TD noté sur l'algorithme EM

Teddy ALEXANDRE

Décembre 2022

Simulation

```
lambdas = c(3, 15)
pis = c(0.4, 0.6)
N = 1000
```

Question 1

On se sert de la fonction *rpois* pour générer les échantillons.

```
echpois1 <- rpois(n = 100, lambda = lambdas[1])
echpois1
```

```
##   [1]  2  1  3  0  3  1  2  1  4  4 11  2  2  3  0  1  5  3  7  6  1  4  3  3  1
##  [26]  2  3  3  4  1  1  0  3  5  2  2  3  4  2  7  5  2  2  3  5  2  3  1  2  7
##  [51]  3  3  3  2  4  4  4  2  3  1  1  4  2  4  2  3  4  7  0  2  9  2  3  4  3
##  [76]  3  6  2  3  5  4  4  3  3  5  1  1  2  2  3  2  5  4  4  3  4  2  1  6  1
```

Question 2

Idem :

```
echpois2 <- rpois(n = 200, lambda = lambdas[2])
echpois2
```

```
##   [1] 17 11 13 18 13 16 27 17 20 11 18 15 22 19 13 15 17 18 15 18 12 14 12 15 20
##  [26] 20 12 13 17 10 18 16 19 12 15 17 16 18 16 17 13 16 11 12 12 14 14 12 11 14
##  [51] 15 16 14 22 21 17 11 17 13 14 13 11 16 18 16 16 21 20 12 12 17 18 11 18 16
##  [76] 18 12  8  9 15 16 17 23 17 10 17 11 16 14 14  7 16 14 11  7 17 12 12 13 10
## [101] 15 18 10 16 20 17 18 12 16 13 21 14 15 14  6 20 20 12 15 12 19 13 13 21 16
## [126]  7 11 10 16 11 19 15 14  9 17 18 20 16 11 16 18 20 13 14 16 12 12 12 19 13
## [151] 17 12 13 13 14 19 12 17 11 22 15 13 11 17 20 11 17 14 16 18 18 16 18 10 13
## [176] 17 21 19 18  9 11 12 16 19  9 20 14 16 18 19 13 17 20 15 18 18 18 20 21 15
```

Question 3

On combine la fonction *rep* avec la création d'un vecteur avec *c()* :

```
int_vec <- c(rep(1, 100), rep(2, 200))
int_vec
```

```
##   [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 1 1
##  [38] 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 1
##  [75] 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 2 2 2 2 2 2 2 2 2 2
```

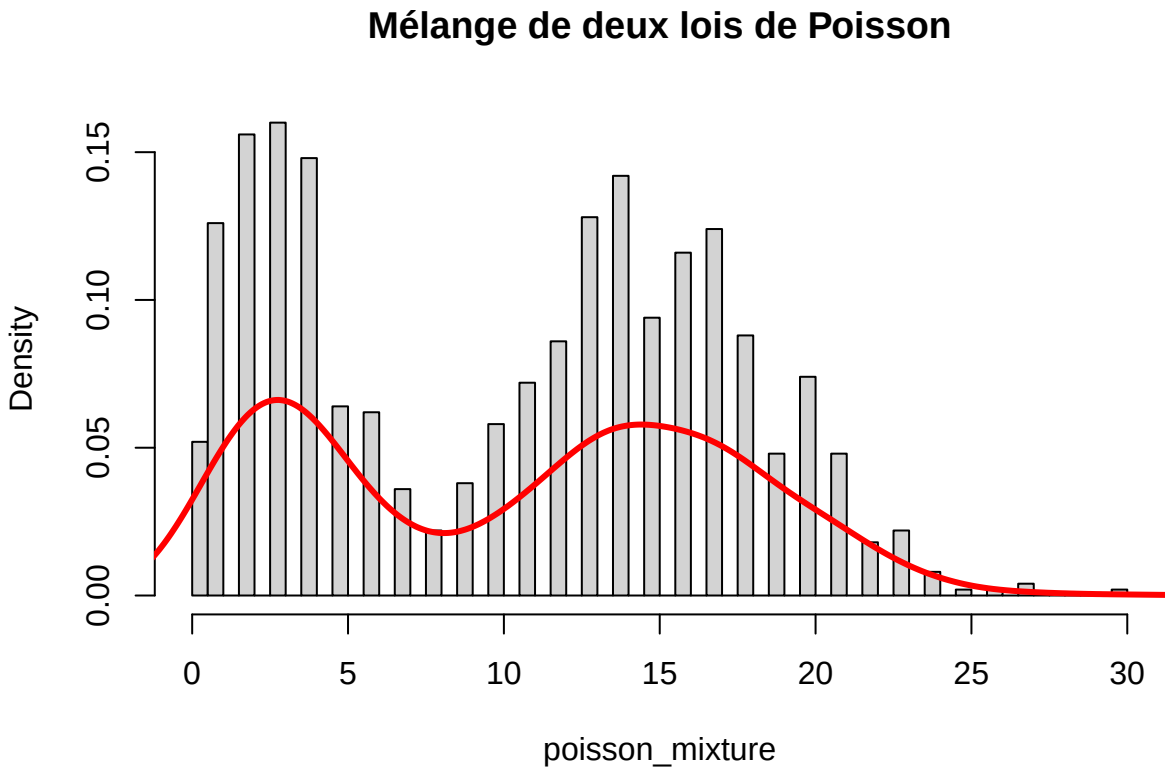
```
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [223] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [260] 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
## [297] 2 2 2 2
```

Question 4

On simule $N = 300$ valeurs dans $\{1, 2\}$ avec une probabilité $\pi_1 = 0.4$ d'obtenir la valeur 1, et, en fonction des valeurs obtenues, générer l'une des deux lois de Poisson issu du mélange à réaliser. On affiche ensuite le tout sous la forme d'un histogramme, avec la fonction de densité associée à l'échantillon :

```r
# Poisson mixture with 2 components
z = sample(1:2, size = N, replace = TRUE, prob = pis)
poisson_mixture = rep(NA, N)
for (i in 1:N) poisson_mixture[i] = rpois(1, lambdas[z[i]])

# Histogram + density
hist(poisson_mixture, breaks = 50, prob = TRUE, main = "Mélange de deux lois de Poisson")
lines(density(poisson_mixture), col = "red", lwd = 3)
```



Algorithme EM - Mélange de Poisson à K composantes

Les calculs et les notations utilisées ci-dessous sont détaillés sur une feuille jointe en annexe, dans le cas d'un mélange de Poisson à K composantes. Les questions 1 à 4 sont traités d'une seule traite, le tout dans une seule fonction. On définit quelques fonctions préliminaires :

```r
norme_deux_vec_carre <- function(u) {
    res = 0
```

```
    for (i in 1:length(u)) {
        res = res + (u[i])^2
    }
    res
}

critere_arret <- function(u, v) {
    res = norme_deux_vec_carre(u - v)/norme_deux_vec_carre(u)
    response = FALSE
    if (res <= 1e-09) {
        response = TRUE
    }
    response
}
```

L'algorithme EM implémenté (Questions 1 à 4):

```
EM_algorithm_poisson <- function(X, K) {
    # Q1 - Phase d'initialisation des paramètres
    N = length(X)
    probas_k = rep(0, K)
    lambda_k = rep(0, K)
    for (k in 1:K) lambda_k[k] = sample(1:15, size = 1)

    probas_k_old = probas_k
    lambda_k_old = lambda_k

    for (k in 1:K) {
        probas_k[k] = 1/K
        lambda_k[k] = sample(1:15, size = 1)
    }
    theta_old_k = c(probas_k_old, lambda_k_old)
    theta_k = c(probas_k, lambda_k)

    # Tant que l'on n'a pas atteint le critère d'arrêt
    while (!critere_arret(theta_old_k, theta_k)) {

        # Q2 - Etape 'E' : Calcul des t_ik
        T = matrix(rep(0, N * K), nrow = N, ncol = K)
        for (i in 1:N) {
            for (k in 1:K) {
                T[i, k] = probas_k[k] * dpois(X[i], lambda_k[k])/sum(probas_k *
                  dpois(X[i], lambda_k))
            }
        }

        # Q3 - Etape 'M' : Calcul des nouveaux probas_k et
        # lambda_k
        for (k in 1:K) {
            sum_i_tik = 0
            sum_i_tik_xi = 0
            sum_i_tiK_xi = 0
            for (i in 1:N) {
```

```
                sum_i_tik = sum_i_tik + T[i, k]
                sum_i_tik_xi = sum_i_tik_xi + T[i, k] * X[i]
                sum_i_tiK_xi = sum_i_tiK_xi + T[i, K] * X[i]
            }
            probas_k[k] = sum_i_tik/N
            lambda_k[k] = sum_i_tik_xi/sum_i_tik
        }

        probas_k[K] = 1 - sum(probas_k[1:K - 1])
        theta_old_k = theta_k
        theta_k = c(probas_k, lambda_k)
    }
    theta_k
}

# Q4 - Test avec les valeurs simulées plus haut
for (i in 1:5) print(EM_algorithm_poisson(poisson_mixture, 2))
```

```
## [1]  0.6062476  0.3937524 15.1634490  3.0023832
## [1]  0.606248  0.393752 15.163443  3.002379
## [1]  0.3937508  0.6062492  3.0023664 15.1634275
## [1]  0.6062367  0.3937633 15.1635935  3.0024965
## [1]  0.3937626  0.6062374  3.0024897 15.1635849
```

Conclusion

L'algorithme EM implémenté est fonctionnel, on obtient une assez bonne estimation des hyperparamètres modélisés ($\pi_1 = 0.4, \pi_2 = 0.6, \lambda_1 = 3, \lambda_2 = 15$ environ). On effectue plusieurs répétitions pour s'en convaincre. On retrouve donc à peu près les valeurs fixées dans la partie "Simulation". On sent néanmoins que les valeurs obtenues par estimation peuvent encore se rapprocher des vraies valeurs. Les valeurs obtenues convergent lorsque N augmente (car échantillon plus grand). Les valeurs ci-dessous sont obtenues pour $N = 3000$.