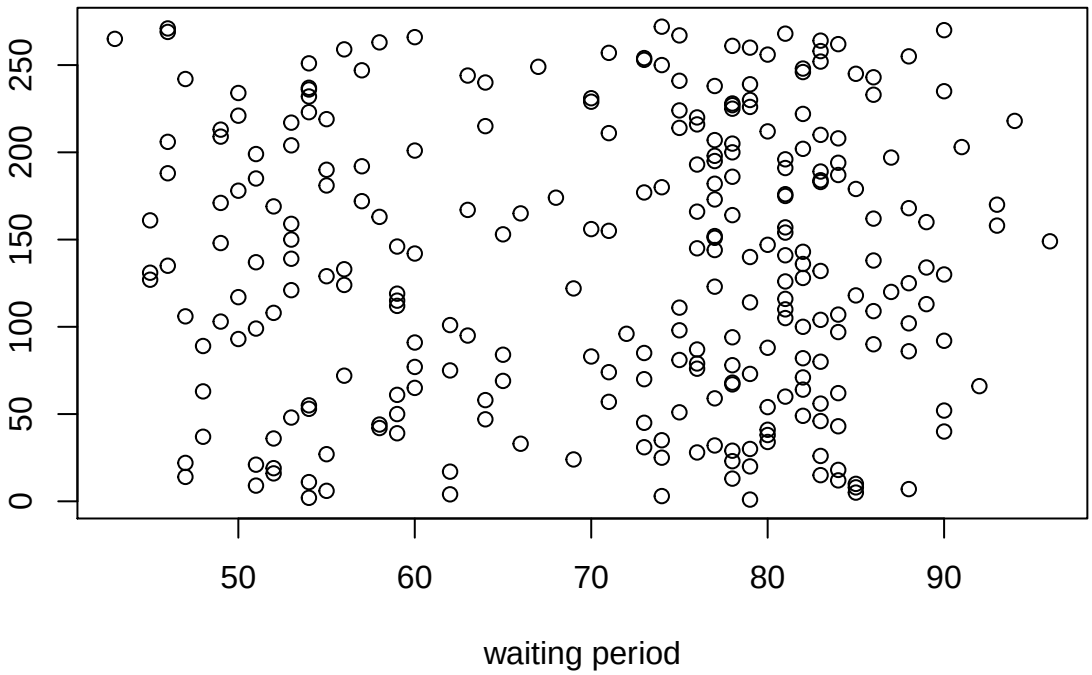


Soft Probability

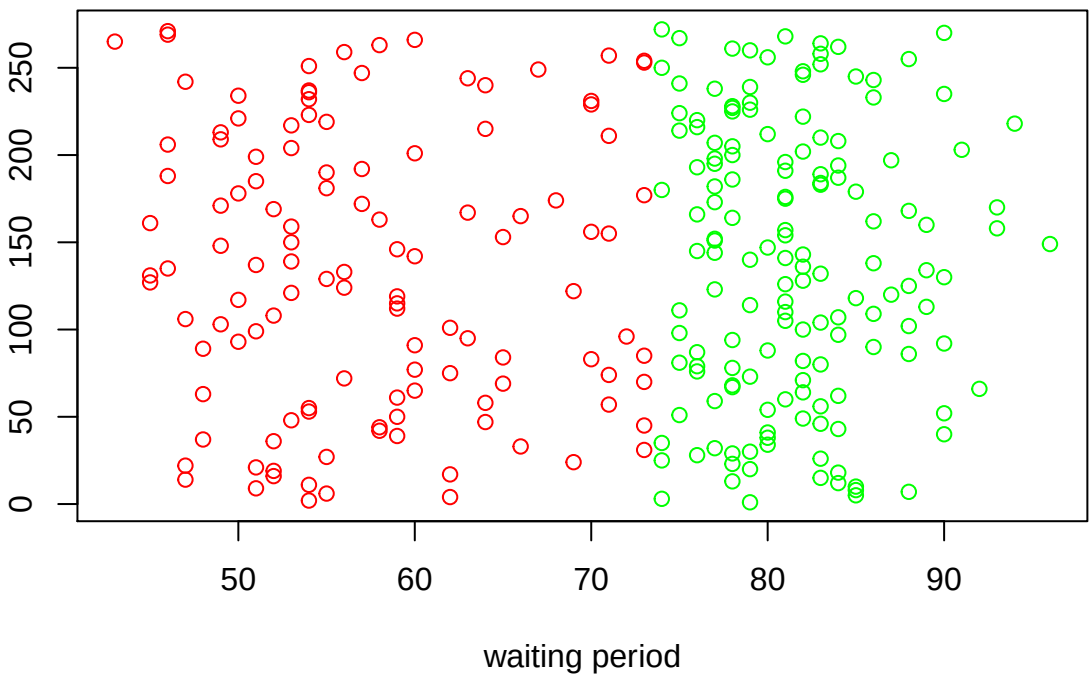
```
Estim_respons = round(Estimated_parameters[[3]]*100,2)
Estim_respons = cbind(waiting_period, Estim_respons)
head(Estim_respons)
```

```
##   waiting_period prod_1 prod_2
## 1             79   0.01  99.99
## 2             54  99.99   0.01
## 3             74   0.41  99.59
## 4             62  96.75   3.25
## 5             85   0.00 100.00
## 6             55  99.98   0.02
```

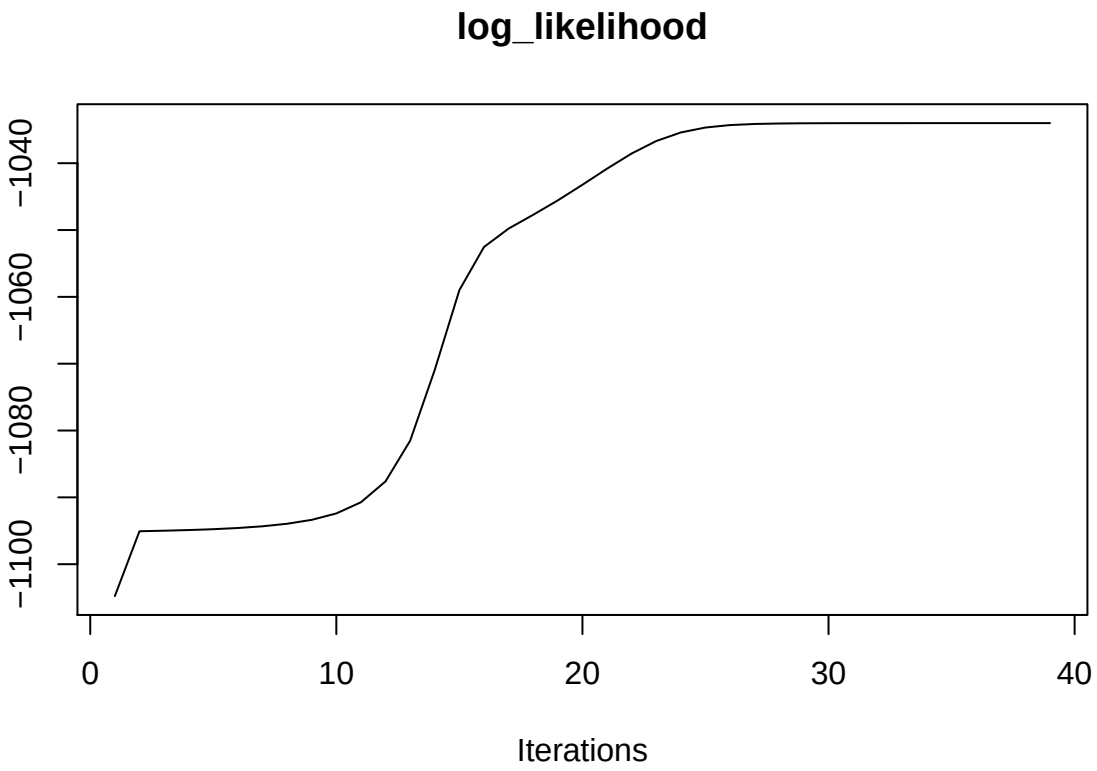
```
boolean = Estim_respons[,2] > 0.70
cluster_1 = waiting_period[boolean]
cluster2_2 = waiting_period[!boolean]

X = data.frame(waiting_period)
plot(samples,1:length(waiting_period),xlab ='waiting period',ylab="")
```



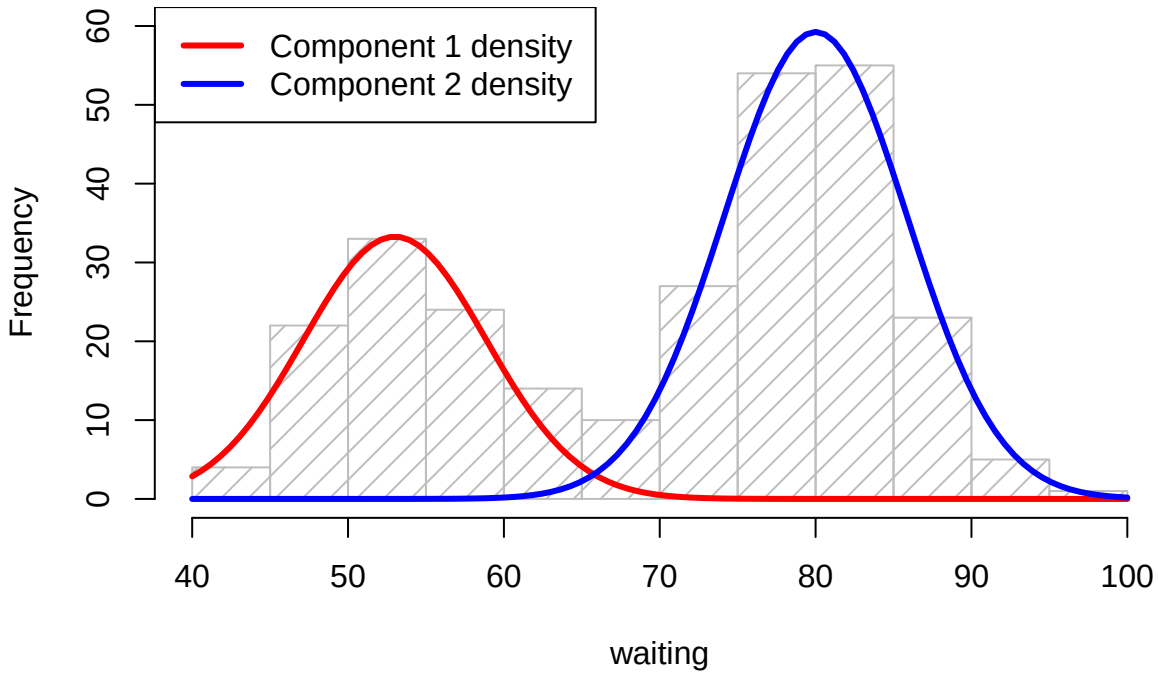

```
X$Colour = 'black'
X$Colour[boolean] = 'red'
X$Colour[!boolean] = 'green'
plot(X[,1], 1:length(X[,1]), col = X[,2],xlab ='waiting period',ylab="",main = "threshold 90%")
```



```
plot(Estimated_parameters[[2]],type = 'l',ylab="",xlab='Iterations', main ="log_likelihood" )
```




```
h <- hist(samples, breaks = 10, density = 10,
          col = "gray", xlab = "waiting", main = "",ylim = c(0,60))
xfit <- seq(40, 100, length = 100)
yfit <- 0.36*dnorm(xfit, mean = 53, sd = 5.87)
yfit <- yfit * diff(h$mids[1:2]) * length(samples)

lines(xfit, yfit, col = "red", lwd = 3)

yfit1 <- 0.64*dnorm(xfit, mean = 80, sd = 5.86)
yfit1 <- yfit1 * diff(h$mids[1:2]) * length(samples)
lines(xfit, yfit1, col = "blue", lwd = 3)

legend("topleft",
             legend=c("Component 1 density ",
                "Component 2 density"),
             col=c("red","blue"),lty=1:1,lwd = 3:3)
```




```
h <- hist(samples, breaks = 10, density = 10,
          col = "gray", xlab = "waiting", main = "",ylim = c(0,60))
lines(xfit, yfit1+yfit, col = "darkgreen",lwd = 3)
legend("topleft",
             legend=c("Mixture density "),
             col=c("darkgreen"),lty=1,lwd = 3)
```