Mélange de Bernoulli

Fatma Mahfoudh

24/10/2018

Modéle

Considérons un vecteur aléatoire binaire $\boldsymbol{x} \in [0,1]^p$ de p variables x_j suivant chacune une distribution de Bernoulli $\mathcal{B}(\mu_j)$. La distribution du vecteur s'exprime comme:

$$p(\boldsymbol{x}|\boldsymbol{\mu}) = \prod_{j=1}^{p} \mu_j^{x_j}(1-\mu_j)^{1-x_j},$$

avec $\boldsymbol{x} = (x_1, \cdots, x_p)^T$ et $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_p)^T$.

Soit une distribution mélange à K composantes de Bernoulli

$$p(\boldsymbol{x}|\boldsymbol{\pi}, \boldsymbol{M}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}|\boldsymbol{\mu}_k)$$

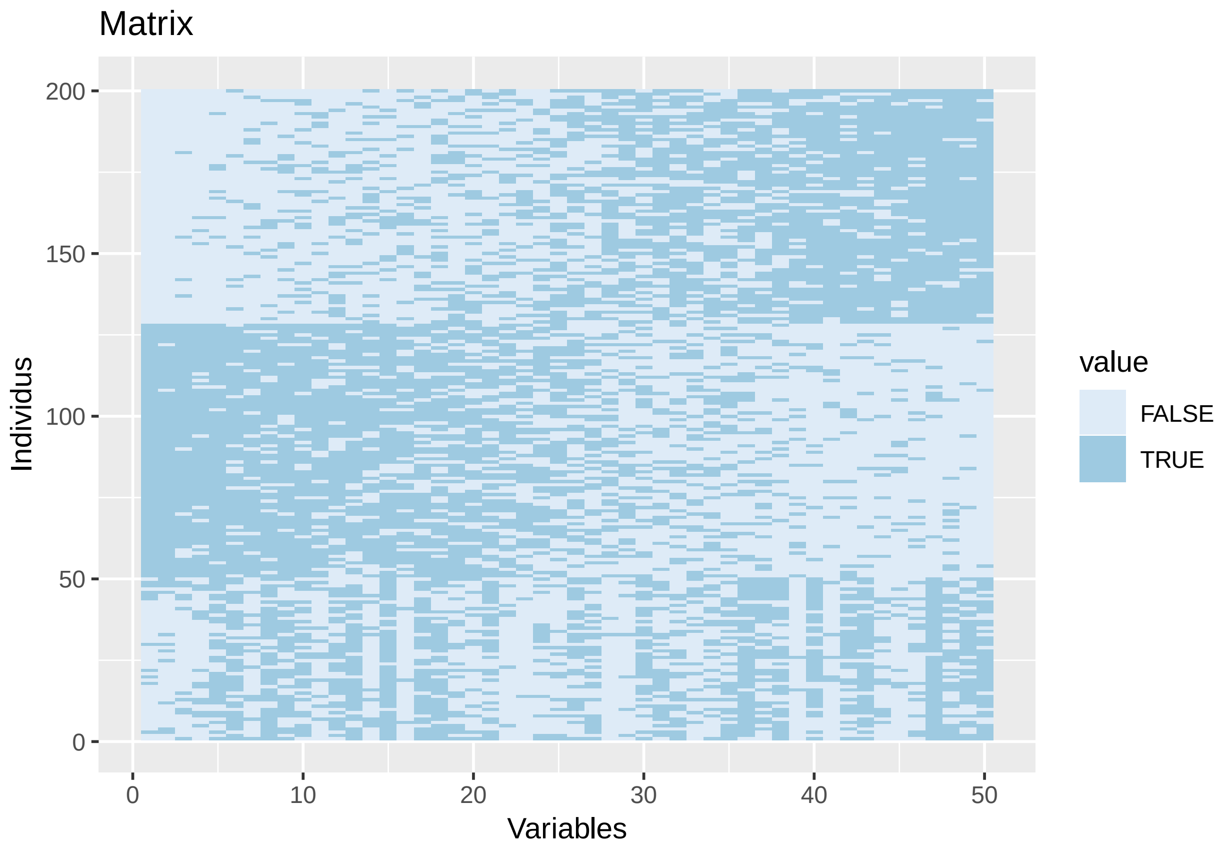
où les π_k sont les proportions du mélange et les $p(\boldsymbol{x}|\boldsymbol{\mu}_k)$ sont des distributions de Bernoulli multivariées de paramètres $\boldsymbol{\mu}_k = (\mu_{k1}, \cdots, \mu_{kp})^T$, et $M = \{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K\}^T$ la matrice des paramètres des densités de classes.

Dans la suite nous considérerons

- un échantillon observé $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ issu de cette distribution mélange,
- des variables latentes $Z = \{z_1, \cdots, z_n\}$ indiquant la composante d'origine de chaque $\boldsymbol{x}_i$.

Simulation

```r
set.seed(3)
K<-3
p<-50
n<-200
pi<-c(1/3,1/3,1/3)
M<-matrix(runif(K*p),K,p)
M[K,]<-1-M[1,] #é
nks<-rmultinom(1,200,prob = pi)
Z<-rep(1:length(nks),nks)
X <-do.call(rbind,
                  mapply(function(nk,k){
                    matrix(rbernoulli(nk*p,p=M[k,]),
                           nrow = nk,
                           ncol=p,
                           byrow = TRUE)}, nks,1:K))

kmeans(X,3,nstart = 10)->res.kmeans
tidyData<-melt(X[order(res.kmeans$cluster),order(M[1,])])

ggplot(tidyData, aes(x = Var2, y = Var1)) +
  geom_raster(aes(fill=value)) +
  scale_fill_brewer(aesthetics = "fill") +
  labs(x="Variables", y="Individus", title="Matrix")
```



Exercice 2

1. Ecrire la log-vraisemblance complete.

$$p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta} = (\pi, M)) = \prod_{i=1}^{n}\prod_{k=1}^{K} \pi_k^{z_{ik}} p(\boldsymbol{x_i}|\boldsymbol{\mu_k})^{z_{ik}}$$

$$avec\ p(\boldsymbol{x_i}|\boldsymbol{\mu_k}) = \prod_{j=1}^{p} \mu_{kj}^{x_{ij}}(1-\mu_{kj})^{1-x_{ij}}$$

$$=> ln(p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta} = (\pi, M))) = \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}(ln(\pi_k) + \sum_{j=1}^{p} x_{ij}ln(\mu_{kj}) + (1-x_{ij})ln(1-\mu_{kj}))$$

2. Exprimer $t_{q,ik} = E[Z_{ik}]$

$$E(Z_{ik}) = \sum_{z_{nk}} z_{nk} p_{\theta_q}(z_{nk}|x_i)$$

$$= \sum_{z_{nk}} z_{nk} \frac{p_{\theta_q}(x_i|z_{nk}, \theta_q)p(z_{nk})}{P_{\theta_q}(x_i)}$$

avec $z_{nk} = 0\, ou\, 1$

$$= \frac{\pi_k p(x_i|\mu_k)}{\sum_{j=1}^{K} \pi_j p(x_i|\mu_j)}$$
$$= t_{ik}$$

3. Ecrire l'esperance de cette log-vraisemblance

$$Q(\theta_q|\theta) = E(ln(p(X,Z|\theta)))$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{K} t_{ik}(ln(\pi_k) + \sum_{j=1}^{p} x_{ij}ln(\mu_{kj}) + (1-x_{ij})ln(1-\mu_{kj})))$$

4. Donner la forme de θ_{q+1}

On dérive Q par rapport é μ_{kj}

$$\frac{dQ}{d\mu_{kj}} = 0$$

$$=> \sum_{i=1}^{n} t_{ik}\frac{x_{ij}-\mu_{kj}}{\mu_{kj}(1-\mu_{kj})} = 0$$

$$=> \mu_{kj}\sum_{i=1}^{n} t_{ik} = \sum_{i=1}^{n} t_{ik}x_{ij}$$

$$=> \mu_{kj} = \frac{\sum_{i=1}^{n} t_{ik}x_{ij}}{\sum_{i=1}^{n} t_{ik}}$$

En ce qui concerne les π_k, il faut prendre en considération la contrainte $\sum \pi_k = 1$, on introduit donc λ et on pose

$$L = Q + \lambda(\sum_{i=1}^{K} \pi_k - 1)$$

On dérive L par rapport à π_k

$$\frac{dL}{d\pi_k} = 0$$

$$=> \sum_{i=1}^{n} \frac{t_{ik}}{\pi_k} = -\lambda$$

$$=> \pi_k\lambda = -\sum_{i=1}^{n} t_{ik}$$

On somme sur les k

$$=> \lambda = -\sum_{k=1}^{K}\sum_{i=1}^{k} t_{ik}$$

On l'injecte dans l'équation précedente pour obtenir

$$=> \pi_k = \frac{\sum_{i=1}^{n} t_{ik}}{\sum_{k=1}^{K}\sum_{i=1}^{k} t_{ik}}$$

5. Detailler les etapes de l'algorithme EM qui permet d'estimer θ.

$$E - step : Calculer\, Q(\theta_{old})\, ce\, qui\, revient\, é\, calculer\, d'abord\, t_{ik}$$

$$t_{ik} = \frac{\pi_k p(x_i|\mu_k)}{\sum_{j=1}^{K} \pi_j p(x_i|\mu_j)}$$

$$M - step :\ détermination\, de\, \theta_{new}$$

$$\mu_{kj} = \frac{\sum_{i=1}^{n} t_{ik} x_{ij}}{\sum_{i=1}^{n} t_{ik}}$$

$$\pi_k = \frac{\sum_{i=1}^{n} t_{ik}}{\sum_{k=1}^{K} \sum_{i=1}^{k} t_{ik}}$$

6.

$$-E(ln(p_{\theta_{q+1}}(Z|X))) = -\sum_{z} ln(p_{\theta_{q+1}}(Z|X)) p_{\theta_{q+1}}(Z|X)$$

$$= -\sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik,q+1} ln(t_{ik,q+1})$$

7.

$$P_\theta(Z|X) = \frac{P_\theta(Z,X)}{P_\theta(X)}$$

$$=> Ln(P_\theta(X|\theta)) = Q(\theta|\theta) - E(ln(P_\theta(Z|X)))$$

$$=> Ln(P_\theta(X|\theta)) = \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}(ln(\pi_k) + \sum_{j=1}^{p} x_{ij} ln(\mu_{kj}) + (1-x_{ij}) ln(1-\mu_{kj}))) - \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik,q} ln(t_{ik,q})$$

8.

$$BIC(K) = ln(P_{\theta_k}(X)) - \frac{d_k}{2} ln(n)$$

$$=> BIC(K) = Q(\theta|\theta) - E(ln(P_\theta(Z|X))) - \frac{Kp + K - 1}{2} ln(n)$$

$$=> BIC(K) = Q(\theta|\theta) - \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik} ln(t_{ik}) - \frac{Kp + K - 1}{2} ln(n)$$

9.

D'aprés le cours,

$$ICL(K) = BIC(K) + E(ln(P_\theta(Z|X)))$$

$$=> ICL(K) = Q(\theta|\theta) - \frac{Kp + K - 1}{2} ln(n)$$

10.

$$1)Initialiser\,\theta_{old}$$

$$2)Tant\,que\,\theta_{new}! = \theta_{old}\,ou\,it\acute{e}ration! = it\acute{e}ration_{max}\,R\acute{e}p\acute{e}ter$$

$$E-step : Calculer\,Q(\theta_{old})\,ce\,qui\,revient\,\acute{e}\,calculer\,d'abord\,t_{ik}$$

$$t_{ik} = \frac{\pi_k p(x_i|\mu_k)}{\sum_{j=1}^K \pi_j p(x_i|\mu_j)}$$

$$M-step :\ d\acute{e}termination\,de\,\theta_{new}$$

$$\mu_{kj} = \frac{\sum_{i=1}^n t_{ik}x_{ij}}{\sum_{i=1}^n t_{ik}}$$

$$\pi_k = \frac{\sum_{i=1}^n t_{ik}}{\sum_{k=1}^K \sum_{i=1}^k t_{ik}}$$

11.

$$1)Initialiser\,\theta_{old}$$

$$2)Tant\,que\,\theta_{new}! = \theta_{old}\,ou\,it\acute{e}ration! = it\acute{e}ration_{max}\,R\acute{e}p\acute{e}ter$$

$$E-step : Calculer\,Q(\theta_{old})\,ce\,qui\,revient\,\acute{e}\,calculer\,d'abord\,t_{ik}$$

$$t_{ik} = \frac{\pi_k p(x_i|\mu_k)}{\sum_{j=1}^K \pi_j p(x_i|\mu_j)}$$

$$C-step : Z = max_{index}(T)$$

$$M-step :\ d\acute{e}termination\,de\,\theta_{new}$$

$$\mu_{kj} = \frac{\sum_{i=1}^n t_{ik}x_{ij}}{\sum_{i=1}^n t_{ik}}$$

$$\pi_k = \frac{\sum_{i=1}^n t_{ik}}{\sum_{k=1}^K \sum_{i=1}^k t_{ik}}$$

Exercice 3

E-step

```
#Pi matrice à k lignes et 1 colonne
E_step <- function(X,M,Pi){

  n <- dim(X)[1]
  K <- dim(M)[1]
  p <- dim(M)[2]

  #l'écriture ci dessous permet d'avoir p(xi|muk) pour tout i et k
  #mais son inconvénient est qu'elle diverge donc le cas où mu=0 ou 1
  #dans ce cas on utilisera la formule initiale avec les produits
```

```r
  #P_x_mu <- exp( X%*%log(t(M)) + (1-X)%*%log(t(1-M)) )
  P_x_mu <- NULL
  for (k in 1:K){
    pk <- mapply(function(i){prod(M[k,]^X[i,]*(1-M[k,])^(1-X[i,]))},1:n)
    P_x_mu <- cbind(P_x_mu,pk)
  }
  colnames(P_x_mu) <- NULL

  #on construit une matrice de taille n*K
  #en répétant le vecteur ligne transposé de Pi
  Pi_rep_n <- matrix(unlist(rep(t(Pi),each=n)),ncol=K)

  #on construit une matrice de taille K*K
  #en répétant le vecteur colonne Pi
  Pi_rep_K <- matrix(unlist(rep(Pi,K)),ncol=K)

  t <- (Pi_rep_n*P_x_mu)/(P_x_mu%*%Pi_rep_K)

}
```

Vérification:

```r
Pi <- matrix(pi,nrow=K,ncol=1)
t <- E_step(X,M,Pi)

max_index <- function(x){
  which(x==max(x))
}
Zt <- apply(t,1,max_index)
identical(Zt,Z)
```

```
## [1] TRUE
```

Les t_{ik} estimé sont donc identiques aux variables latentes Z.

M-step

```r
M_step <- function(X,t){

  n <- dim(X)[1]
  p <- dim(X)[2]
  K <- dim(t)[2]

  #Pi: renvoie une ligne et k colonnes
  Pi <- colSums(t)/n

  #M: K lignes , p colonnes
  t_scol_rep <- matrix(unlist(rep(colSums(t),p)),nrow=K)
  M <- (t(t)%*%X)/t_scol_rep

  return(list("pi"=Pi, "M"=M))

}
```

Vérification:

```
res <- M_step(X,t)
Pi_em <- matrix(unlist(res["pi"]))
M_em <- matrix(unlist(res["M"]),nrow=K)

print("RMSE Pi-Pi_em")
```

```
## [1] "RMSE Pi-Pi_em"
```

```
print(sum(apply(Pi-Pi_em,1,function(x){x^2})))
```

```
## [1] 0.01085243
```

On peut dire que les valeurs estimées de Pi ne sont pas éloignées des valeurs de Pi

3&4. Ecrire l'algorithme EM qui estime les paramétres d'un mélange de Bernoulli en K classes. Tracer l'évolution de la vraisemblance é chaque demi-étape (E et M) lorsque vous appliquez l'algorithme aux données simulées.

```
#Calcul de Q
comp_Q <-function(X,t,M,Pi){

  n <- dim(X)[1]
  K <- dim(M)[1]
  p <- dim(X)[2]

  Pi_rep_n <- matrix(unlist(rep(t(Pi),each=n)),ncol=K)

  epsilon <- 1e-5
  Q <- sum(t*(log(Pi_rep_n)+X%*%log(t(M)+epsilon)+(1-X)%*%log(t(1-M)+epsilon)))

}
```

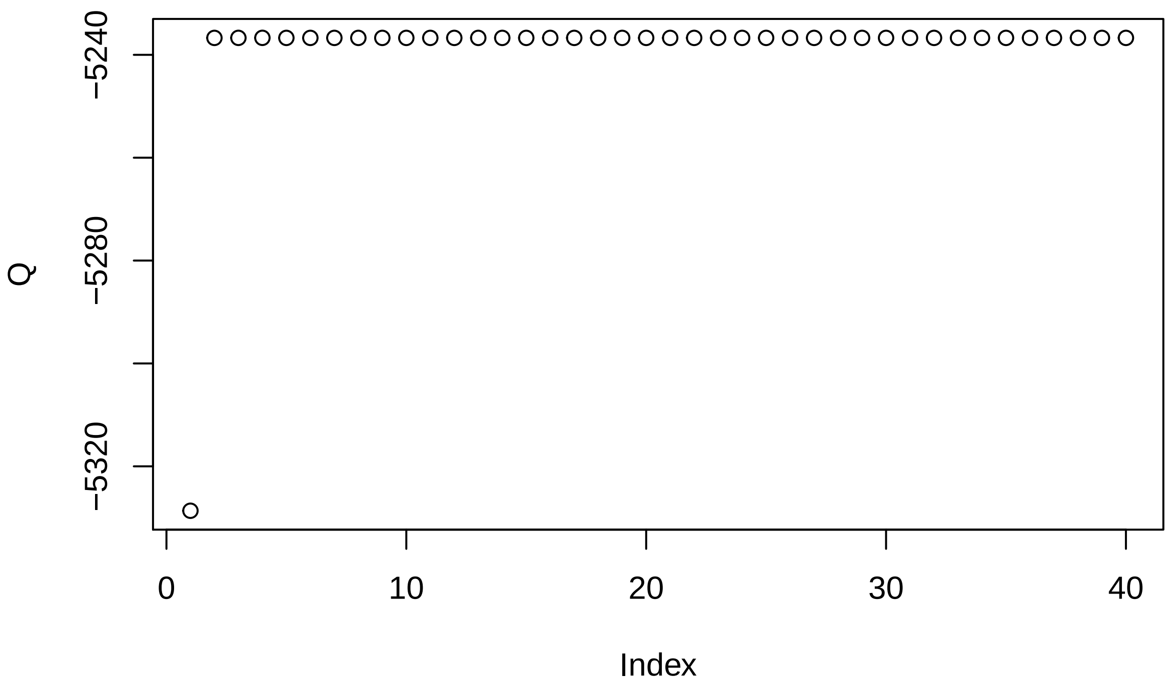
L'ajout du ϵ est utile pour continuer à calculer Q lorsqu'on a des valeurs nulles ou égales à 1 dans M

```
EM <- function(X,K){

  set.seed(3)

  n <- dim(X)[1]
  p <- dim(X)[2]

  #Initialisation de Theta
  Pi_old <- matrix(1/K,nrow=K,ncol=1)
  M_old <- matrix(runif(K*p),nrow=K,ncol=p)

  M_old[K,]<-1-M_old[1,]

  #i: itération demi-étape
  i <- 0
  #j: itération étape EM
  j <- 1
  Q <- vector()

  while (j<=20){

    #Expectation
```

```r
    t <- E_step(X,M_old,Pi_old)

    #Qold
    i <- i+1
    Q[i] <- comp_Q(X,t,M_old,Pi_old)

    #Maximization
    res <- M_step(X,t)
    Pi_new <- matrix(unlist(res["pi"]))
    M_new <- matrix(unlist(res["M"]),nrow=K)

    if (identical(M_new,M_old) & identical(Pi_new,Pi_old)){
      break
    } else{
      #Qnew
      i <- i+1
      Q[i] <- comp_Q(X,t,M_new,Pi_new)

      j <- j+1
      M_old <- M_new
      Pi_old <- Pi_new
    }

  }

  return(list("m"=M_old,"pi"=Pi_old,"t"=t,"q"=Q))
}
```

Un nombre d'itérations à 20 est suffisant (On verra que Q converge assez vite)

```r
res_EM <- EM(X,K)
M_EM <- matrix(unlist(res_EM["m"]),ncol=K)
Pi_EM <- matrix(unlist(res_EM["pi"]))
t <- matrix(unlist(res_EM["t"]),ncol=K)
Q <- matrix(unlist(res_EM["q"]))
plot(Q)
```

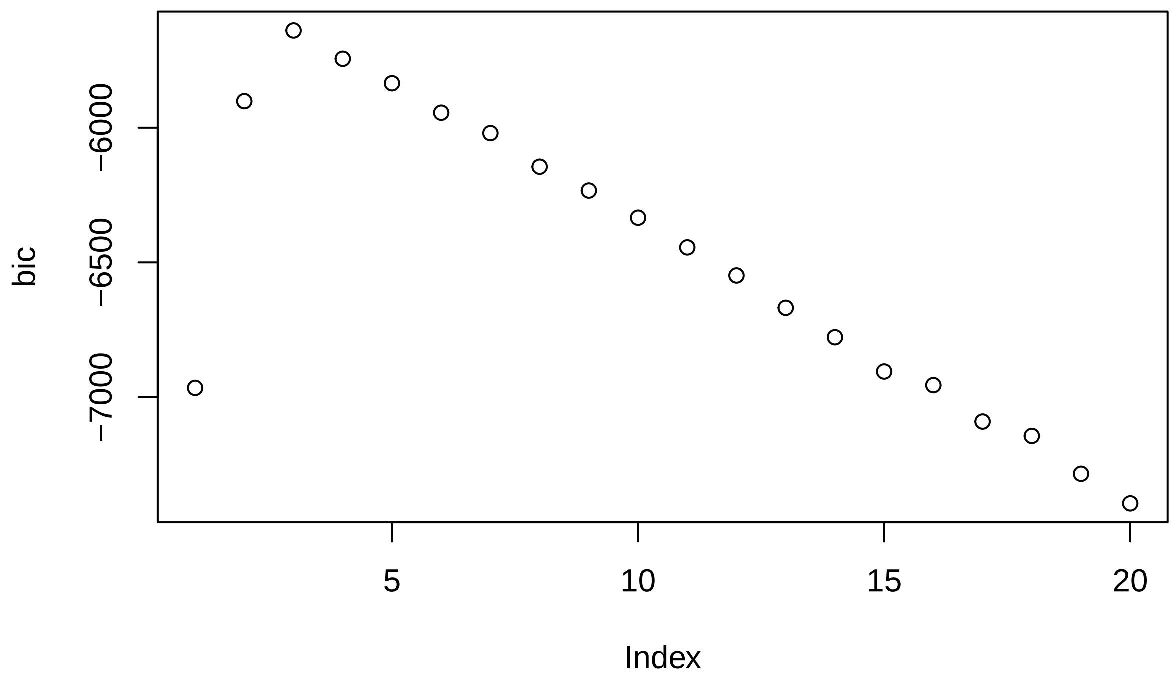


On observe bien une log vraisemblance croissante qui atteint un plateau.

5. Programmer la fonction BIC qui prend la sortie de votre algorithme EM et rend le critére BIC.

```
comp_BIC <- function(X){
  bic <- vector()
  p <- dim(X)[2]
  n <- dim(X)[1]

  for (k in 1:20){

    res <- EM(X,k)
    t <- matrix(unlist(res["t"]),ncol=k)
    Q <- matrix(unlist(res["q"]))

    epsilon <- 1e-5
    ln_t <- log(t+epsilon)
    bic[k] <- Q[length(Q)]-sum(t*ln_t)-(k*p+k-1)*log(n)/2
  }
  plot(bic)
}
```

```
comp_BIC(X)
```

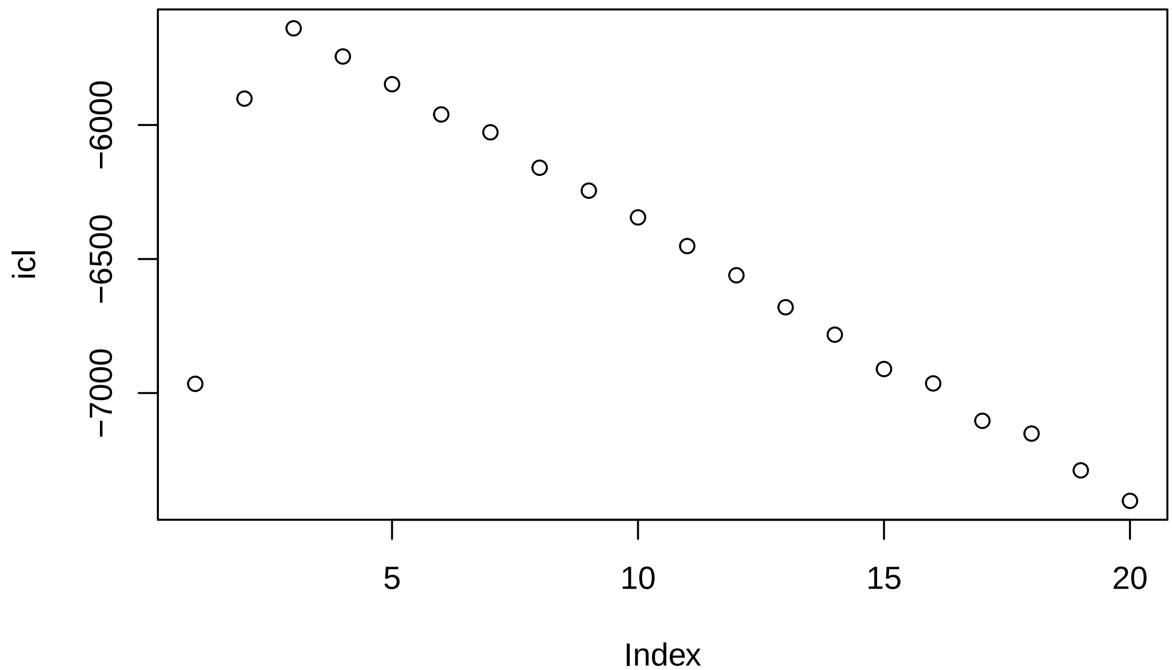


Le maximum de BIC est atteint pour k=3 (résultat attendu).

6. Programmer la fonction ICL qui prend la sortie de votre algorithme EM et rend le critére ICL.

```
comp_ICL <- function(X){
  icl <- vector()
  p <- dim(X)[2]
  n <- dim(X)[1]

  for (k in 1:20){
    res <- EM(X,k)
    Q <- matrix(unlist(res["q"]))

    icl[k] <- Q[length(Q)]-(k*p+k-1)*log(n)/2
  }
  plot(icl)
}
```

```
comp_ICL(X)
```



Le maximum de ICL est atteint également pour k=3 (résultat attendu).

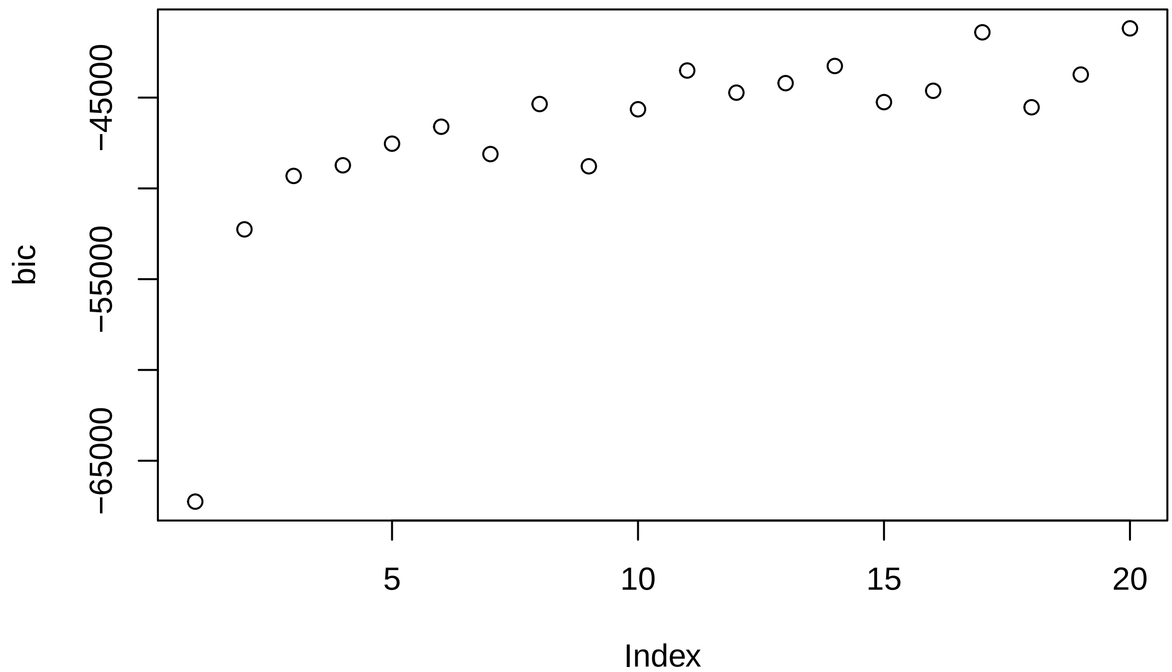
Exercice 4

Appliquer votre algorithme au jeu de données state-firearms sur les lignes et les colonnes et commentez.

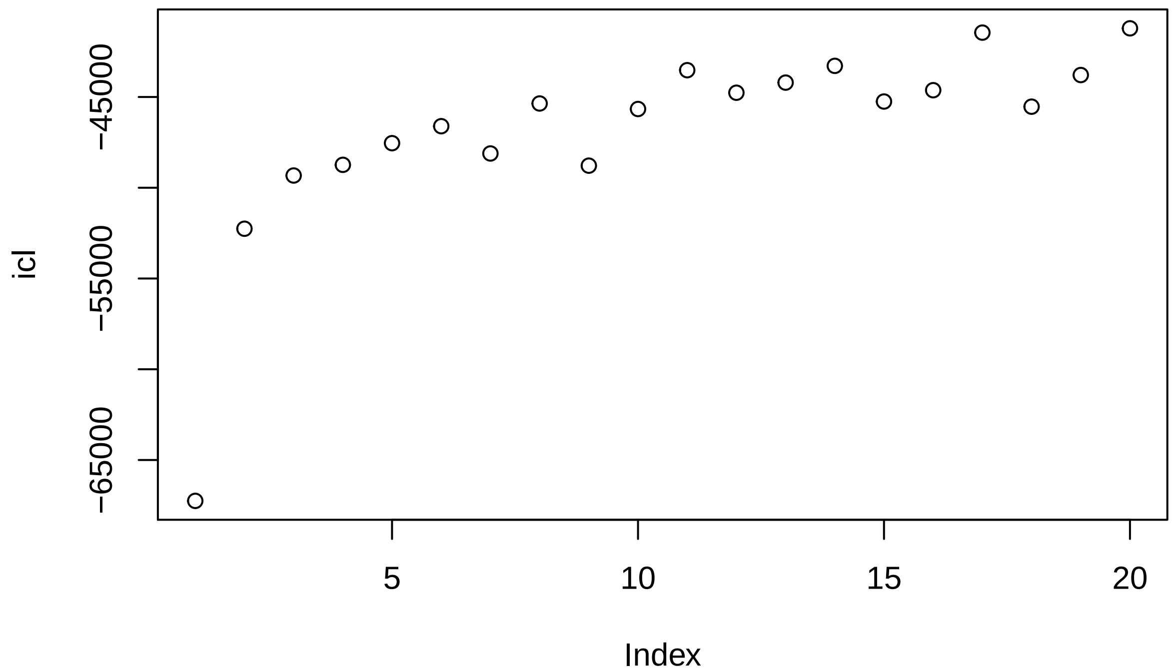
```
#le fichier csv "raw_data.csv" est à déposer dans un dossier state-firearms
#dans le working directory
df <- read.csv("state-firearms/raw_data.csv",header=T,sep=",")
```

```
X <- df
X$state <- NULL
X$year <- NULL
X$lawtotal <- NULL
X <- as.matrix(X)
```

```
comp_BIC(X)
```




```
comp_ICL(X)
```



Pour les 2 critéres, BIC et ICL atteignent le max pour K=17. On a donc 17 classes.