Digression : Le package Mclust

Carvaillo, Côme, Pralon

Nous avons, dans un élan d'audace, commencé par programmer à la main l'algorithme EM, en nous appuyant sur le pseudo-code explicité en première partie du chapitre II.

Cependant, il existe une librairie *R* - la librairie *mclust* - contenant une implémentation de l'algorithme EM. Notre algorithme étant fonctionnel, nous ne détaillerons pas ici le fonctionnement de ce Package. Il est néanmoins pertinent de l'expérimenter, voire de comparer ces résultats avec ceux notre algorithme.

Pour commencer, installons et chargeons le Package *mclust*.

```
#install.packages("mclust")
library("mclust")
```

```
## Package 'mclust' version 5.4.9
## Type 'citation("mclust")' for citing this R package in publications.
```

Nous reprenons les données des nids d'oiseaux :

```
bird_names = c("European Goldfinch", "Common Linnet", "Common Chaffinch",
               "European Greenfinch", "Eurasian Bullfinch", "Hawfinch",
               "Stonechat", "European Robin", "Whinchat", "Song Thrush",
               "Common Blackbird", "Ring Ouzel", "Mistle Thrush")

mean_volume = c(38.0, 60.9, 58.3, 74.5, 45.0, 71.6, 91.0, 68.4, 51.9, 288.9,
                293.6, 298.6, 266.1)

sd_volume = c(9.1,  20.8, 15.0,  12.2,  3.8,  12.9,  46.5, 29.8, 27.4, 55.9,
              78.5,  125.1,  56.6)
```

Puis, il suffit de construire des dataframe. Ici, nous considérerons un mélange à deux lois et un autre à trois lois.

```
df_2= data.frame(bird_names = c("European Goldfinch", "Ring Ouzel")
                           ,proportion_alpha = c(0.3, 0.7), mean = c(38, 298.6),
                           sd = c(9.1, 125.1))

df_3= data.frame(bird_names = c("Common Linnet", "Common Chaffinch", "Hawfinch" )
                           ,proportion_alpha = c(0.6, 0.3, 0.1), mean = c(60.9, 58.3, 71.6),
                           sd = c(20.8, 15.0, 12.9))
```

On reprend la fonction de simulation :

```
simulation = function(data_th, n=100){
  X = rep(NA,n) #échantillon
  J = dim(data_th)[1] #nb de mélange
  vect_alpha = data_th[,2]
  vect_mean = data_th[,3]
  vect_sd = data_th[,4]
  for(i in 1:n){
```

```
    Z = runif(1)
    if (Z <= vect_alpha[1]){
      X[i] = rnorm(1, vect_mean[1], vect_sd[1])
    }else{
      k = 1
      l = 2
      Bool = FALSE
      cumul_alpha = cumsum(vect_alpha)
      while(Bool == FALSE){
        if((cumul_alpha[k]<=Z) & (cumul_alpha[l]>=Z)){
          Bool = TRUE
          param_index = l
        }
        k = k+1
        l = l+1
      }
      X[i] = rnorm(1, vect_mean[param_index], vect_sd[param_index])
    }
  }
  return(X)
}
```

Les prérequis étant posés, on simule un échantillon X2 de deux espèces d'oiseaux et un autre X3 de trois espèces d'oiseaux :

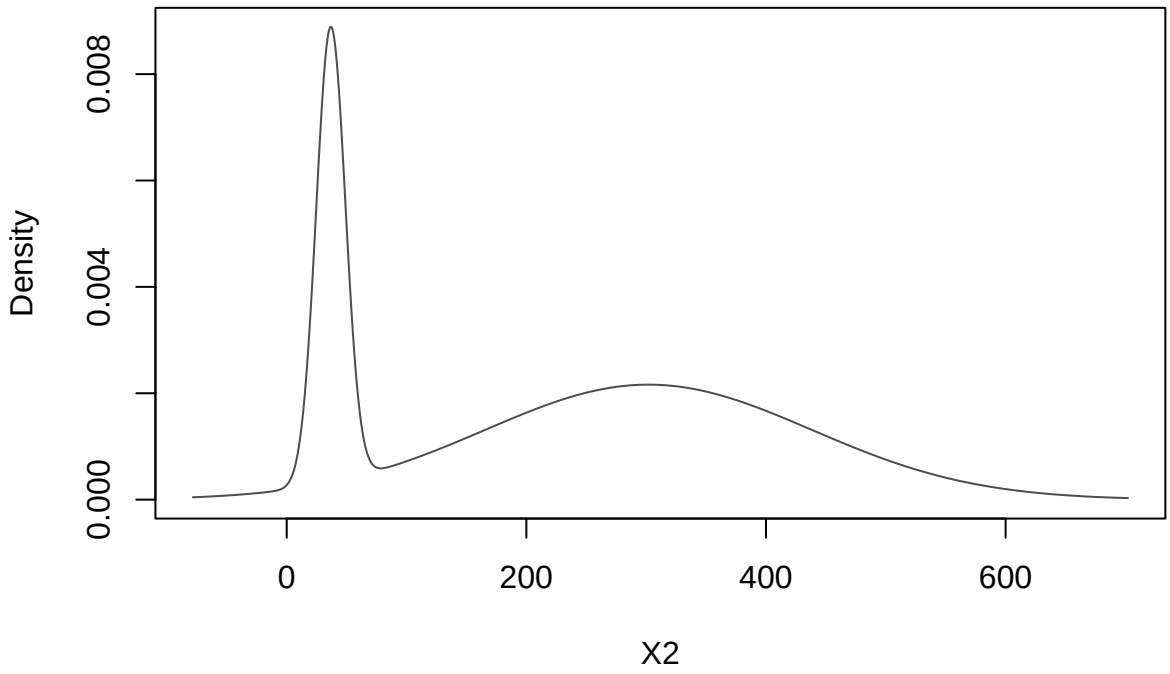
```
set.seed(1907)
X2 <- simulation(df_2)
X3 <- simulation(df_3)
```

Le Package *mclust* est des plus complet; les possibilités étant très vastes et hors du cadre de ce projet (notamment les fonctionnalités de clustering), nous regarderons uniquement la fonction qui nous intéresse, à savoir la fonction *densityMclust*.

Cette dernière prend en argument des fonctionnalités pertinentes, comme le nombre de mélanges, mais non des valeurs initiales pour les paramètres.

Commençons par le cas d'un mélange de deux lois:

```
est_2 <- densityMclust(X2)
```



Nous obtenons ainsi une première sortie nous présentant la densité du mélange de lois.

```
est_2
```

```
## 'densityMclust' model object: (V,2)
##
## Available components:
##  [1] "call"           "data"           "modelName"      "n"
##  [5] "d"              "G"              "BIC"            "loglik"
##  [9] "df"             "bic"            "icl"            "hypvol"
## [13] "parameters"     "z"              "classification" "uncertainty"
## [17] "density"
```

Ici nous voulons les paramètres estimés, nous nous concentrerons donc que sur la treizième coordonnée de ce vecteur.

Rappelons que les divers paramètres de ce mélange sont : 0.3 et 0.7 en proportions; 38 et 298.6 pour les moyennes; et 9.1 et 125.1 pour les variances.

```
est_2[13]
```

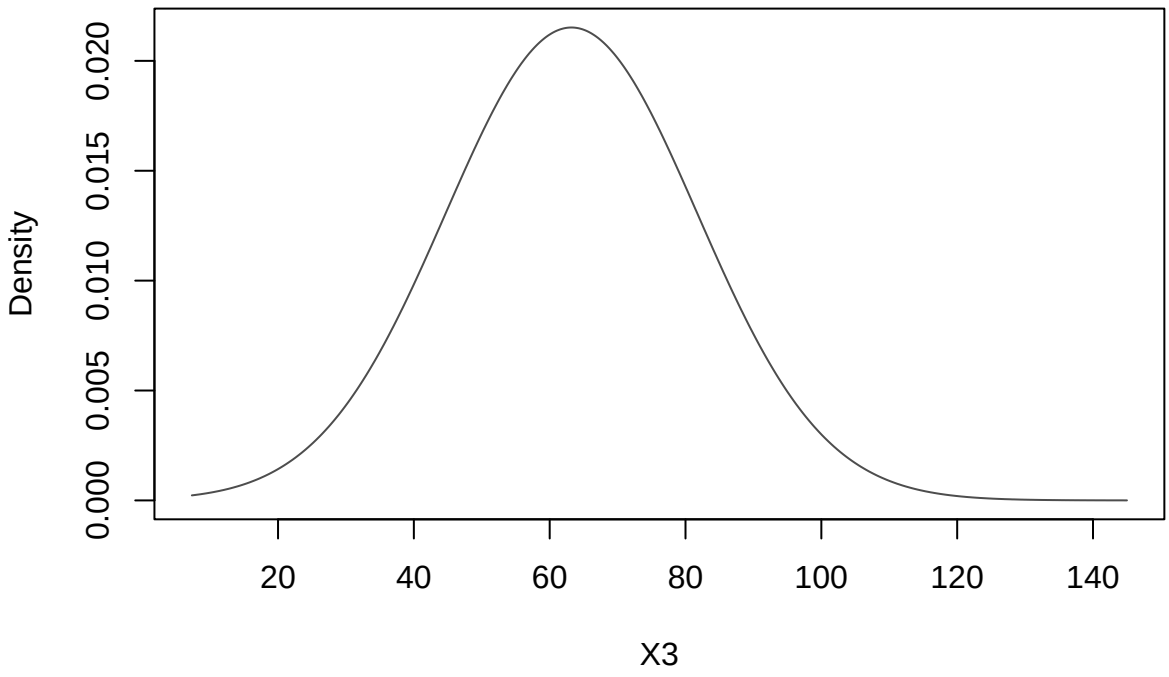
```
## $parameters
## $parameters$pro
## [1] 0.2612243 0.7387757
##
## $parameters$mean
```

```
##         1         2
##  36.80898 301.97108
##
## $parameters$variance
## $parameters$variance$modelName
## [1] "V"
##
## $parameters$variance$d
## [1] 1
##
## $parameters$variance$G
## [1] 2
##
## $parameters$variance$sigmasq
## [1]   147.928 18583.841
##
## $parameters$variance$scale
## [1]   147.928 18583.841
```

Ici, le nombre de mélange est exact. Les proportions sont très bien estimées, l'erreur la plus importante est de l'ordre de 12%. Il en va de même pour les moyennes, les erreurs sont d'ordres inférieures à 10%. Les variances sont elles aussi très bien estimées, les erreurs sont négligeables.

Regardons maintenant le cas d'un mélange de trois lois. Afin de mettre à rude épreuve l'algorithme, nous allons choisir les espèces telles que les moyennes et variances soient proches. Les proportions seront quant à elles bien distinctes, nous allons voir pourquoi.

```
est_3<- densityMclust(X3)
```




```
est_3[13]
```

```
## $parameters
## $parameters$pro
## [1] 1
##
## $parameters$mean
## [1] 63.20547
##
## $parameters$variance
## $parameters$variance$modelName
## [1] "X"
##
## $parameters$variance$d
## [1] 1
##
## $parameters$variance$G
## [1] 1
##
## $parameters$variance$sigmasq
## [1] 343.7438
```

Nous obtenons ici quelque chose de fascinant; la fonction de densité de ce mélange de trois lois paraît toute à fait gaussienne. Sans une exploration plus approfondie des données, nous commettrions une chagrinante erreur et des conclusions totalement faussées...

Effectuons un test de Shapiro sur le jeu de données, afin de tester sa normalité.

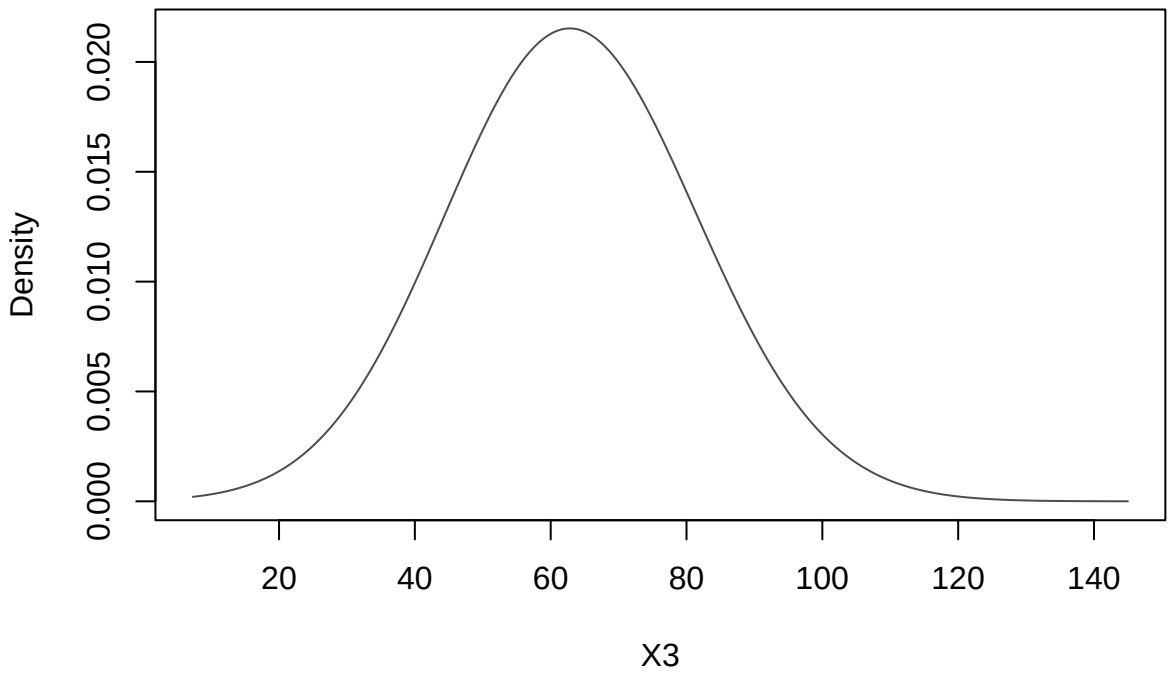
```
shapiro.test(X3)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  X3
## W = 0.97982, p-value = 0.1287
```

La p-value est de 0.1287, ce qui est certes peu élevée, mais pas assez pour rejeter l'hypothèse (H_0) de normalité. Nous sommes ici dans une situation ambïgue.

Aussi, *mclust* échoue à établir le nombre correct de lois. Nous allons donc le préciser.

```
est_3b <- densityMclust(X3, G = 3)
```




```
est_3b[13]
```

```
## $parameters
## $parameters$pro
## [1] 0.2552686 0.5642272 0.1805042
##
## $parameters$mean
##        1        2        3
```

```
## 56.63179 62.36268 75.13635
##
## $parameters$variance
## $parameters$variance$modelName
## [1] "E"
##
## $parameters$variance$d
## [1] 1
##
## $parameters$variance$G
## [1] 3
##
## $parameters$variance$sigmasq
## [1] 306.618
```

Les proportions sont plutôt bien estimées, les erreurs sont faibles. Il en est étonnament de même pour les moyennes. Ceci est surprenant au vue de l'allure de la densité. Cependant, il n'est estimée qu'une unique variance, ce qui n'est guère étonnant. Notons que celle ci est à peu prêt égale à la moyenne des variances des différentes lois.

Bibliographie :

https://cran.r-project.org/web/packages/mclust/mclust.pdf

https://rstudio-pubs-static.s3.amazonaws.com/154174_78c021bc71ab42f8add0b2966938a3b8.html