EM

Nicolas Desan, François Pedeboy

2022-12-01

Simulation

1) On simule un échantillon de taille n=100 de loi de Poisson de paramètre $\lambda = 3$:

```
n1<-100
lambda1<-3

x<-rpois(n1,lambda1)

print(x)
```

```
##   [1] 3 3 4 2 2 2 3 1 2 2 2 6 6 5 2 1 3 6 2 2 2 4 2 5 2 2 6 5 1 2 5 3 3 0 4 4 4
##  [38] 2 3 5 2 2 5 2 2 2 4 3 6 3 0 2 0 6 6 2 5 4 0 2 1 4 4 2 4 9 2 2 4 1 1 2 3 3
##  [75] 4 3 1 2 2 2 5 4 2 3 1 3 6 2 2 2 3 6 3 5 8 1 4 3 4 0
```

2) On simule un échantillon de taille $n_2 = 100$ de loi de Poisson de paramètre $\lambda_2 = 15$:

```
n2<-200
lambda2<-15

x2<-rpois(n2,lambda2)

print(x2)
```

```
##   [1] 19 16 16 14 19 16 12 16 12 13 13 11 16 16 17 19 11 14 15 13  9 19 13 16 20
##  [26] 12  8 20 11 15 13 16 11 17 20 13 22 12  7 12 20 15 19 21 15 13  9 20 16 13
##  [51] 22 25 12 16 13 16 26 17 18 14 15 10  9 17 13 14 14 12 17 16  9 18 13 18  9
##  [76] 10 24 12 18 17  8 20 22 14 12 14 19 16 20 22 17 13 12  8 20 15  9 16 17 19
## [101] 14 15 24 12 17 17 16 10 12 21  9 16 20 20 14 15  7 14 14 13 20 12 29 15 15
## [126] 13 20 25 12 22 13 25 25 18 14 15 13 15 15 12 19 16 19 16  6 19 10 21 14 12
## [151] 12 12 14 19  9 19 16 20 13 11 24 12 12 18 11 20 22 16 11 14 10 20 23 10 15
## [176] 16 10 20 13 21 16 22  7 21 14 21 15 14 13 23 16 12 18 22 16 20 22 16 16 16
```

3) On simule un vecteur de 300 valeurs entières (100 “1”, suivi de 200 “2”) :

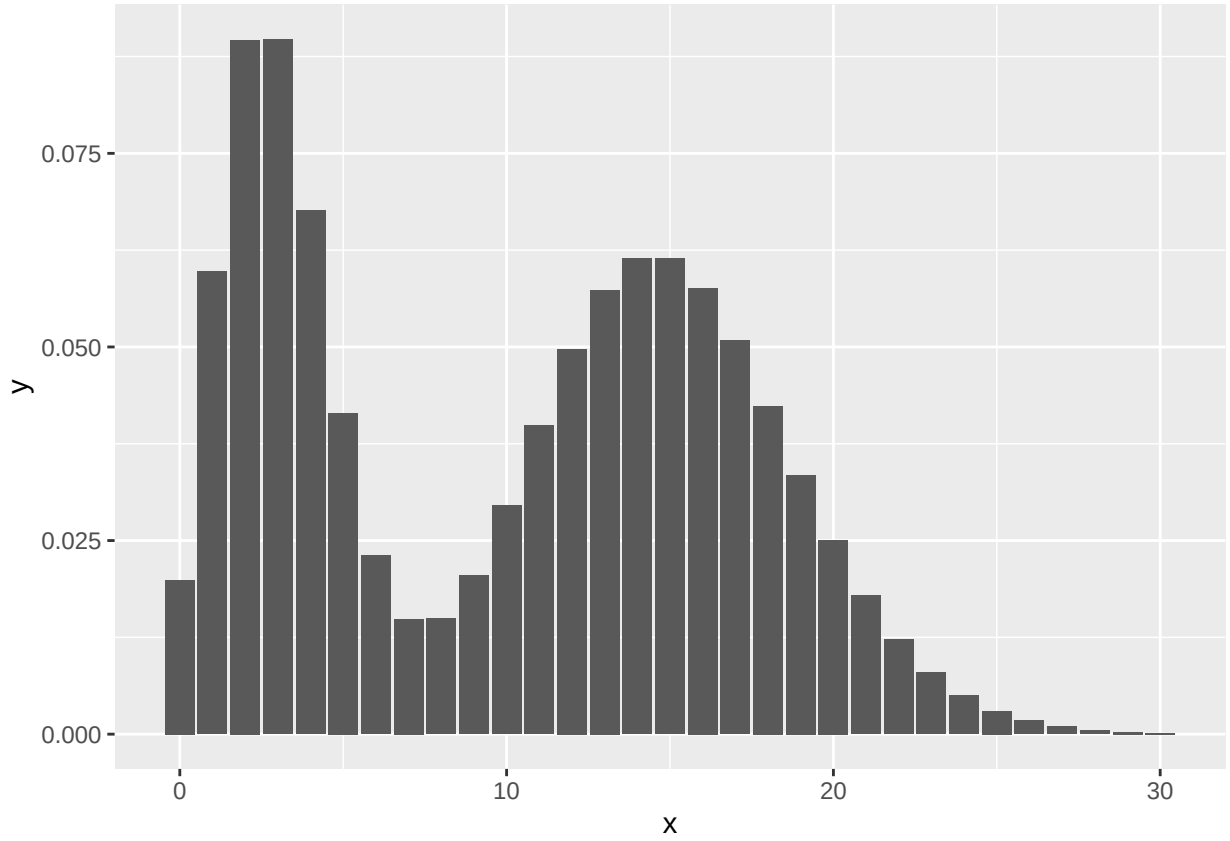
```
x3<-c()

for(i in 1:n1){
  x3[i]<-1
```

```
}

for(i in 1:n2){
  x3[i]<-2
}

print(x3)
```

```
##   [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
##  [38] 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
##  [75] 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
## [112] 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
## [149] 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
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

4) On simule un mélange de lois de Poisson à deux composantes $\lambda_1 = 3$ et $\lambda_2 = 15$ ainsi que des poids $\pi_1 = 0.4$ et $\pi_2 = 0.6$:

```
p1<-0.4
p2<-0.6

P <-function(x){
  y<-0
  y<-p1*exp(-lambda1)*(lambda1^x)/factorial(x) + p2*exp(-lambda2)*(lambda2^x)/factorial(x)
  return(y)
}

y_sim<-c()

for(i in 0:30){
  y_sim[i+1]<-P(i)
}

data_sim<-data.frame(x=seq(0,30),y=y_sim)

ggplot(data = data_sim, aes(x = x, y = y)) + geom_bar(stat="identity")
```



Algorithme EM pour un mélange de lois de Poisson à K composantes

Il est important de remarquer que la moyenne d'une loi de Poisson de paramètre λ vaut λ. Ainsi, l'algorithme EM qui généralement est appliqué sur des lois gaussiennes en réévaluant les valeurs des μ_i à chaque itération est aussi appliquable dans l'étude d'un mélange de lois de Poisson de manière similaire en réévaluant les λ_i (du modèle de Poisson) de la même manière que les μ_i (du modèle gaussien).

1) On a l'initialisation de l'algorithme EM qui prend en argument un échantillon X ainsi que K le nombre de composantes voulues et qui renvoie l'ensemble des $\pi_i = 1/K$ ainsi que des valeurs initiales aléatoires des λ_i :

```
Initialisation<-function(X,K=2){
 poids<-c()
 for(i in 1:K){
   poids[i]<-1/K
 }

 lambdas<-X[sample(1:nrow(X),K),1]
 return(list(pis=poids,lambdas=lambdas))
}
```

2) Pour l'étape E on cherche à calculer la matrice qui contient l'ensemble des nouveaux poids π_i pour chacune des valeurs de l'échantillon qu'on manipule. Ces nouveaux poids sont calculés ainsi :

$$\pi_{k.i}^{(r+1)} = \frac{\pi_{k.i}^{(r)} f_k(x_i, \theta_k^{(r)})}{\sum_{l=1}^{g} \pi_{l.i}^{(r)} f_l(x_i, \theta_l^{(r)})}$$

avec : r l'itération correspondante, $\pi_{k.i}$ le poids de la i-ème valeur dans le k-ème cluster, g le nombre de clusters et f_l la l-ème densité de loi de Poisson de paramètre θ_l

On obtient donc l'étape E de l'algorithme avec le code suivant qui prend en argument un échantillon X, des valeurs de λ et π contenues dans parameters (objet list) ainsi qu'une matrice T (initialisé à une matrice contenant que des 1 par la suite) et renvoie :

```
Etape_E<-function(X,parameters,T){
 for (k in 1:ncol(T)){
 T[,k]<-parameters$pis[k]*dpois(unlist(X),lambda=parameters$lambdas[k])
 }
 return(T=t(apply(T,1,function(x){x/sum(x)})))
}
```

3) L'étape de maximisation M consiste à recalculer les différents λ et π du modèle grâce à la matrice T obtenu précédemment lors de l'étape E.

Pour cela on calcule la moyenne empirique de l'ensemble des poids de chaque colonne de T pour obtenir la nouvelle valeur des π.

Pour obtenir la nouvelle valeur des λ on effectue le calcul suivant :

$\lambda_i = \frac{\sum_{j=1}^{n} \pi_j x_j}{\sum_{j=1}^{n} \pi_j}$

avec : n la taille de l'échantillon X qu'on manipule et λ_i la valeur de λ du i_ème cluster.

On obtient donc l'étape M avec le code suivant :

```
Etape_M<-function(X,T){
 pis = apply(T,2,mean)
 lambdas<-colSums(T*(as.matrix(X)%*%rbind(rep(1,K))))/colSums(T)
 return(list(pis=pis,lambdas=lambdas))
}
```

4) Le but de l'algorithme EM est de réaliser les étapes E et M et de calculer la différence entre les paramètres de l'étape r et r+1. On s'arrête lorsqu'on estime qu'il y a une convergence des paramètres.

Dans un premier temps on réalise un échantillon X du mélange précédent de la manière suivante :

```
nks<-rmultinom(1,300,c(0.4,0.6))
lambdas<-c(3,15)
x1<-rpois(nks[1],lambda=lambdas[1])
x2<-rpois(nks[2],lambda=lambdas[2])

X<-data.frame(x=c(x1,x2))
```

On a donc l'algorithme EM :

```r
K<-2

EM<-function(X,K){
 parameters<-Initialisation(X,K)
 parameters_tmp<-parameters
 parameters_tmp$pis<-rep(1,K)
 parameters_tmp$lambdas<-rep(1,K)
 T<-matrix(1,nrow(X),K)
 while(mean((unlist(parameters)-unlist(parameters_tmp))^2)>1e-16){
 parameters_tmp<-parameters
 T<-Etape_E(X,parameters,T)
 parameters<-Etape_M(X,T)
 }
 print(parameters)
 print(mean((unlist(parameters)-unlist(parameters_tmp))^2))
 return(list(T=T,parameters=parameters))
}
```

On obtient :

```
## $pis
## [1] 0.5 0.5
##
## $lambdas
## [1] 10.56 10.56
##
## [1] 0
```