projetMAD

Project of MAD

owner: Guangyue CHEN

E1 Simulation

Q1 Simulate sample of 1000 vectors from a 2 dimensional mixture with 2 components

```
library(MASS)
```

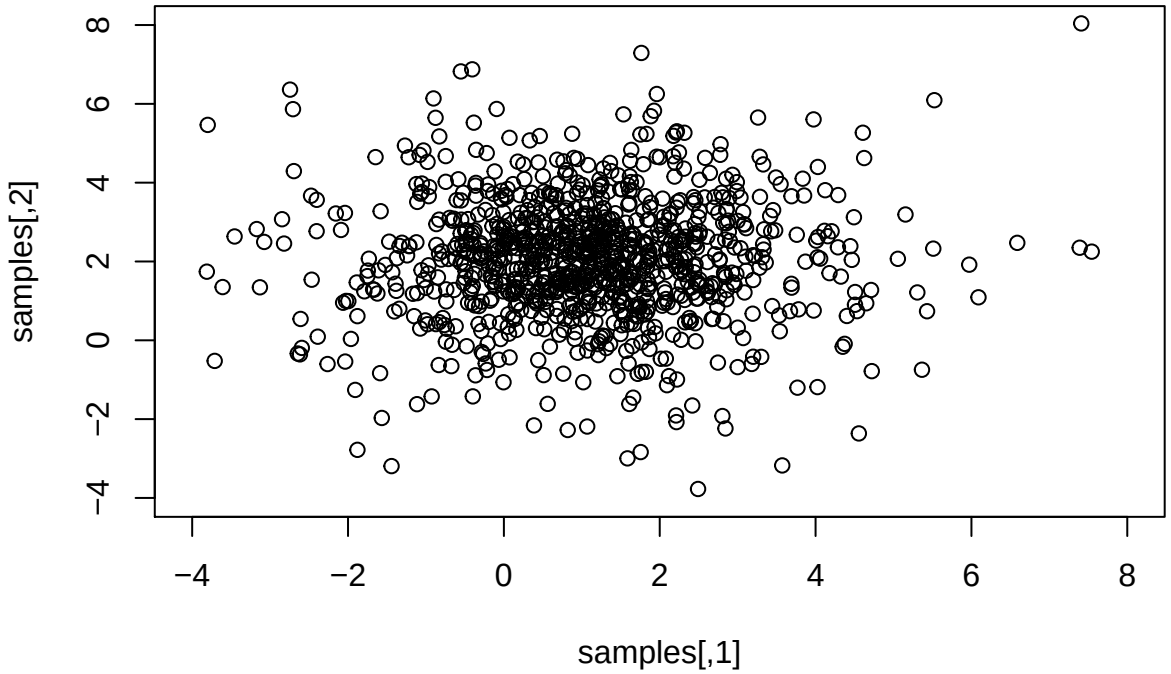
```
## Warning: package 'MASS' was built under R version 3.4.4
```

```
nks<-rmultinom(1,1000,prob=c(1/2,1/2))
mu <- as.vector(c(1,2))
Sigma1 <- matrix(c(1,0,0,1),2,2)
Sigma2 <- 4*Sigma1

samples<- rbind( mvrnorm(nks[1], mu, Sigma1), mvrnorm(nks[2], mu, Sigma2))
```

Q2 Display the sample

```
plot(samples,xlim=c(-4,8),ylim=c(-4,8))
```



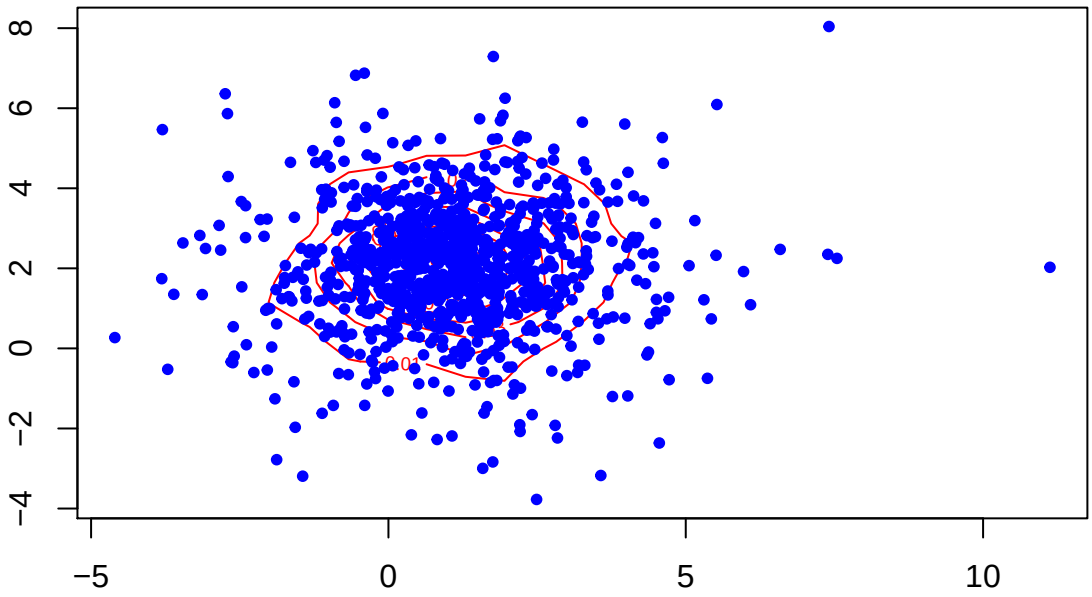
###Q3 Display the coutour plot of the two dimensional density. There are two ways to show it.

```
require("ggplot2")
```

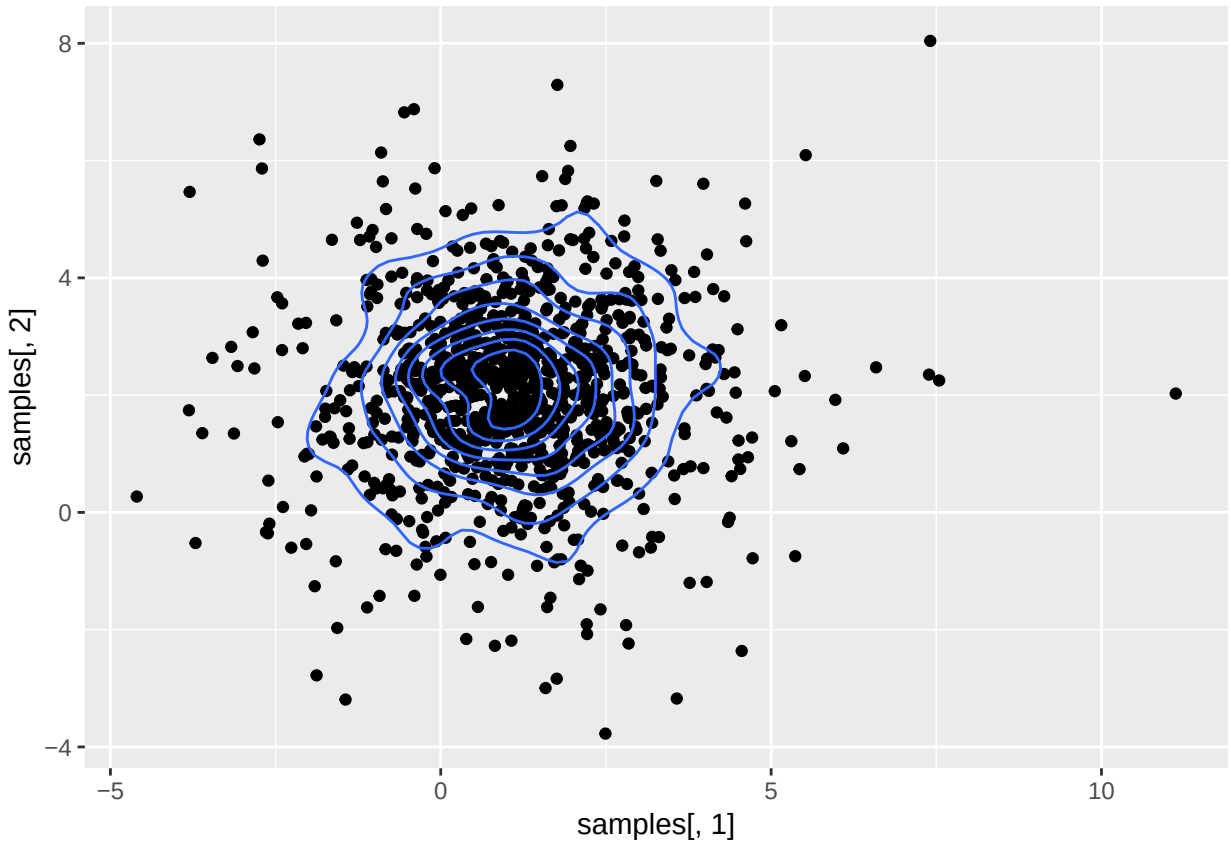
```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
z<-kde2d(samples[,1],samples[,2])
contour(z,col="red")
#plot
points(samples[,c(1,2)],col="blue",pch=20)
```




```
ggplot(data.frame(samples), aes(samples[,1], samples[,2])) + geom_point() + stat_density2d()
```



#E2 Exercise 2 Mclust versus kmeans

Q1 Run Mclust on the simulated data from the first exercise and comment the result.

```
require("mclust")
```

```
## Loading required package: mclust
```

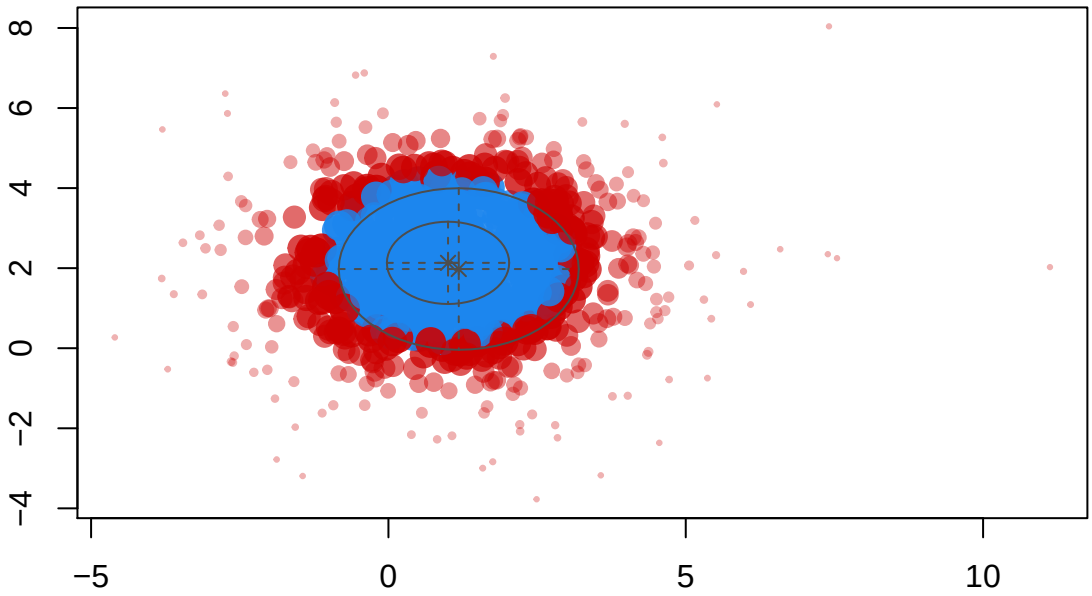
```
## Warning: package 'mclust' was built under R version 3.4.4
```

```
## Package 'mclust' version 5.4.2
## Type 'citation("mclust")' for citing this R package in publications.
```

```
Mclust(samples, 2, modelNames = "VII") -> res
summary(res)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VII (spherical, varying volume) model with 2 components:
##
##  log.likelihood    n df       BIC       ICL
##       -3689.781 1000  7 -7427.916 -8067.127
##
## Clustering table:
##   1   2
## 642 358
```

```
plot(res,"uncertainty")
```



comment We find that the clustering is not exactly same as the real classification, and there are so many points are uncertain.

Q2 Estimate the parameters of the simulated data from the first exercise using mclust.

```
print("pi")
```

```
## [1] "pi"
```

```
res$parameters$pro
```

```
## [1] 0.5326084 0.4673916
```

```
print("mu")
```

```
## [1] "mu"
```

```
res$parameters$mean
```

```
##          [,1]     [,2]
## [1,] 1.002280 1.182652
## [2,] 2.132564 1.978601
```

```
print("Sigma")
```

```
## [1] "Sigma"
```

```
res$parameters$variance$sigma
```

```
## , , 1
##
##          [,1]     [,2]
## [1,] 1.057178 0.000000
## [2,] 0.000000 1.057178
##
```

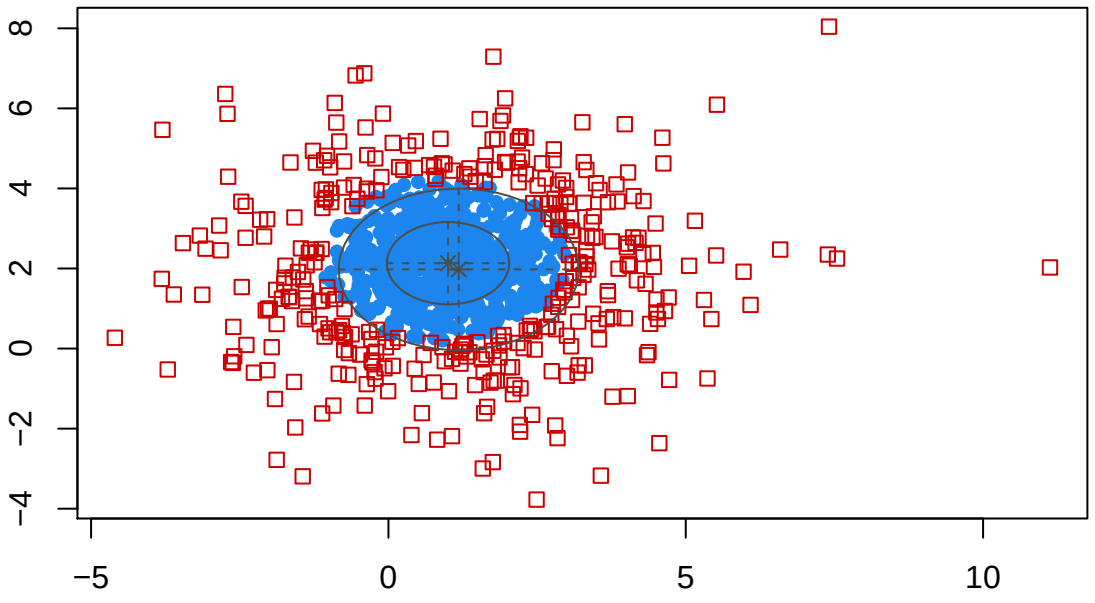
```
## , , 2
##
##          [,1]     [,2]
## [1,] 4.073189 0.000000
## [2,] 0.000000 4.073189
```

Q3 Find a partition of the simulated data into two classes using mclust

```
res$classification
```

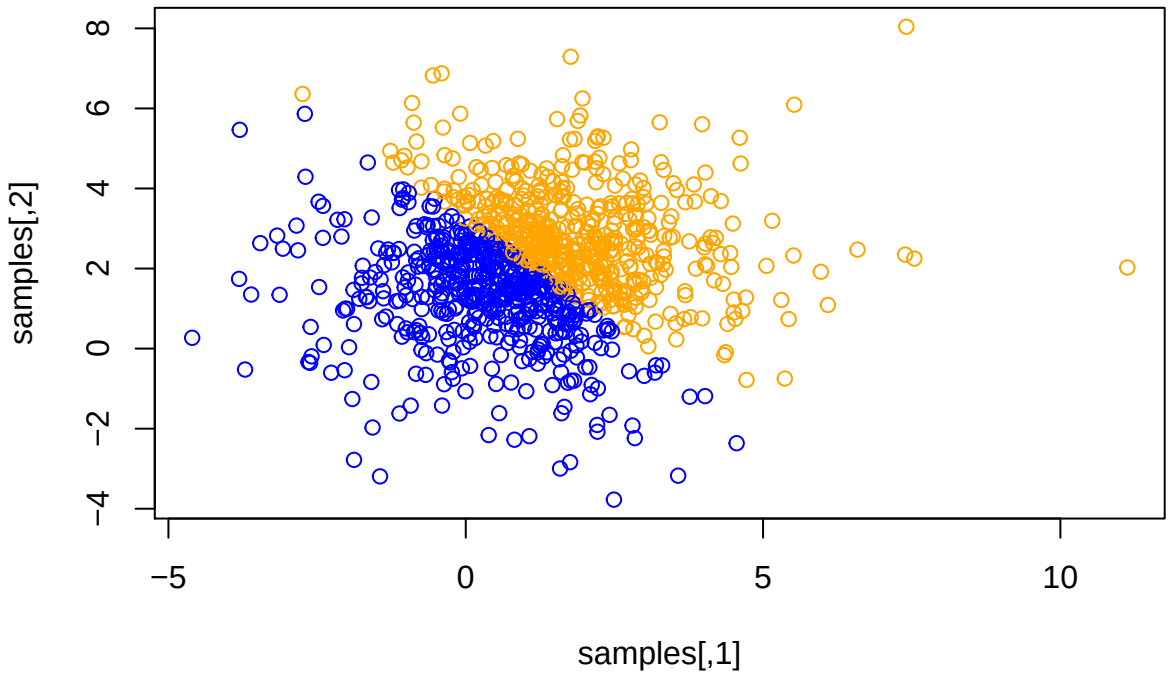
```
##    [1] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 1 2 2
##   [35] 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1
##   [69] 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
##  [103] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
##  [137] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1
##  [171] 1 1 1 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1
##  [205] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
##  [239] 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1
##  [273] 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [307] 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2
##  [341] 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
##  [375] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [409] 1 2 1 1 1 1 1 1 2 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
##  [443] 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1
##  [477] 1 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
##  [511] 1 1 1 2 1 1 2 2 2 2 2 2 2 1 1 2 1 2 2 2 1 2 2 1 2 1 2 1 2 1 2 1 2 2
##  [545] 1 2 2 2 2 1 2 2 2 1 2 1 1 2 2 1 2 2 2 1 1 2 2 2 1 1 2 2 2 2 1 2 2 1 1
##  [579] 2 2 2 1 1 2 1 1 2 1 2 2 1 2 1 1 1 2 2 2 2 1 2 2 1 1 2 2 2 2 2 2 1 1
##  [613] 2 1 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 2 1 2 1 1 1 2 2 2 2 1 2 2 1 2
##  [647] 2 1 1 1 2 2 2 1 1 2 2 2 2 2 2 2 1 2 1 2 2 1 2 1 1 2 2 1 2 2 2 1 1 2
##  [681] 1 1 1 2 2 2 1 1 1 1 2 1 1 2 2 2 2 1 1 1 1 2 1 2 1 2 1 2 2 2 1 2 2 1
##  [715] 2 2 1 1 2 1 2 1 2 2 2 2 2 2 1 2 2 2 1 2 1 2 1 2 2 2 2 1 1 2 1 1 2 2 1
##  [749] 1 2 2 1 2 2 2 1 2 2 2 2 2 1 2 2 1 2 1 1 1 1 2 1 1 2 1 2 2 2 2 1 1 2 2
##  [783] 1 2 1 2 1 1 2 2 2 1 1 2 2 1 1 2 2 2 2 1 2 1 2 2 2 2 2 1 1 2 2 2 2 2 2
##  [817] 1 2 2 2 1 1 1 2 1 1 1 2 2 2 2 1 1 2 2 1 2 1 1 2 2 2 1 2 2 2 1 2 1 1
##  [851] 1 2 1 1 1 2 2 1 2 1 2 1 1 1 2 2 2 1 2 1 1 2 1 1 2 2 2 2 1 2 2 1 1 2
##  [885] 1 2 2 2 2 1 1 2 1 1 1 2 1 2 1 2 2 1 2 1 2 1 2 2 1 2 2 2 1 2 2 2 2 2
##  [919] 2 2 1 1 2 2 2 1 2 1 1 1 2 1 1 2 2 1 1 2 2 2 1 2 2 1 2 2 2 1 2 1 2 2
##  [953] 2 2 2 1 2 2 2 1 2 1 1 1 2 2 1 2 1 1 2 1 1 1 2 2 1 2 2 2 2 1 2 2 2 2
##  [987] 2 2 2 2 1 2 2 2 2 2 1 2 2 2
```

```
plot(res,"classification")
```



Q4 Find a partition of the simulated data into two classes using kmeans

```
kmeans(samples, 2) -> res.kmean
plot(samples, col = c("blue", "orange")[ res.kmean$cluster])
```



###Q5 Compare the two partitions (from kmeans and mclust). Comment your result. We can see that the results of kmeans and mclust are different. K-Means separate almostly the data by a straight line, but mclust separate the data by a circle. For our situation, mclust perform better.

E3

Q1 Detail the computation of the tqik = E[Zik] with respect to p q(Z|X) where Zik = I(Zi=k).

$$\mathbb{P}(x_i|Z_{ik}=1)) = \mathbb{N}_p(x_i|\mu_k,\Sigma_k) = \frac{1}{(2\pi)^{\frac{p}{2}} \times |\Sigma_k|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k))$$

```
mvpnorm<-function(x,mu,Sigma){

  p<-length(x)
   return( as.numeric(exp(-1/2*((x-mu)%*%solve(Sigma)%*%t(t(x-mu))))/det(Sigma)^(1/2)/(2*pi)^(p/2)))
}
mvpnorm(c(0,0), mu = c(0,0),Sigma = matrix(c(1,0,0,1),2,2))
```

```
## [1] 0.1591549
```

$$\begin{aligned}
t_{ik} &= \mathbb{E}_{Z_{ik}|x_i}[Z_{ik}=1] \\
&= \mathbb{P}(Z_{ik}=1|x_i) \\
&= \frac{\mathbb{P}(Z_{ik}=1,x_i)}{\mathbb{P}(x_i)} \\
&= \frac{\mathbb{P}(Z_{ik}=1)\times\mathbb{P}(x_i|Z_{ik}=1)}{\sum_{l=1}^K \mathbb{P}(Z_{il}=1)\times\mathbb{P}(x_i|Z_{il}=1)} \\
&= \frac{\pi_k \times \frac{1}{2\pi^{\frac{p}{2}}\times|\Sigma_k|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k))}{\sum_{l=1}^K \pi_l \times \frac{1}{2\pi^{\frac{p}{2}}\times|\Sigma_l|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i-\mu_l)^T\Sigma_l^{-1}(x_i-\mu_l))}
\end{aligned}$$

Q2 Express Q(q|) the expectation of the complete log-likelihood with respect to p q (Z|X).

$$\begin{aligned}
Q(\theta^q|\theta) &= \mathbb{E}_{Z|x}[log\mathbb{P}_\theta(x,Z)|x,\theta^q] \\
&= \mathbb{E}_{Z|x}[\sum_{i=1}^n log\Pi_{k=1}^K \mathbb{P}(x_i,Z_{ik})^{Z_{ik}}] \\
&= \mathbb{E}_{Z|x}[\sum_{i=1}^n\sum_{k=1}^K Z_{ik}log(\pi_k\times\mathbb{P}(x_i|Z_{ik}=1))] \\
&= \sum_{i=1}^n\sum_{k=1}^K \mathbb{E}_{Z|x}[Z_{ik}=1,\theta^q]\times log(\pi_k\mathbb{P}(x_i|Z_{ik}=1)) \\
&= \sum_{i=1}^n\sum_{k=1}^K t_{ik}^q \times log(\pi_k\mathbb{P}(x_i|Z_{ik}=1)) \\
&= \sum_{i=1}^n\sum_{k=1}^K t_{ik}^q \times [log(\pi_k) - \frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k) - \frac{p}{2}log(2\pi) - \frac{1}{2}log(|\Sigma_k|)]
\end{aligned}$$

Q3 Detail the computation of q+1 = argmax Q(q|)

$$\begin{aligned}
\pi^{q+1} &= argmax_{\pi^q} Q(\theta^q|\theta) \\
&= argmax_{\pi^q} \sum_{i=1}^n\sum_{k=1}^K t_{ik}^q \times log(\pi_k)
\end{aligned}$$

$$\pi_j^{q+1} = \frac{1}{n}\sum_{i=1}^{n} t_{ij}^q$$

$$(\mu_j^{q+1}, \Sigma_j^{q+1}) = argmax_{\mu_j^q, \Sigma_j^q} Q(\theta^q|\theta)$$

$$= argmax_{\mu_j^q, \Sigma_j^q} \sum_{i=1}^{n} t_{ij}^q \times [-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j) - \frac{1}{2} log(|\Sigma_j|)]$$

$$\mu_j^{q+1} = \frac{\sum_{i=1}^{n} t_{ij}^q \times x_i}{\sum_{i=1}^{n} t_{ij}^q}$$

$$\Sigma_j^{q+1} = \frac{\sum_{i=1}^{n} t_{ij}^q \times (x_i - \mu_j^{q+1})^T (x_i - \mu_j^{q+1})}{\sum_{i=1}^{n} t_{ij}^q}$$

Q4 Write the pseudo-code of an EM algorithm for estimating .

Init:Create the θ randomly; Repeat:(for iteration small than the max iteration) Compute the matrix T, compute the number Q; Compute the new θ by the matrix T; if ΔQ small than ξ: break; For each observation of data, compute the probability of their belongs to every classes; Chose the class with the max probability;

Q5 Write a E-step function that produces the tik from . Check the results by injecting the real parameters of your simulation and comparing the tik estimated against the latent variables Z in your simulation.

```
require("mvtnorm")
```

```
## Loading required package: mvtnorm
```

```
## Warning: package 'mvtnorm' was built under R version 3.4.4
```

```
Estep<-function(X,pi,mu,sigma,K){

  n<-nrow(X)
  p<-ncol(X)
  T<-matrix(0,ncol=K,nrow=n)


  for(i in 1:n){
     sum<-0
  for(l in 1:K){
    sum=sum+pi[l]*dmvnorm(X[i,], mean = mu[,l],sigma = sigma[,l*p-p+1:p])
  }

    for(k in 1:K){
      T[i,k]=pi[k]*dmvnorm(X[i,], mean = mu[,k],sigma = sigma[,k*p-p+1:p])/sum

    }

  }


  Q<-0
```

```
   for(i in 1:n){
       for(k in 1:K){
         Q=Q+T[i,k]*log(pi[k]*dmvnorm(X[i,], mean = mu[,k],sigma = sigma[,k*p-p+1:p]))
       }
   }

   result <- list(T=T, Q=Q)
  return(result)
}
```

Q6 Write a M-step function that produces q+1 from the sample and the tqiks.

```
Mstep<-function(X,T,pi,mu,sigma,K){
   p<-nrow(mu)
  n<-nrow(T)
  for(j in 1:K){

  sum<-0
  for(i in 1:n){
    sum=sum+T[i,j]
  }
    pi[j]=sum/n


    sum1<-vector(length=ncol(X))
     for(i in 1:n){
    sum1=sum1+T[i,j]*X[i,]
  }
    mu[,j]=sum1/sum


   sum2<-matrix(0,p,p)
   for(i in 1:n){
    sum2=sum2+T[i,j]*(t(X[i,]-t(mu[,j]))%*%(X[i,]-t(mu[,j])))
   }
   for(i in 1: p){
     sigma[,j*p-p+i]=sum2[,i]/sum
   }
  }
   result <- list(pi=pi, mu=mu,Sigma=sigma)
  return(result)
}
```

Q7 Program the EM algorithm (you could check that each step increases the log-likelihood.)

```
data("iris")
iris<-iris[,1:4]
EM<-function(data,k){
  X<-as.matrix(data)
  p<-ncol(X)
   n<-nrow(X)
```

```
  pi<-vector(length=k)
  mu<-matrix(runif(k*p),ncol=k,nrow=p)
  Sigma<-diag(p)*abs(1*10)
  for(i in 1:(k-1)){
    Sigma=cbind(Sigma,diag(p)*abs(i+1)*10)
  }

  for(i in 1:k){
    pi[i]=1/k
  }

  T<-matrix(0,ncol=k,nrow=n)
  Q<-2
  Q.new<-1
  max_iteration<-10
  for(i in 1:max_iteration){
    Q=Q.new
  Eres<-Estep(X,pi=pi,mu=mu,sigma=Sigma,K=k)
  T<-Eres$T
  Q.new<-Eres$Q

  #print(Q.new)

  Mres<-Mstep(X,T,pi=pi,mu=mu,sigma=Sigma,K=k)
  mu<-Mres$mu
  pi<-Mres$pi
  Sigma<-Mres$Sigma


  #print(Sigma)

  }
  res<-vector(length=n)

  for(i in 1:n){
    un<-c(0,0,0)
    for(j in 1:k){
      un[j]<- dmvnorm(X[i,], mean = mu[,j],sigma = Sigma[,j*p-p+1:p])
    }
    res[i]=which.max(un)
  }
  return(res)


}
EM(iris[,1:4],3)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 2 2 2 2 2 2 3 2 3 2 2 2 2 2 3 2
##  [71] 3 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 3 3 2 3 3
## [106] 2 3 2 2 3 3 3 3 3 3 3 2 3 2 2 3 3 2 3 3 3 3 3 3 2 2 3 3 2 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3
```

E4 Exercise 4 Data iris

Q1 Run your algorithm with the iris dataset and compare the results to the one obtained using the kmeans algorithm.

```
res<-kmeans(iris,3)
res$cluster
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [71] 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1 3 1
## [106] 1 3 1 1 1 1 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1 1 3
## [141] 1 1 3 1 1 1 3 1 1 3
```

Q2 Comment

We can see that, the result of my EM function is similar with the result of kmeans. So it's a good function.