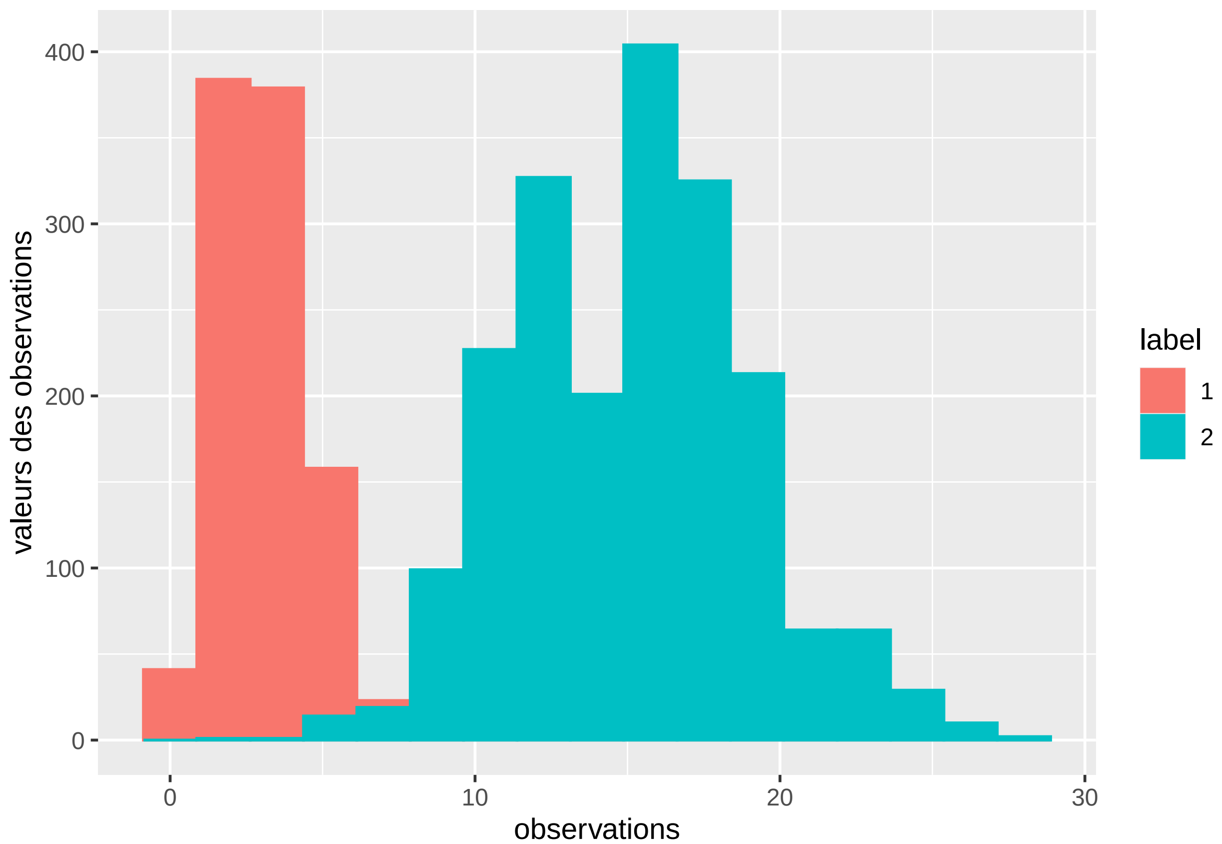
Projet algorithme EM

Manzione Amaury, Antoine Tireau

Partie 1

On simule deux lois de poisson comme cela est décrit dans l'énoncé. On augmente d'un facteur 10 la taille des données pour avoir une meilleur précision dans l'algorithme EM.

```r
library(ggplot2)

lambda1 <- 3
lambda2 <- 15
pi1 <- 1/3
pi2 <- 2/3

sample1 <- rpois(1000, lambda1)
sample2 <- rpois(2000, lambda2)

vector <- c(rep(1,1000), rep(2,2000))

data <- data.frame( obs = c(sample1,sample2), label=factor(vector))

ggplot(data,aes(x=obs,fill= label,color=label))+
  geom_histogram(position = "identity",bins = 17)+
  xlab("observations")+
  ylab("valeurs des observations")
```



Algorithme EM pour un mélange poissonien à K composantes

L'algorithme EM cherche à maximiser l'espérance de la log vraisemblance incomplète :

- l'expression de la log vraisemblance incomplète à l'étape p est :

$$L_{\theta^p}(X, Z, \theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} 1_{(Z_i=k|\theta^p)}(\log(\mathbb{P}(Z_i = k|X, \theta)) + \log(\mathbb{P}(X_i|Z_i = k, \theta)))$$

  où X_i sont les données observées,

$$\theta^p = \{\pi_1^p, .., \pi_K^p, \lambda_1^p, ..., \lambda_K^p\}$$

  les probabilités d'appartenance au cluster K et les valeurs des λ calculées à l'étape p, et Z_i sont des variables manquantes telles que si : $Z_i = 1$ $X_i \sim \mathcal{P}(\lambda_1)$ et réciproquement pour λ_K.

- On cherche donc à maximiser selon θ:

$$Q(\theta, \theta^0) = \mathbb{E}_{\theta^p}(L(X, Z, \theta)$$

$$Q(\theta, \theta^p) = \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^{(p)}(\log(\mathbb{P}(Z_i = k|X, \theta)) + \log(\mathbb{P}(X_i|Z_i = k, \theta)))$$

  avec $t_{ik}^p = \mathbb{P}(Z_i = k|\theta^p)$

$$Q(\theta, \theta^p) = \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^{(p)}(\log(\pi_k) + \log(\frac{\lambda_k^{X_i} \exp(-\lambda_k)}{X_i!}))$$

Initialisation

On initialise
$$\theta^0 = \{\pi_1^0, .., \pi_K^0, \lambda_1^0, ..., \lambda_K^0\}$$
avec des valeurs choisies au hasard.

```
lambdas_test <- c(8,10)
probs_test <- c(0.4,0.6)
```

Etape E

On calcule $Q(\theta, \theta^p)$ avec θ^p calculé à l'étape p. On calcule $t_{ik}^{(p)}$ avec la formule de Bayes pour les probabilités conditionnelles:
$$t_{ik}^p = \frac{\mathbb{P}(X_i|Z_i = k, \theta^p)\mathbb{P}(Z_i = k|X, \theta^p)}{\sum_{j=1}^K \mathbb{P}(X_j|Z_i = j, \theta^p)\mathbb{P}(Z_i = j|X, \theta^p)}$$

Soit:

$$t_{ik} = \frac{\frac{\lambda_k^{X_i}\exp(-\lambda_k)}{X_i!}\pi_k}{\sum_{j=1}^K \frac{\lambda_j^{X_i}\exp(-\lambda_j)}{X_i!}\pi_j}$$

```
# retourne la matriced des tik
Posterior <- function(X, lambdas, probs) {
  n = length(X) #nombre d' observations
  k = length(lambdas) # nombre de clusters
  Post <- matrix(0, nrow = n , ncol = k) # initalisation de la matrice
  for (i in 1:n) {
    x = X[i]
    liste = c()
    for (j in 1:k) {
      a = dpois(x, lambdas[j]) * probs[j]
      liste = append(liste, a)
    }
    Post[i,] = liste / sum(liste)
  }
  return(Post)
}
```

Etape M

On cherche maintenant à maximiser $Q(\theta, \theta^p)$ selon θ. Pour cela on calcule les dérivées partielles selon $\pi_1, ..\pi_K, \lambda_1, \lambda_K$. Pour simplifier on prend $K = 2$, car on cherche de toute façon 2 clusters.

- $$\frac{\partial Q(\theta, \theta^p)}{\partial \pi_1} = \sum_{i=1}^n \sum_{k=1}^2 t_{ik}^{(p)} \frac{\partial(\log(\pi_k) + \log(\frac{\lambda_k^{X_i}\exp(-\lambda_k)}{X_i!}))}{\partial \pi_1}$$

  $$\frac{\partial Q(\theta, \theta^p)}{\partial \pi_1} = \sum_{i=1}^n \sum_{k=1}^2 t_{ik}^{(p)} \frac{1}{\pi_k} = \sum_{i=1}^n t_{i1}^{(p)} \frac{1}{\pi_1} + t_{i2}^{(p)} \frac{1}{1-\pi_1}$$

On cherche π_1 tel que $\frac{\partial Q(\theta,\theta^0)}{\partial \pi_1} = 0$. On a donc:

$$\frac{\partial Q(\theta,\theta^p)}{\partial \pi_1} = 0 \iff \sum_{i=1}^{n} t_{i1}^{(p)} \frac{1}{\pi_1} + t_{i2}^{(p)} \frac{1}{1-\pi_1} = 0$$

$$\iff \pi_1 = \frac{\sum_{i=1}^{n} t_{i1}^{(p)}}{\sum_{i=1}^{n} t_{i1}^{(p)} + \sum_{i=1}^{n} t_{i2}^{(p)}} = \frac{\sum_{i=1}^{n} t_{i1}^{(p)}}{n}$$

- π_1 et π_2 jouent un rôle symétrique donc:

$$\pi_2 = \frac{\sum_{i=1}^{n} t_{i2}^{(p)}}{\sum_{i=1}^{n} t_{i1}^{(p)} + \sum_{i=1}^{n} t_{i2}^{(p)}} = \frac{\sum_{i=1}^{n} t_{i2}^{(p)}}{n}$$

-

$$\frac{\partial Q(\theta,\theta^p)}{\partial \lambda_1} = \sum_{i=1}^{n}\sum_{k=1}^{2} t_{ik}^{(p)} \frac{\partial(\log(\pi_k) + \log(\frac{\lambda_k^{X_i}\exp(-\lambda_k)}{X_i!}))}{\partial \lambda_1}$$

$$\frac{\partial Q(\theta,\theta^p)}{\partial \lambda_1} = \sum_{i=1}^{n} t_{i1}^{(p)} \frac{\partial(\log(\frac{\lambda_1^{X_i}\exp(-\lambda_1)}{X_i!}))}{\partial \lambda_1} = \sum_{i=1}^{n} \frac{t_{i1}^{(p)}}{X_i!}(\frac{\partial(X_i\log(\lambda_1) - \lambda_1)}{\partial \lambda_1}) = \sum_{i=1}^{n} \frac{t_{i1}^{(p)}}{X_i!}(\frac{X_i}{\lambda_1} - t_{i1})$$

On a :

$$\frac{\partial Q(\theta,\theta^p)}{\partial \lambda_1} = 0 \iff \sum_{i=1}^{n} \frac{t_{i1}^{(p)}}{X_i!}\frac{X_i}{\lambda_1} - t_{i1}^{(p)} = 0 \iff \sum_{i=1}^{n} t_{i1}^{(p)}\frac{X_i}{\lambda_1} - t_{i1}^{(p)} = 0 \iff \lambda_1 = \frac{\sum_{i=1}^{n} t_{i1}^{(p)} X_i}{\sum_{i=1}^{n} t_{i1}^{(p)}}$$

- λ_1 et λ_2 jouent un rôle symétrique donc:

$$\lambda_2 = \frac{\sum_{i=1}^{n} t_{i2}^{(p)} X_i}{\sum_{i=1}^{n} t_{i2}^{(p)}}$$

On trouve finalement :

$$\theta^{p+1} = \{\pi_1^{p+1} = \frac{\sum_{i=1}^{n} t_{i1}^{(p)}}{n}, \pi_2^{p+1} = \frac{\sum_{i=1}^{n} t_{i2}^{(p)}}{n}, \lambda_1^{p+1} = \frac{\sum_{i=1}^{n} t_{i1}^{(p)} X_i}{\sum_{i=1}^{n} t_{i1}^{(p)}}, \lambda_2^{p+1} = \frac{\sum_{i=1}^{n} t_{i2}^{(p)} X_i}{\sum_{i=1}^{n} t_{i2}^{(p)}}\}$$

```r
#retourne la liste des probabilities d'appartenance a un cluster
prob_opt <- function(posterior) {

  k = ncol(posterior)
  liste = c()
  for (i in 1:k ) {
    val = sum(posterior[,i])
    liste = append(liste, val)
  }
  return(liste / sum(liste))
}
```

```r
#retourne la list des lambdas
lamba_opt <- function(X, posterior){

  k = ncol(posterior)
  liste1 = c()
  liste2 = c()
  for (i in 1:k ) {
    val = posterior[,i]
    liste1 = append(liste1, sum(val*X))
    liste2 = append(liste2,sum(val))
  }

  return(liste1 / liste2)

}
```

On repète l'étape E et M jusqu'a ce que :

$$\frac{\|\theta^q - \theta^{q+1}\|^2}{\|\theta^q\|^2} < epsilon$$

avec epsilon petit. On code ci dessus la l'algorithme EM à l'aide des fonctions codées avant. Comme les lambdas sont des valeurs entières on arrondi les valeurs trouvées avec l'entier le plus proche.

```r
#retourne les lambdas and les probabilites optimales
# prend des valeurs aléatoire de lambdas and probs en argument
ExpectationMaximisation2 <- function(X, lambdas, probs,max_iterations,epsilon) {
  old_teta = c(lambdas,probs)

  posterior = Posterior(X,lambdas,probs)
  lambdas = lamba_opt(X,posterior)
  probs = prob_opt(posterior)

  teta = c(lambdas,probs)
  diff = sum((teta -old_teta)**2) / sum(old_teta**2)
  iter = 0
  #tant que la différence est plus grande que'espilon on continue de calculer
  while( diff > epsilon  && iter < max_iterations){
    posterior = Posterior(X,lambdas,probs) # Etape E : nouveau tik
    lambdas = lamba_opt(X,posterior) # Etape M : nouveaux lambdas
    probs = prob_opt(posterior) # Etape M : nouvelles probabilites

    old_teta = teta
    teta = c(lambdas,probs)
    diff = sum((teta -old_teta)**2) / sum(old_teta**2)
    print(iter)
    print(teta)
    iter = iter + 1 # nombre d' iterations
  }

  # recherche l'entier le plus proche
  for( i in 1:length(lambdas) ){
      lambdas[i] = ifelse((ceiling(lambdas[i])-lambdas[i]) > 0.5,floor(lambdas[i]),ceiling(lambdas[i]))
  }
```

```
  return(c(lambdas,probs))
}
```

```
ExpectationMaximisation2(data$obs, c(2,11),c(0.5,0.5),50,1e-10)
```

```
## [1] 0
## [1]  2.9054040 14.9750911  0.3210322  0.6789678
## [1] 1
## [1]  3.0003410 15.0840911  0.3296789  0.6703211
## [1] 2
## [1]  3.030757 15.114904  0.332218  0.667782
## [1] 3
## [1]  3.0402366 15.1241257  0.3329882  0.6670118
## [1] 4
## [1]  3.0431683 15.1269425  0.3332245  0.6667755
## [1] 5
## [1]  3.0440729 15.1278084  0.3332972  0.6667028
## [1] 6
## [1]  3.0443519 15.1280751  0.3333196  0.6666804
## [1] 7
## [1]  3.0444379 15.1281573  0.3333265  0.6666735
```

```
## [1]  3.0000000 15.0000000  0.3333265  0.6666735
```