INDR421 HW#7, Prepared by Arda Kırkağaç, 49799.

In this homework, our goal was to employ an Expectation-Maximization (EM) algorithm, whose mean values are initiated by k-Means algorithm. The bivariate Gaussian data is generated using mvrnorm as usual. K-Means algorithm code is below, it basically randomly chooses five points from our input data, and then iterates the algorithm two times:

```
centroids <<- NULL
assignments <<- NULL
K <- 5
N <- length(class_sizes)
numOfPoints <- sum(class_sizes)

km_iteration <- 2

for(i in 1:km_iteration ){

  if (is.null(centroids) == TRUE) {
    centroids <- X[sample(1:numOfPoints, K),]
  }
  else {
    for (k in 1:K) {
      centroids[k,] <- colMeans(X[assignments == k,])
    }
  }

  D <- as.matrix(dist(rbind(centroids, X), method = "euclidean"))
  D <- D[1:nrow(centroids), (nrow(centroids) + 1):(nrow(centroids) + nrow(X))]
  assignments <<- sapply(1:ncol(D), function(h) {which.min(D[,h])})

}
```

I defined N as the length of class sizes, which is different from my previous assignments. Before we use EM algorithm, we first need to initiate multivariate gaussian random variables four our 5 clusters whose means are already calculated by k-Means algorithm. The covariance matrices, however, needs to be calculated from the sample data. After initialization, the iteration is run until 100 iterations are completed. Below you can see the E-step:

```
#E-step
  sumG <- matrix(0,sum(class_sizes))
  for (i in 1:numOfPoints){
    for ( j in 1:K){

      G[i,j] <- class_priors[j] * ((det(sample_covariances[,,j]))^(-0.5))%*%
        exp((-0.5)*t(X[i,]- centroids[,j])%*% solve(sample_covariances[,,j])%*%(X[i,]- centroids[,j]))

      sumG[i] <- sumG[i]+G[i,j]
    }
  }

  P=matrix(0,numOfPoints,K)
    for ( j in 1:K){
      P[,j] <- G[,j]/sumG
    }
```

Using calculated h values in E-step, we can proceed to M-step, where the means and covariances are updated accordingly to prepare for the next E-step:

```
#M-step
  for (i in 1:K){
    centroids[,i] <- t(X)%*%P[,i]/sum(P[,i])
  }

  for (i in 1:K){
    c_var=matrix(0,2,2)
    for (j in 1:numOfPoints){
      c_var=c_var+P[j,i]*(X[j,]-centroids[,i])%*%t(X[j,]-centroids[,i])
    }
    sample_covariances[,,i] <- c_var/sum(P[,i])
  }

  class_priors <- colSums(P)/numOfPoints
```

After 100 iterations, you can see the resulting mean matrix for gaussians on the right, which is identical to the matrix in the document, with index differences:

```
             [,1]        [,2]
[1,] -2.6759195  2.44658900
[2,]  2.4887435  2.67687075
[3,]  2.6622246 -2.30911078
[4,] -2.0441920 -2.69776844
[5,]  0.1553518  0.05773826
```

After plotting points, and finding 0.05-density curves for each Gaussian using ellipse() function, resulting plot is below: