



Projet Master 1 SSD

Un modèle pour les nids d'oiseaux

Rédigé par

CARVAILLO Thomas

CÔME Olivier

PRALON Nicolas

Encadrante : Elodie BRUNEL-PICCININI

IMAG
INSTITUT MONTPELLIERAIN
ALEXANDER GROTHENDIECK



Table des matières

Introduction	2
1 Modélisation du problème et liminaire mathématique	3
1.1 Modélisation du problème	3
1.2 Une histoire de densités	4
1.3 Une approche idéaliste	5
1.4 Une situation concordante à la réalité	7
2 L'algorithme EM	12
2.1 Présentation laconique et pseudo-code	12
2.1.1 L'étape E (Expectation)	12
2.1.2 L'étape M (Maximization)	12
2.1.3 Pseudo-code de l'algorithme EM	13
2.1.4 Implémentation de l'algorithme EM	13
2.2 Une preuve de croissance	15
3 Etude de simulations	17
3.1 La fonction simulation	17
3.2 Étude de simulations	18
3.2.1 Étude d'un mélange gaussien à "forte séparation"	18
3.2.2 Étude d'un mélange gaussien à "faible séparation"	21
3.3 Automatisation du choix des paramètres initiaux	22
3.3.1 Fonction param_quantile1	23
3.3.2 Fonction param_quantile2	23
3.3.3 Fonction param_kmeans	24
4 Un modèle pour les nids d'oiseaux	26
4.1 Préambule	26
4.2 Etude d'un mélange à deux lois	28
4.2.1 Première réalisation de l'étude	28
4.2.2 Seconde réalisation de l'étude	30
Conclusion	32
Bibliographie	33
Annexes	34
A Le package <i>mclust</i>	35
A.1 Un exemple sur un mélange à deux lois	36
A.2 Un exemple sur un mélange à trois lois	38
B Etudes des fonctions	40

Introduction

L'observation est une composante essentielle en sciences appliquées, et tout particulièrement pour nous, futurs statisticiens. Cette dernière nous permet d'élaborer des modèles, et dans le cadre de l'estimation paramétrique, de construire des estimateurs.

Dans la situation où les données sont correctement observées, cadre idéal, la statistique inférentielle nous permet d'obtenir dans certaines situations d'agréables expressions analytiques des estimateurs, notamment par la méthode du maximum de vraisemblance, qui est d'une redoutable efficacité. Toujours est-il que ces situations sont peu courantes ; nous sommes souvent confrontés à des situations présentant des données « cachées » ou manquantes. Dans ce dernier cas, nous verrons que les méthodes analytiques exactes ne suffisent plus, et qu'il est nécessaire d'introduire de nouvelles méthodes, également basées sur le principe du maximum de vraisemblance, pour tenter d'estimer les paramètres.

Nous étudierons dans le présent projet l'une de ces méthodes, conçue par Dempster et al. [1], l'algorithme Expectation-Maximization.

Nous nous appuierons sur un exemple extrait de l'ornithologie ; les données observées seront celles des volumes des nids d'oiseaux, et les données manquantes seront l'espèce qui l'a construit.

Ceci constituera le fil rouge de ce projet, et sera le cœur de notre problématique ; comment estimer les divers paramètres associés aux observations quand une partie des données est inaccessible ?

Dans un premier chapitre, nous modéliserons mathématiquement le problème et poserons les bases théoriques nécessaires à l'étude ; nous verrons brièvement le cas des données complètes, puis étudierons le cas des observations manquantes. Un second chapitre sera consacré à une description et une implémentation en langage *R* de l'algorithme EM. Puis, dans un troisième chapitre, nous étudierons les performances des résultats de notre implémentation à l'aide de simulations ; nous détaillerons notre implémentation, et étudierons son risque d'erreur. Dans un quatrième et dernier chapitre, nous considérerons une suite d'observations, et proposerons un modèle pour les nids d'oiseaux à l'aide des outils que nous aurons implémentés.

Chapitre 1

Modélisation du problème et liminaire mathématique

Dans ce premier chapitre, nous introduirons les outils nécessaires pour modéliser mathématiquement le problème. Puis, nous présenterons successivement les deux situations existantes dans le cadre du recueil de données des nids d'oiseaux : celle dans laquelle et l'espèce et le volume du nid ont été observés ; et celle dans laquelle seul le volume a pu être observé. Nous verrons ce qui diffère entre ces deux situations et les raisons qui ont incité à la création de l'algorithme EM, l'objet de notre projet.

Ce chapitre s'inspirera des références [2], [3] et [4].

1.1 Modélisation du problème

Nous allons pour commencer donner une première définition, qui est au coeur du présent projet.

Définition 1 (Loi de mélange). On appelle loi de mélange toute loi dont la densité s'écrit sous la forme d'une combinaison convexe de plusieurs densités. Si l'on se donne J densités $f_1(x), \dots, f_J(x)$, alors toute variable aléatoire X dont la densité f s'exprime, pour tout $x \in \mathbb{R}$, sous la forme

$$f(x) := \sum_{j=1}^J \alpha_j f_j(x) \text{ , } \alpha_j \in \mathbb{R}_+^* \text{ et } \sum_{j=1}^J \alpha_j = 1$$

suit une loi de mélange continue.

Afin de modéliser commodément le problème, nous introduisons les variables aléatoires suivantes :

- ✂ La variable aléatoire X , modélisant le volume des nids, de densité f ,
- ✂ La variable aléatoire discrète Z , à valeurs dans $\llbracket 1, J \rrbracket$, représentant l'espèce d'oiseau qui a construit le nid.

Enfin, nous nous placerons sous les hypothèses suivantes :

Hypothèse 1. Nous supposons que, $\forall j \in \llbracket 1, J \rrbracket$, le volume des nids d'une espèce j (i.e. X conditionnellement à $(Z = j)$) suit une loi normale $\mathcal{N}(\mu_j, v_j)$. Nous dénoterons par $f(x|Z = j) := \gamma_{\mu_j, v_j}(x)$ cette densité.

Hypothèse 2. Soit $\Theta := \{\theta = (\alpha_j, \mu_j, v_j)_{1 \leq j \leq J} \text{ tels que } \alpha_j > 0 \ \forall j \in \llbracket 1, J \rrbracket \text{ et } \sum_{j=1}^J \alpha_j = 1\}$. Soit X_1, \dots, X_n un échantillon de même loi que X . On supposera qu'il existe un $\theta \in \Theta$ tel que les données récoltées, ici les volumes des nids, soient la réalisation du précédent échantillon.

Remarque 1. La variable Z est discrète et à valeurs dans un sous-ensemble fini de \mathbb{N} , elle suit donc une loi

$$\sum_{j=1}^J \alpha_j \delta_j$$

où J représente le nombre d'espèces d'oiseaux considéré et les α_j sont des réels, strictement positifs, représentant la proportion de nids de l'espèce j , tels que $\sum_{j=1}^J \alpha_j = 1$.

Il s'ensuit la proposition suivante, qui sera essentielle dans la suite.

Proposition 1. *La distribution du volume des nids des oiseaux X admet pour densité, au point x et par rapport à la mesure de Lebesgue sur \mathbb{R} , la fonction f_θ définie comme suit*

$$f_\theta(x) = \sum_{j=1}^J \alpha_j \gamma_{\mu_j, v_j}(x)$$

Démonstration. En effet, la fonction de répartition de X s'écrit

$$\begin{aligned} \mathbb{P}(X \leq x) &= \mathbb{P}\left(\bigcup_{j=1}^J (X \leq x) \cap (Z = j)\right) \\ &= \sum_{j=1}^J \mathbb{P}((X \leq x) \cap (Z = j)) \\ &= \sum_{j=1}^J \mathbb{P}(Z = j) \times \mathbb{P}(x \leq X | Z = j) \\ &= \sum_{j=1}^J \alpha_j \times \mathbb{P}(x \leq X | Z = j) \end{aligned}$$

On en déduit donc que la densité de X vaut :

$$\sum_{j=1}^J \alpha_j \times f(x|Z = j) = \sum_{j=1}^J \alpha_j \times \gamma_{\mu_j, v_j}(x)$$

2

Le but de ce projet sera d'étudier des méthodes permettant l'estimation des divers paramètres de cette densité.

1.2 Une histoire de densités

Les variables X et Z sont toutes deux intégrables, nous pouvons dès lors considérer la loi conditionnelle de Z sachant X , et la loi du couple (X, Z) . Nous introduisons donc une dernière densité et une dernière probabilité, qui nous seront fort utiles dans la suite.

Proposition 2. *Nous avons les résultats suivants :*

1. *La loi du couple (X, Z) est donnée par*

$$\begin{aligned} \mathbb{P}(X \leq x, Z = j) &= \int_{-\infty}^x f_\theta(u|Z = j) \times \mathbb{P}(Z = j) du \\ &= \int_{-\infty}^x \underbrace{\gamma_{\mu_j, v_j}(u) \times \alpha_j}_{:= h_\theta(u, j)} du \end{aligned}$$

où $h_\theta(u, j)$ est la "sous-densité" de $X \times \mathbb{1}_{(Z=j)}$

2. La probabilité de la loi de Z sachant $X = x$ est donnée par :

$$\mathbb{P}_\theta(Z = j|X = x) = \frac{\gamma_{\mu_j, v_j} \times \alpha_j}{f_\theta(x)}$$

où $f_\theta(x)$ est donnée par la proposition 1.

Démonstration. En effet,

$$\begin{aligned} \mathbb{P}(X \leq x, Z = j) &= \mathbb{P}(X \leq x|Z = j) \times \mathbb{P}(Z = j) \\ &= \int_{-\infty}^x \mathbb{P}(Z = j|X = u) \times f_\theta(u) du \end{aligned}$$

On obtient dès lors

$$\mathbb{P}(Z = j|X = u) \times f_\theta(u) = \gamma_{\mu_j, v_j}(u) \times \alpha_j$$

Soit

$$\mathbb{P}(Z = j|X = u) = \frac{\gamma_{\mu_j, v_j}(u) \times \alpha_j}{f_\theta(u)} := \frac{h_\theta(u, j)}{f_\theta(u)}$$

2

Remarque 2. Nous pouvons dès à présent noter que pour un échantillon X_1, \dots, X_n de même loi que X , nous avons

$$\forall i \in \llbracket 1, n \rrbracket, h_\theta(X_i, j) = f_\theta(X_i) \times \mathbb{P}_\theta(Z = j|X = X_i)$$

Ceci nous sera utile dans la suite.

Nous allons dès à présent nous intéresser à l'estimation de ces paramètres.

1.3 Une approche idéaliste

Regardons dans un premier temps un cas simplifié, un cas ne décrivant pas la réalité des observations mais qui constitue une agréable entrée en matière.

Nous supposons ici qu'ont été relevés simultanément et les mesures des tailles des nids et l'espèce d'oiseau qui l'a construit. Les observations considérées ici sont donc composées des couples (X_i, Z_i) , $i \in \llbracket 1, n \rrbracket$. On considérera dès lors la fonction de densité $h_\theta(x, z)$, donnée par la proposition 2.

L'estimation des divers paramètres est alors élémentaire, en témoigne les propositions suivantes :

Proposition 3 (Fonction de Log-vraisemblance). *La Log-vraisemblance du modèle s'écrit*

$$\mathcal{L}_\theta(X_1, \dots, X_n, Z_1, \dots, Z_n) = \sum_{j=1}^J \#A_j \ln(\alpha_j) + \sum_{j=1}^J \sum_{i \in A_j} \ln(\gamma_{\mu_j, v_j}(X_i))$$

où les A_j sont définis par $A_j := \{i \in \llbracket 1, n \rrbracket \text{ tels que } Z_i = j\}$ i.e. $\bigcup_{j=1}^J A_j = \llbracket 1, n \rrbracket$

Démonstration. La Log-vraisemblance du modèle s'écrit :

$$\begin{aligned} \mathcal{L}_\theta(X_1, \dots, X_n, Z_1, \dots, Z_n) &= \ln \left(\prod_{i=1}^n h_\theta(X_i, Z_i) \right) \\ &= \ln \left(\prod_{i=1}^n \alpha_{Z_i} \gamma_{\mu_{Z_i}, v_{Z_i}}(X_i) \right) \\ &= \sum_{i=1}^n \ln(\alpha_{Z_i}) + \ln(\gamma_{\mu_{Z_i}, v_{Z_i}}(X_i)) \end{aligned}$$

Z_i est à valeur dans $\llbracket 1, J \rrbracket$, on partitionne donc $I := \llbracket 1, n \rrbracket$ comme $I = \bigcup_{j=1}^J A_j$ pour obtenir

$$\begin{aligned}\mathcal{L}_\theta(X_1, \dots, X_n, Z_1, \dots, Z_n) &= \sum_{j=1}^J \sum_{i \in A_j} \ln(\alpha_j) + \sum_{j=1}^J \sum_{i \in A_j} \ln(\gamma_{\mu_j, v_j}(X_i)) \\ &= \sum_{j=1}^J \sum_{i \in A_j} \ln(\alpha_j) + \sum_{j=1}^J \sum_{i \in A_j} \ln(\gamma_{\mu_j, v_j}(X_i)) \\ &= \sum_{j=1}^J \#A_j \ln(\alpha_j) + \sum_{j=1}^J \sum_{i \in A_j} \ln(\gamma_{\mu_j, v_j}(X_i))\end{aligned}$$

2

Nous pouvons dès lors maximiser la log-vraisemblance afin d'obtenir les estimateurs souhaités :

Proposition 4 (Estimateurs). *Les estimateurs du maximum de vraisemblance $\widehat{\alpha}_j$ (resp. $\widehat{\mu}_j$, et \widehat{v}_j) de α_j (resp. μ_j et v_j) sont donnés par*

$$\begin{aligned}\widehat{\alpha}_j &= \frac{\#A_j}{n} \\ \widehat{\mu}_j &= \frac{\sum_{i \in A_j} X_i}{\#A_j} \\ \widehat{v}_j &= \frac{\sum_{i \in A_j} (X_i - \widehat{\mu}_j)^2}{\#A_j}\end{aligned}$$

Démonstration. Soit $\theta = (\alpha_j, \mu_j, v_j)_{j \in \llbracket 1, J \rrbracket}$. Il s'agit de déterminer

$$\operatorname{argmax}_{\theta \in \mathbb{R}^{3J}, \sum_{j=1}^J \alpha_j = 1} \left(\sum_{j=1}^J \#A_j \ln(\alpha_j) + \sum_{j=1}^J \sum_{i \in A_j} \ln(\gamma_{\mu_j, v_j}(X_i)) \right)$$

Nous avons donc à résoudre un programme de minimisation d'une fonction convexe sur un convexe avec une contrainte égalité, il est ainsi naturel de faire appel au Lagrangien.

Ce dernier s'écrit

$$\begin{aligned}L(\theta) &= \sum_{j=1}^J \#A_j \ln(\alpha_j) + \sum_{j=1}^J \sum_{i \in A_j} \ln(\gamma_{\mu_j, v_j}(X_i)) - \lambda \times \left(\sum_{j=1}^J \alpha_j - 1 \right) \\ &= \sum_{j=1}^J \#A_j \ln(\alpha_j) + \sum_{j=1}^J \sum_{i \in A_j} \ln \left(\frac{1}{\sqrt{2\pi v_j}} \exp \left(-\frac{(X_i - \mu_j)^2}{2v_j} \right) \right) - \lambda \times \left(\sum_{j=1}^J \alpha_j - 1 \right) \\ &= \sum_{j=1}^J \#A_j \ln(\alpha_j) + \sum_{j=1}^J \sum_{i \in A_j} \left(\frac{-1}{2} \ln(2\pi v_j) - \frac{(X_i - \mu_j)^2}{2v_j} \right) - \lambda \times \left(\sum_{j=1}^J \alpha_j - 1 \right)\end{aligned}$$

Il reste maintenant à résoudre le système suivant, afin d'obtenir le vecteur $\widehat{\theta} := (\widehat{\alpha}_j, \widehat{\mu}_j, \widehat{v}_j)_{j \in \llbracket 1, J \rrbracket}$ solution du programme.

$$\begin{cases} \frac{\#A_j}{\widehat{\alpha}_j} - \lambda &= 0 \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{i \in A_j} (X_i - \widehat{\mu}_j) / \widehat{v}_j &= 0 \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{i \in A_j} \frac{-0.5 \times 2 \times \pi}{2\pi \widehat{v}_j} + \frac{(X_i - \widehat{\mu}_j)^2}{2\widehat{v}_j^2} &= 0 \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{j=1}^J \widehat{\alpha}_j &= 1 \end{cases}$$

Ceci équivaut à

$$\begin{cases} \frac{\#A_j}{\widehat{\alpha}_j} &= \lambda \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{i \in A_j} X_i &= \sum_{i \in A_j} \widehat{\mu}_j \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{i \in A_j} (X_i - \widehat{\mu}_j)^2 &= \sum_{i \in A_j} \widehat{v}_j \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{j=1}^J \widehat{\alpha}_j &= 1 \end{cases} \Leftrightarrow \begin{cases} \frac{\#A_j}{\widehat{\alpha}_j} &= \lambda \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{i \in A_j} \frac{X_i}{\#A_j} &= \widehat{\mu}_j \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{i \in A_j} \frac{(X_i - \widehat{\mu}_j)^2}{\#A_j} &= \widehat{v}_j \quad \forall j \in \llbracket 1, J \rrbracket \\ \sum_{j=1}^J \widehat{\alpha}_j &= 1 \end{cases}$$

En sommant les J premières lignes du système, on obtient $\sum_{j=1}^J \#A_j = \sum_{j=1}^J \widehat{\alpha}_j \lambda$, i.e. $\lambda = n$. En injectant ceci dans le précédent système, on obtient finalement ce qui était annoncé :

$$\begin{cases} \widehat{\alpha}_j &= \frac{\#A_j}{n} \quad \forall j \in \llbracket 1, J \rrbracket \\ \widehat{\mu}_j &= \sum_{i \in A_j} \frac{X_i}{\#A_j} \quad \forall j \in \llbracket 1, J \rrbracket \\ \widehat{v}_j &= \sum_{i \in A_j} \frac{(X_i - \widehat{\mu}_j)^2}{\#A_j} \quad \forall j \in \llbracket 1, J \rrbracket \end{cases}$$

1.4 Une situation concordante à la réalité

Nous nous placerons désormais dans un contexte tout autre que celui du paragraphe précédent, un contexte concordant davantage à la réalité. Dans ce qui suit, nous supposons que ne sont observés que les volumes des nids, les diverses espèces d'oiseaux les ayant construit étant en quelque sorte des données inobservées ou "cachées". Nous avons donc un échantillon X_1, \dots, X_n de même loi que la variable X comme définie ci-dessus. La log-vraisemblance des observations \mathcal{L}_{obs} s'obtient aisément :

$$\mathcal{L}_{obs}(\theta, X_1, \dots, X_n) := \ln \left(\prod_{i=1}^n f_{\theta}(X_i) \right) = \sum_{i=1}^n \ln \left(\sum_{j=1}^J \alpha_j \gamma_{\mu_j, v_j}(X_i) \right)$$

Nous voyons dès lors que l'existence d'une expression analytique du maximum de la log-vraisemblance n'est pas assurée. Il est donc nécessaire de trouver un moyen d'approcher les valeurs des différents estimateurs.

Un cheminement de pensées naturel est de considérer l'espérance conditionnelle du modèle d'échantillonnage $(X_1, Z_1), \dots, (X_n, Z_n)$ sachant l'échantillon X_1, \dots, X_n observé. En effet, cette dernière constitue la meilleure approximation de la log-vraisemblance du modèle par rapport à ce qui est observable.

Définition 2 (Log-vraisemblance conditionnelle). On définit la log-vraisemblance conditionnelle $\mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n)$ par

$$\mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n) := \mathbb{E}_{\tilde{\theta}}[\mathcal{L}_{\theta}(X_1, \dots, X_n, Z_1, \dots, Z_n) | X_1, \dots, X_n]$$

Nous allons maintenant travailler sur l'expression de la log-vraisemblance conditionnelle et en donner une expression simplifiée, qui nous sera fort utile ultérieurement, et une expression plus substantielle, qui nous sera immédiatement utile.

Proposition 5. *Nous avons*

$$\mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n) = \sum_{i=1}^n \sum_{j=1}^J \ln(h_\theta(X_i, j)) \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)$$

Avant de démontrer cette proposition, nous allons introduire une notation qui nous sera immédiatement utile :

Notation 1. Dans ce qui suit, nous noterons

$$\delta_{i,j} := \mathbb{1}_{(Z_i=j)} = \begin{cases} 1 & \text{si } Z_i = j \\ 0 & \text{sinon} \end{cases}$$

Ainsi,

$$h_\theta(X_i, Z_i) = \prod_{j=1}^J h_\theta(X_i, j)^{\delta_{i,j}}$$

Démonstration. En effet

$$\begin{aligned} \mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n) &= \mathbb{E}_{\tilde{\theta}}[\mathcal{L}_\theta(X_1, \dots, X_n, Z_1, \dots, Z_n) | X_1, \dots, X_n] \\ &= \mathbb{E}_{\tilde{\theta}} \left[\ln \left(\prod_{i=1}^n \prod_{j=1}^J h_\theta(X_i, j)^{\delta_{i,j}} \right) \middle| X_1, \dots, X_n \right] \\ &= \sum_{i=1}^n \sum_{j=1}^J \mathbb{E}_{\tilde{\theta}}[\delta_{i,j} \times \ln(h_\theta(X_i, j)) | X_1, \dots, X_n] \\ &= \sum_{i=1}^n \sum_{j=1}^J \mathbb{E}_{\tilde{\theta}}[\mathbb{1}_{(Z_i=j)} \times \ln(h_\theta(X_i, j)) | X_1, \dots, X_n] \end{aligned}$$

Or, les couples (X_i, Z_i) sont indépendants, donc

$$\begin{aligned} \mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n) &= \sum_{i=1}^n \sum_{j=1}^J \mathbb{E}_{\tilde{\theta}}[\mathbb{1}_{(Z_i=j)} \times \ln(h_\theta(X_i, j)) | X_i] \\ &= \sum_{i=1}^n \sum_{j=1}^J \ln(h_\theta(X_i, j)) \mathbb{E}_{\tilde{\theta}}[\mathbb{1}_{(Z_i=j)} | X_i] \\ &= \sum_{i=1}^n \sum_{j=1}^J \ln(h_\theta(X_i, j)) \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \end{aligned}$$

☞

Nous nous appuyerons sur l'expression suivante pour l'expression des estimateurs du maximum de vraisemblance :

Proposition 6. *La fonction $\mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n)$ se réécrit sous la forme suivante :*

$$\begin{aligned} \mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n) &= -\frac{n}{2} \ln(2\pi) + \sum_{j=1}^J \left(\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \right) \times \ln(\alpha_j) \\ &\quad - \frac{1}{2} \sum_{j=1}^J \left(\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \times \left(\log(v_j) + \frac{(X_i - \mu_j)^2}{v_j} \right) \right) \end{aligned}$$

Démonstration. Nous commençons par considérer la forme précédente de la log-vraisemblance conditionnelle ; nous avons ainsi, d'après la proposition 5 :

$$\begin{aligned}
\mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n) &= \sum_{i=1}^n \sum_{j=1}^J \ln(h_{\theta}(X_i, j)) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \\
&= \sum_{i=1}^n \sum_{j=1}^J \ln(\alpha_j \gamma_{\mu_j, v_j}(X_i)) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \\
&= \sum_{i=1}^n \sum_{j=1}^J \left(\ln(\alpha_j) + \ln(\gamma_{\mu_j, v_j}(X_i)) \right) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \\
&= \sum_{i=1}^n \sum_{j=1}^J \ln(\alpha_j) \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) + \sum_{i=1}^n \sum_{j=1}^J \ln(\gamma_{\mu_j, v_j}(X_i)) \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)
\end{aligned}$$

Traitons pour commencer la double somme

$$\Delta := \sum_{i=1}^n \sum_{j=1}^J \ln(\gamma_{\mu_j, v_j}(X_i)) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)$$

Nous avons :

$$\gamma_{\mu_j, v_j}(X_i) = \frac{1}{\sqrt{2\pi v_j}} e^{-\frac{1}{2} \frac{(X_i - \mu_j)^2}{v_j}}$$

et, d'après la proposition 2

$$\mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) = \frac{\alpha_j \times \gamma_{\mu_j, v_j}}{\sum_{k=1}^J \alpha_k \times \gamma_{\mu_k, v_k}}$$

La double somme devient alors

$$\begin{aligned}
\Delta &= \sum_{i=1}^n \sum_{j=1}^J \ln \left(\frac{1}{\sqrt{2\pi v_j}} e^{-\frac{1}{2} \frac{(X_i - \mu_j)^2}{v_j}} \right) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \\
&= \sum_{i=1}^n \sum_{j=1}^J \left[\ln \left(\frac{1}{\sqrt{2\pi v_j}} \right) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) - \frac{1}{2} \left(\frac{(X_i - \mu_j)^2}{v_j} \right) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \right] \\
&= \sum_{i=1}^n \sum_{j=1}^J \left[-\frac{1}{2} \ln(2\pi) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) - \frac{1}{2} \ln(v_j) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) - \frac{1}{2} \left(\frac{(X_i - \mu_j)^2}{v_j} \right) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \right] \\
&= -\frac{n}{2} \ln(2\pi) - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^J \left(\ln(v_j) + \frac{(X_i - \mu_j)^2}{v_j} \right) \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \\
&= -\frac{n}{2} \ln(2\pi) - \frac{1}{2} \sum_{j=1}^J \left(\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \times \left(\ln(v_j) + \frac{(X_i - \mu_j)^2}{v_j} \right) \right)
\end{aligned}$$

On obtient de fait le résultat espéré :

$$\begin{aligned}
\mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n) &= -\frac{n}{2} \log(2\pi) + \sum_{j=1}^J \left(\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \right) \times \ln(\alpha_j) \\
&\quad - \frac{1}{2} \sum_{j=1}^J \left(\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \times \left(\ln(v_j) + \frac{(X_i - \mu_j)^2}{v_j} \right) \right)
\end{aligned}$$

Nous allons dès à présent énoncer une proposition vitale, celle de l'expression des estimateurs du maximum de vraisemblance de la log-vraisemblance conditionnelle. L'expression de ces derniers seront le pivot de l'algorithme EM, que nous présenterons dans le chapitre suivant.

Proposition 7. *La fonction $\theta \mapsto \mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n)$ admet un unique maximum θ_M donné par :*

$$\begin{aligned}\widehat{\alpha}_j &= \frac{1}{n} \sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \\ \widehat{\mu}_j &= \frac{\sum_{i=1}^n X_i \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)}{\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)} \\ \widehat{v}_j &= \frac{\sum_{i=1}^n (X_i - \widehat{\mu}_j)^2 \times \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)}{\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)}\end{aligned}$$

Démonstration. Soit $\theta = (\alpha_j, \mu_j, v_j)_{j \in \llbracket 1, J \rrbracket}$. Il s'agit ici de maximiser la fonction $\theta \mapsto \mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n)$.

Puisqu'il s'agit d'un problème d'optimisation, nous appliquons la même méthode que précédemment, en introduisant le Lagrangien du problème sous la contrainte $\sum_{i=1}^n \alpha_i = 1$. Nous rappelons que le Lagrangien L est défini par

$$L : \begin{cases} \Theta \times \mathbb{R} & \longrightarrow & \mathbb{R} \\ (\theta, \lambda) & \longmapsto & \mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n) - \lambda \times \left(\sum_{j=1}^J \alpha_j - 1 \right) \end{cases}$$

Nous reprenons ici l'écriture de $\mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n)$ donnée dans la précédente proposition, nous obtenons ainsi l'expression suivante du Lagrangien

$$\begin{aligned}L(\theta, \lambda) &= -\frac{n}{2} \ln(2\pi) + \sum_{j=1}^J \left(\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \right) \ln(\alpha_j) \\ &\quad - \frac{1}{2} \sum_{j=1}^J \left(\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \times \left(\ln(v_j) + \frac{(X_i - \mu_j)^2}{v_j} \right) \right) - \lambda \left(\sum_{i=1}^n \alpha_i - 1 \right)\end{aligned}$$

Le Lagrangien admet un maximum sous la contrainte et ce maximum θ^* vérifie le système suivant :


$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \alpha_j}(\theta^*) = \frac{\sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)}{v_j} - \lambda & = 0 \quad \forall j \in \llbracket 1, J \rrbracket \\ \frac{\partial \mathcal{L}}{\partial \mu_j}(\theta^*) = \sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \times (-2X_i + 2\mu_j) & = 0 \quad \forall j \in \llbracket 1, J \rrbracket \\ \frac{\partial \mathcal{L}}{\partial v_j}(\theta^*) = -\frac{1}{2v_j} \sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) + \frac{1}{2v_j^2} \sum_{i=1}^n \mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i) \times (X_i - \mu_j)^2 & = 0 \quad \forall j \in \llbracket 1, J \rrbracket \\ \frac{\partial \mathcal{L}}{\partial \lambda}(\theta^*) = \sum_{i=1}^n \alpha_i - 1 & = 0 \end{cases}$$

Sous $\tilde{\theta}$ fixé, et ce qui est bien le cas, nous avons le fait que $\mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)$ est une constante. Le système devient alors :

$$\left\{ \begin{array}{ll} \alpha_j = \frac{\sum_{i=1}^n g_{\hat{\theta}}(j|X = X_i)}{\lambda} & \forall j \in \llbracket 1, J \rrbracket \\ \mu_j = \frac{\sum_{i=1}^n X_i \times \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)}{\sum_{i=1}^n \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)} & \forall j \in \llbracket 1, J \rrbracket \\ v_j = \frac{\sum_{i=1}^n (X_i - \mu_j)^2 \times \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)}{\sum_{i=1}^n \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)} & \forall j \in \llbracket 1, J \rrbracket \\ \sum_{i=1}^J \alpha_i = 1 \end{array} \right.$$

Sous la contrainte $\sum_{j=1}^J \alpha_j = 1$ et la première équation du système précédent on obtient l'égalité suivante :

$$\begin{aligned} \sum_{j=1}^J \alpha_j &= \sum_{j=1}^J \left(\frac{\sum_{i=1}^n \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)}{\lambda} \right) \\ &= \frac{\sum_{j=1}^J \sum_{i=1}^n \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)}{\lambda} \\ &= \frac{\sum_{i=1}^n \sum_{j=1}^J \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)}{\lambda} \\ &= \frac{\sum_{i=1}^n 1}{\lambda} = 1 \end{aligned}$$

On en déduit ainsi $\lambda = n$, ainsi que le résultat énoncé. 

Tous ces inesthétiques et fastidieux calculs n'ont pas été effectués en vain. Nous les avons réalisés suite à l'introduction d'une notion nouvelle, celle de la log-vraisemblance conditionnelle ; qui elle-même a été introduite faute de ne pouvoir obtenir une expression analytique de la log-vraisemblance des observations. Nous allons maintenant tâcher de mettre en exergue les raisons de ce développement théorique.

Chapitre 2

L'algorithme EM

Le présent chapitre aura une double tâche. La première est d'introduire l'élément fondamental de notre projet, l'algorithme EM. La seconde sera de mettre en lumière la réponse apportée par les éléments déjà introduit à notre problématique.

2.1 Présentation laconique et pseudo-code

Dans cette partie, nous allons faire le lien entre les outils développés dans le chapitre précédent, et expliciter en quel sens ils apportent une solution à notre problématique. Rappelons que cette dernière consiste en l'estimation de paramètres d'une log-vraisemblance des observations impossibles à maximiser. Comme mentionné lors de l'introduction, nous allons présenter l'algorithme EM. Il s'agit d'une méthode itérative, constituée de deux étapes ; à savoir une étape "Expectation" et une étape "Maximization". Dans la suite de ce chapitre, nous verrons que cet algorithme nous permet de calculer une suite de paramètres qui auront la particularité de faire croître la log-vraisemblance des observations à chaque itération.

Nous allons dans un premier temps décrire le fonctionnement de l'algorithme, l'explication de sa raison d'être n'en sera que plus limpide.

Dans ce qui suit, nous noterons par X_1, \dots, X_n l'échantillon observé, et par $\theta := (\alpha_j, \mu_j, \sigma_j)_{j \in \llbracket 1, J \rrbracket}$ le vecteur des paramètres.

2.1.1 L'étape E (Expectation)

La phase E consiste à calculer de manière effective non pas la log-vraisemblance conditionnelle $\mathcal{L}_c(\theta, \tilde{\theta}, X_1, \dots, X_n)$, mais seulement la probabilité

$$\mathbb{P}_{\tilde{\theta}}(Z = j | X = X_i)$$

puisque seule son expression est nécessaire pour obtenir les maximums établis à la proposition 7. Son calcul sera rendu possible grâce à la formule établie dans la proposition 2 et grâce à l'utilisation des paramètres $\alpha_j, \mu_j, \sigma_j$ calculés à l'itération précédente lors de l'étape M. À l'itération zéro, il est donc indispensable de renseigner des paramètres initiaux. Leur détermination occupera donc une place centrale dans notre projet et nous en discuterons dans une section ultérieure.

2.1.2 L'étape M (Maximization)

La phase M consiste à maximiser la vraisemblance trouvée à l'étape E. Ce maximum sera atteint en $\theta_M = (\hat{\alpha}_j, \hat{\mu}_j, \hat{\sigma}_j)$ et ces trois paramètres seront calculés à l'aide des maximas établis à la proposition 7. Une fois qu'ils auront été déterminés, ce sont ces trois paramètres qui seront utilisés dans l'itération suivante pour mettre à jour la log-vraisemblance conditionnelle lors de l'étape M.

2.1.3 Pseudo-code de l'algorithme EM

Pour l'implémentation de cet algorithme, nous nous sommes appuyés sur le pseudo-code suivant, inspiré de la référence [3].

Algorithm 1 L'algorithme EM (Dempster et al., 1977).

Entrée(s) : $\tilde{\theta}_0 \in \Theta$, un jeu de données $X_1 \cdots X_n$, $K \in \mathbb{N}$;

1: **pour** k allant de 1 à K **faire**

2: **ETAPE E :** Calculer la probabilité $\mathbb{P}_{\tilde{\theta}_{k-1}}(Z = j|X = X_i) = \frac{\alpha_j \times \gamma_{\mu_j, j_v}}{\sum_{k=1}^J \alpha_k \times \gamma_{\mu_k, v_k}}, \forall i \in \llbracket 1, n \rrbracket$

3: **ETAPE M :** Calculer $\tilde{\theta}_k = \underset{\theta=(\alpha_j, \mu_j, v_j)_{j \in \llbracket 1, J \rrbracket}}{\operatorname{argmax}} \mathbb{P}_{\tilde{\theta}_{k-1}}(Z = j|X = X_i)$;

4: **fin du pour**

5: **retourner** $\tilde{\theta}_K$;

2.1.4 Implémentation de l'algorithme EM

L'algorithme EM constitue sans aucun doute le pilier central de notre projet, une description de son implémentation semble donc indispensable.

```
algo_EM = function(data_init, X, K){
  J = dim(data_init)[1]
  n = length(X)
  data_stateE = param_State_E(n, J)

  for(k in 1:K){
    vect_alpha = data_init[,2] #de longueur J
    vect_mean = data_init[,3]
    vect_sd = data_init[,4]
    v = rep(0,n)

    # Etape E
    for(j in 1:J){
      data_stateE[,j] = vect_alpha[j]*dnorm(X,vect_mean[j],vect_sd[j])
      v = v+data_stateE[,j]
    }
    for(j in 1:J){
      data_stateE[,j] = data_stateE[,j]/v
    }

    # Etape M
    H = data_stateE
    for(col in 2:4){ #on met a jour le data_init
      for(ind in 1:n){
        # on met à jour les alpha
        if(col == 2){
          data_init[,col][ind] = mean(H[,ind])
        }
        # on met à jour les mu
        if(col == 3){
          data_init[,col][ind] = (sum(X*H[,ind]))/(sum(H[,ind]))
        }
        # on met à jour les sigma
        if(col == 4){
          data_init[,col][ind] = sqrt((sum( (X-rep(data_init[,col-1][ind],n))^2
            *H[,ind] ))/sum(H[,ind]))
        }
      }
    }
  }
}
```

```

new_df = data_init
colnames(new_df) = c('bird_names', 'alpha', 'mu', 'sigma')
return(new_df)
}

```

La fonction *algo_EM* prend en argument :

- *data_init*, le dataframe contenant les paramètres initiaux choisis à savoir α_{init} , μ_{init} , σ_{init}
- *X*, l'échantillon issu d'un mélange gaussien (il a été créé par la fonction *simulation* et est de taille *n*)
- *K* le nombre d'itérations de l'algorithme EM

Cette fonction va retourner un dataframe contenant les valeurs des paramètres α , μ et σ estimées par l'algorithme.

- Lors de l'étape E nous déterminons la probabilité $\mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)$ via la formule suivante :

$$\mathbb{P}_{\hat{\theta}}(Z = j|X = X_i) = \frac{\alpha_j \times \gamma_{\mu_j, j_v}}{\sum_{k=1}^J \alpha_k \times \gamma_{\mu_k, v_k}}$$

- Lors de l'étape M, nous déterminons les estimateurs du maximum de vraisemblance $(\widehat{\alpha}_j, \widehat{\mu}_j, \widehat{\sigma}_j)$ via les formules établies à la proposition 7 dont nous rappelons les expressions :

$$\begin{aligned} \widehat{\alpha}_j &= \frac{1}{n} \sum_{i=1}^n \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i) \\ \widehat{\mu}_j &= \frac{\sum_{i=1}^n X_i \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)}{\sum_{i=1}^n \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)} \\ \widehat{v}_j &= \frac{\sum_{i=1}^n (X_i - \widehat{\mu}_j)^2 \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)}{\sum_{i=1}^n \mathbb{P}_{\hat{\theta}}(Z = j|X = X_i)} \end{aligned}$$

Il n'existe pas de preuve de convergence de la suite de paramètres établie par l'algorithme EM ; cette dernière peut en effet stagner dans des extremas locaux. Le choix de bons paramètres initiaux est de fait primordial. Toutefois, nous sommes assurés que l'algorithme permet de faire croître la log-vraisemblance des observations en les paramètres itérativement créés, en témoigne la section suivante.

2.2 Une preuve de croissance

Dans cette concise section, nous énonçons un théorème essentiel. Ce dernier, et unique théorème est un résultat fondamental, il apporte une réponse à la raison d'être de l'algorithme, une justification à son élaboration.

Rappelons que notre problématique est de maximiser une log-vraisemblance des observations, qui présente une expression qu'il n'est possible de maximiser analytiquement. Nous cherchons donc à maximiser la quantité $\mathcal{L}_{obs}(\theta, X_1, \dots, X_n)$ définie à la proposition 2, en les paramètres $\alpha_1, \dots, \alpha_J, \mu_1, \dots, \mu_J, v_1, \dots, v_J$.

Le théorème suivant nous affirme, qu'à chaque itération k , l'algorithme EM produit une suite de paramètres

$$(\theta_k)_{k \in \llbracket 1, K \rrbracket} := (\alpha_{1_k}, \dots, \alpha_{J_k}, \mu_{1_k}, \dots, \mu_{J_k}, v_{1_k}, \dots, v_{J_k})_{k \in \llbracket 1, K \rrbracket}$$

qui ont la particularité de faire croître la log-vraisemblance des observations, K étant le nombre d'itérations de l'algorithme.

Enonçons précisément ce théorème et démontrons le.

Théorème 1. *Soit $(\theta_k)_{k \in \llbracket 1, K \rrbracket}$ la suite de paramètres construite à l'aide de l'algorithme EM. La log-vraisemblance \mathcal{L}_{obs} des observations vérifie*

$$\mathcal{L}_{obs}(\theta_{k+1}, X_1, \dots, X_n) \geq \mathcal{L}_{obs}(\theta_k, X_1, \dots, X_n)$$

Démonstration. Cette preuve s'inspirera des références [2] et [4].

Nous allons commencer cette preuve en donnant une autre forme de la log-vraisemblance, dépendant de $\mathcal{L}_{obs}(\theta, X_1, \dots, X_n)$ et d'un terme $\kappa_{\theta, \theta_k}$. Nous avons :

$$\begin{aligned} \mathcal{L}_c(\theta, \theta_k, X_1, \dots, X_n) &= \sum_{i=1}^n \sum_{j=1}^J \ln(h_\theta(X_i, j)) \mathbb{P}_{\theta_k}(Z = j | X = X_i) \\ &= \sum_{i=1}^n \sum_{j=1}^J \ln[f_\theta(X_i) \times \mathbb{P}_\theta(Z = j | X = X_i)] \mathbb{P}_{\theta_k}(Z = j | X = X_i) \\ &= \sum_{i=1}^n \sum_{j=1}^J \ln(f_\theta(X_i)) \mathbb{P}_{\theta_k}(Z = j | X = X_i) + \sum_{i=1}^n \sum_{j=1}^J \ln(\mathbb{P}_\theta(Z = j | X = X_i)) \mathbb{P}_{\theta_k}(Z = j | X = X_i) \\ &= \sum_{i=1}^n \ln(f_\theta(X_i)) \times \underbrace{\sum_{j=1}^J \mathbb{P}_{\theta_k}(Z = j | X = X_i)}_{=1} + \sum_{i=1}^n \sum_{j=1}^J \ln(\mathbb{P}_\theta(Z = j | X = X_i)) \mathbb{P}_{\theta_k}(Z = j | X = X_i) \\ &= \sum_{i=1}^n \ln(f_\theta(X_i)) + \sum_{i=1}^n \sum_{j=1}^J \ln(\mathbb{P}_\theta(Z = j | X = X_i)) \mathbb{P}_{\theta_k}(Z = j | X = X_i) \\ &= \mathcal{L}_{obs}(\theta, X_1, \dots, X_n) + \kappa_{\theta, \theta_k} \end{aligned}$$

Dès lors, on obtient

$$\mathcal{L}_{obs}(\theta_{k+1}, X_1, \dots, X_n) - \mathcal{L}_{obs}(\theta_k, X_1, \dots, X_n) = \mathcal{L}_c(\theta_{k+1}, \theta_k, X_1, \dots, X_n) - \kappa_{\theta_{k+1}, \theta_k} - \mathcal{L}_c(\theta_k, \theta_k, X_1, \dots, X_n) + \kappa_{\theta_k, \theta_k}$$

Or, la quantité \mathcal{L}_c est maximisée en θ_{k+1} lors de l'étape M de l'algorithme, donc

$$\mathcal{L}_c(\theta_{k+1}, \theta_k, X_1, \dots, X_n) - \mathcal{L}_c(\theta_k, \theta_k, X_1, \dots, X_n) \geq 0$$

Il reste donc à prouver que

$$\kappa_{\theta_k, \theta_k} - \kappa_{\theta_{k+1}, \theta_k} \geq 0$$

En effet, nous avons

$$\begin{aligned}
\kappa_{\theta_k, \theta_k} - \kappa_{\theta_{k+1}, \theta_k} &= \sum_{i=1}^n \sum_{j=1}^J \ln(\mathbb{P}_{\theta_k}(Z = j|X = X_i)) \times \mathbb{P}_{\theta_k}(Z = j|X = X_i) \\
&\quad - \sum_{i=1}^n \sum_{j=1}^J \ln(\mathbb{P}_{\theta_{k+1}}(Z = j|X = X_i)) \times \mathbb{P}_{\theta_k}(Z = j|X = X_i) \\
&= \sum_{i=1}^n \sum_{j=1}^J \ln\left(\frac{\mathbb{P}_{\theta_k}(Z = j|X = X_i)}{\mathbb{P}_{\theta_{k+1}}(Z = j|X = X_i)}\right) \times \mathbb{P}_{\theta_k}(Z = j|X = X_i) \\
&= -n \sum_{i=1}^n \sum_{j=1}^J \ln\left(\frac{\mathbb{P}_{\theta_{k+1}}(Z = j|X = X_i)}{\mathbb{P}_{\theta_k}(Z = j|X = X_i)}\right) \times \mathbb{P}_{\theta_k}(Z = j|X = X_i) \times \frac{1}{n} \\
&\geq -n \times \ln\left(\sum_{i=1}^n \sum_{j=1}^J \frac{\mathbb{P}_{\theta_{k+1}}(Z = j|X = X_i)}{\mathbb{P}_{\theta_k}(Z = j|X = X_i)} \times \mathbb{P}_{\theta_k}(Z = j|X = X_i) \times \frac{1}{n}\right) \\
&\quad \left[\text{Cette dernière inégalité est due à la convexité de } \ln \text{ et au fait que } \sum_{i=1}^n \sum_{j=1}^J \mathbb{P}_{\theta_k}(Z = j|X = X_i) \times \frac{1}{n} = 1\right] \\
&= -n \times \ln\left(\sum_{i=1}^n \sum_{j=1}^J \mathbb{P}_{\theta_{k+1}}(Z = j|X = X_i) \times \frac{1}{n}\right) \\
&= -n \times \ln(1) \\
&= 0
\end{aligned}$$

On obtient ainsi

$$\kappa_{\theta_k, \theta_k} - \kappa_{\theta_{k+1}, \theta_k} \geq 0$$

Et finalement

$$\mathcal{L}_{obs}(\theta_{k+1}, X_1, \dots, X_n) \geq \mathcal{L}_{obs}(\theta_k, X_1, \dots, X_n)$$

☞

Comme sus-mentionné, ce résultat ne prétend pas que l'algorithme EM maximisera la log-vraisemblance des observations. Ce dernier peut tout à fait produire une suite de paramètres correspondant à un maximum local ; et une augmentation du nombre d'itérations n'apportera pas de solution à ce problème.

Chapitre 3

Etude de simulations

Dans ce troisième chapitre, il s'agira de présenter des algorithmes complémentaires à celui de l'algorithme EM. Ces derniers nous permettront de mener à bien notre brève étude sur "l'efficacité" de notre implémentation, et ce pour divers mélanges.

3.1 La fonction simulation

Afin de pouvoir exécuter l'algorithme EM, nous avons besoin d'un échantillon issu d'un mélange gaussien. Comme nous le verrons dans le chapitre 4, nous disposons dans le cadre de ce projet uniquement de données répertoriant les moyennes et les écarts-types des volumes des nids de treize espèces d'oiseaux différentes. Grâce à ces données et à la commande $rnorm(\mu, \sigma)$ de *R*, nous avons implémenté la fonction *simulation*, qui a pour mission de générer aléatoirement un échantillon d'un mélange Gaussien. Il s'agit d'une des plus importantes fonction de ce projet, nous allons donc la décrire.

```
simulation = function(data_th, n=100){
  X = rep(NA,n) #echantillon
  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)
}
```

La fonction prend en argument *data_th* (le dataframe contenant les paramètres α , μ et σ) et n , qui indique la taille de l'échantillon à générer aléatoirement.

La fonction retourne le vecteur du mélange gaussien de taille n qui a été généré aléatoirement.

La partie la plus importante et subtile de ce script est celle dans laquelle nous distribuons aléatoirement les lois des différents $(X_i)_{i \in \{1, \dots, n\}}$ de l'échantillon, de sorte à avoir un bon mélange gaussien.

Pour simplifier les choses nous allons prendre un exemple dans lequel nous voulons générer un mélange de trois gaussiennes, ayant pour paramètres respectifs $\theta_1 = (\alpha_1, \mu_1, \sigma_1)$, $\theta_2 = (\alpha_2, \mu_2, \sigma_2)$ et $\theta_3 = (\alpha_3, \mu_3, \sigma_3)$. Nous

rappelons une fois de plus que $\sum_{j=1}^3 \alpha_j = 1$. Nous procédons de la manière suivante :

Nous générons une variable aléatoire Z de loi uniforme à l'aide de la commande *runif* de *R*; puis :

- Si $Z < \alpha_1$ alors $X \sim \mathcal{N}(\mu_1, \sigma_1)$
- Sinon si $\alpha_1 \leq Z \leq \alpha_1 + \alpha_2$, alors $Z \sim \mathcal{N}(\mu_2, \sigma_2)$
- Sinon si $\alpha_1 + \alpha_2 \leq Z \leq \alpha_1 + \alpha_2 + \alpha_3$, alors $Z \sim \mathcal{N}(\mu_3, \sigma_3)$

Notre implémentation fonctionne dans le cas général d'un mélange de J gaussiennes, $J \in \mathbb{N}^*$, et repose exactement sur le même principe que celui précédemment expliqué.

3.2 Étude de simulations

Les différentes sections ci-dessous auront pour vocations de présenter plusieurs simulations, dans le but de tester l'efficacité de notre implémentation de l'algorithme EM.

3.2.1 Étude d'un mélange gaussien à "forte séparation"

Dans cette partie, nous nous intéresserons à un mélange de trois lois gaussiennes, ayant comme lois respectives $\mathcal{N}(\mu_1, \sigma_1)$, $\mathcal{N}(\mu_2, \sigma_2)$ et $\mathcal{N}(\mu_3, \sigma_3)$ bien séparées, c'est à dire avec des paramètres bien différents les uns des autres (i.e. $|\alpha_i - \alpha_j|_{i \neq j}$; $|\mu_i - \mu_j|_{i \neq j}$; $|\sigma_i - \sigma_j|_{i \neq j}$ assez grands). Nous avons généré un échantillon d'un mélange de trois gaussiennes à fortes séparations de taille $n = 500$. Les trois lois du mélange sont les suivantes :

- $\mathcal{N}(11, 1)$
- $\mathcal{N}(49, 10)$
- $\mathcal{N}(92, 3)$

La figure ci-dessous représente la distribution du mélange

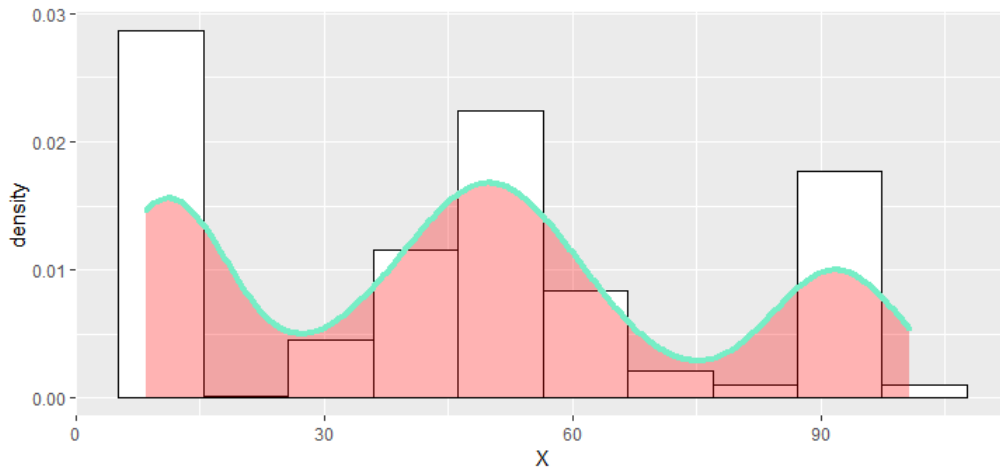


FIGURE 3.1 – Mélange gaussien à "forte séparation"

Nous avons lancé l'algorithme EM en choisissant un nombre d'itération $k = 10$. Comme on peut le voir sur la capture d'écran ci-dessous, les paramètres estimés par l'algorithme EM sont assez proche de ceux théoriques.

```
> print(good_dfTestTh3)
  bird_names proportion_alpha mean_th sd_th
1 Good species 1          0.3      11     1
2 Good species 2          0.5      49    10
3 Good species 3          0.2      92     3
> algo_EM(good_dfTestInit3, xGood3, 10)
  bird_names      alpha      mu      sigma
1 Good species 1 0.2939353 11.05072 0.9611203
2 Good species 2 0.5042109 49.33787 9.7178553
3 Good species 3 0.2018537 91.92101 2.9731759
```

FIGURE 3.2 – Paramètres théoriques VS paramètres estimés

Nous allons maintenant regarder si l'augmentation de la taille de l'échantillon du mélange gaussien avec "fortes séparations" aura pour effet de diminuer l'erreur absolue entre les paramètres estimés par l'algorithme EM et les valeurs théoriques. Pour cela nous allons utiliser notre fonction *Monte-Carlo*.

```
Monte_Carlo = function(data_th, data_init, k, n, N){
  J = dim(data_th)[1]
  df_MonteCarlo = data.frame()
  iteration = c()
  for(i in 1:k){
    X = simulation(data_th, n)
    data_EM = algo_EM(data_init, X, N)
    df_error = calcul_error(data_th, data_EM)
    df_MonteCarlo = rbind(df_MonteCarlo, df_error)
    v_iter = rep(paste("itération", i, sep="_"), J)
    iteration = append(iteration, v_iter)
  }
  df_MonteCarlo = cbind(iteration, df_MonteCarlo)
  return(df_MonteCarlo)
}
```

Le principe de fonctionnement de cette dernière est le suivant :

Pour une taille d'échantillon n fixée, nous exécutons k fois l'algorithme EM pour des échantillons différents. À partir de cela, nous calculons les $3 \times 3 \times K$ erreurs absolues entre les paramètres estimés par l'algorithme EM et les paramètres théoriques. Cette fonction prend donc en arguments :

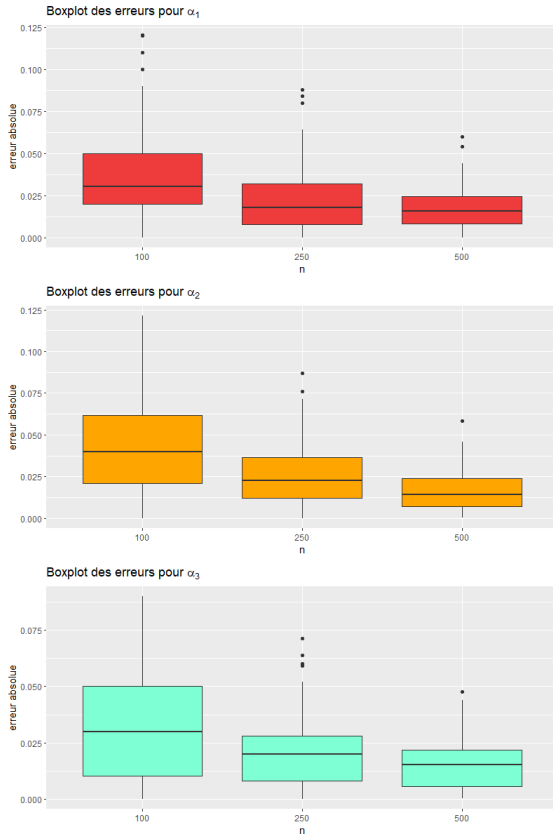
- data_th , le dataframe contenant les paramètres α , μ et σ théoriques
- data_init , le dataframe contenant les paramètres (α, μ, σ) initialement choisis
- k , le nombre d'échantillons que l'on souhaite générer aléatoirement
- n la taille des échantillons (les k échantillons seront de tailles n), et
- N , le nombre d'itérations de l'algorithme EM

Elle retourne un dataframe contenant les erreurs absolues de chaque paramètres des k échantillons de Monte Carlo.

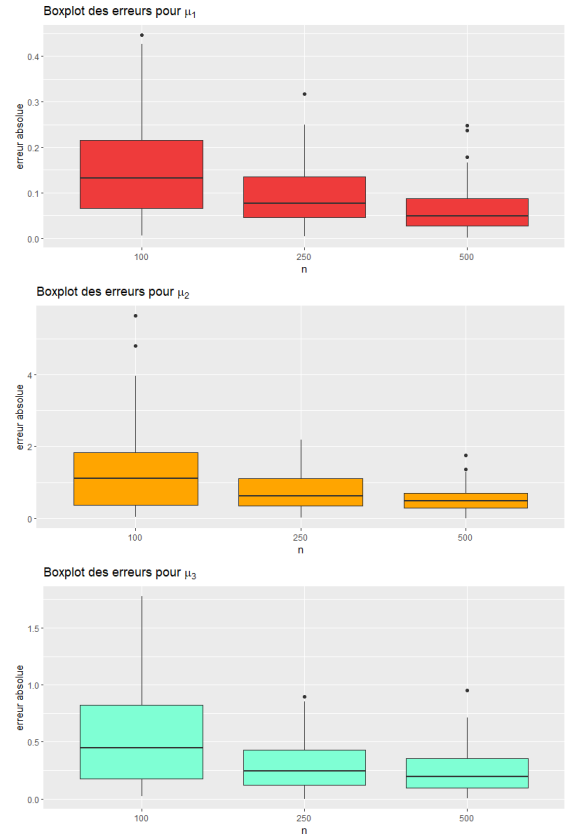
Nous avons effectué 100 itérations de Monte-Carlo pour les différentes tailles d'échantillons suivantes :

- $n = 100$
- $n = 250$
- $n = 500$

Les boîtes à moustaches (boxplot) ci-dessous, modélisent l'erreur absolue entre les paramètres μ et σ estimés par l'algorithme EM et les paramètres $(\alpha_i)_{i \in \llbracket 1,3 \rrbracket}$ et $(\mu_i)_{i \in \llbracket 1,3 \rrbracket}$ théoriques.



(a) Boxplot des erreurs absolues pour α_1 , α_2 et α_3



(b) Boxplot des erreurs absolues pour μ_1 , μ_2 et μ_3

FIGURE 3.3 – Boxplot des erreurs absolues de $(\alpha_j)_{j \in \{1 \dots 3\}}$ et $(\mu_j)_{j \in \{1 \dots 3\}}$

En observant les boxplots de la figure 3.3, nous remarquons que plus la taille n de l'échantillon est grande, plus la longueur des boxplots ainsi que leurs médianes diminuent. En d'autre terme, cela indique que l'erreur absolue entre les paramètres estimés par l'algorithme EM et ceux théoriques diminue. Nous en déduisons donc que, dans le cas d'un mélange gaussien "à fortes séparations", l'augmentation de la taille de l'échantillon a pour effet d'améliorer la précision de l'algorithme EM. D'ailleurs, dans le cas ici présent, les estimateurs produits par ce dernier convergent effectivement vers les valeurs théoriques des paramètres.

3.2.2 Étude d'un mélange gaussien à "faible séparation"

Dans cette partie, nous nous intéresserons à un mélange de trois lois gaussiennes ayant comme loi respectives $\mathcal{N}(\mu_1, \sigma_1)$, $\mathcal{N}(\mu_2, \sigma_2)$ et $\mathcal{N}(\mu_3, \sigma_3)$ faiblement séparées, c'est à dire avec des paramètres assez proches les uns des autres (i.e. $|\alpha_i - \alpha_j|_{i \neq j}$; $|\mu_i - \mu_j|_{i \neq j}$; $|\sigma_i - \sigma_j|_{i \neq j}$ assez petits). De même que pour la première étude, nous avons généré un échantillon d'un mélange de trois lois gaussiennes de taille 500, cette fois-ci avec "faible séparation". Les trois lois du mélange sont les suivantes :

- $\mathcal{N}(5, 1)$
- $\mathcal{N}(13.6, 11)$
- $\mathcal{N}(15.3, 6)$

La figure ci-dessous représente la distribution du mélange à "faible séparation" ;

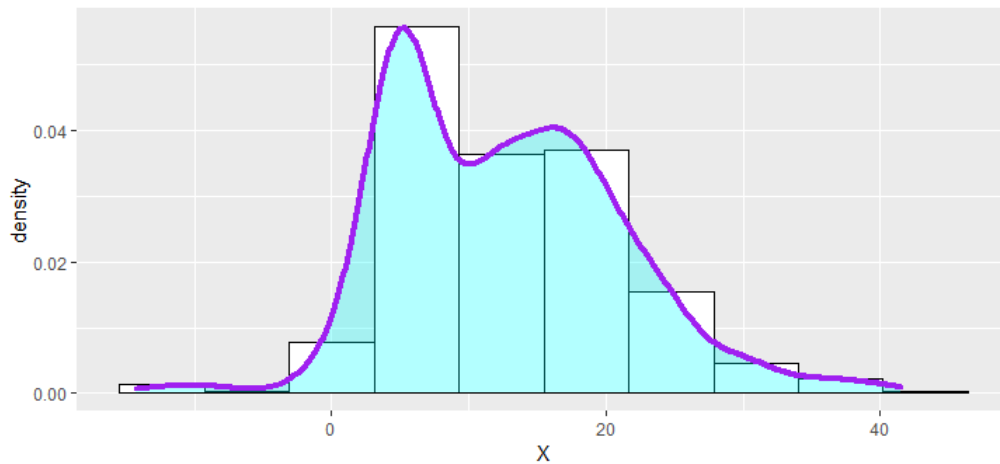


FIGURE 3.4 – Mélange gaussien à faibles séparations

Ici aussi, nous avons lancé l'algorithme EM en choisissant un nombre d'itération $k = 10$. Comme nous pouvons le voir sur la capture d'écran ci-dessous, même avec un mélange à faibles séparations, les paramètres estimés par l'algorithme EM sont assez proches de ceux théoriques.

```
> print(bad_dfTestTh3)
  bird_names proportion_alpha mean_th sd_th
1 Bad species 1          0.24     5.0    1
2 Bad species 2          0.36    13.6   11
3 Bad species 3          0.40    15.3    6
> algo_EM(bad_dfTestInit3, xBad3, 10)
  bird_names  alpha      mu  sigma
1 Bad species 1 0.2508061  4.93471 1.143349
2 Bad species 2 0.5426620 14.56490 8.936383
3 Bad species 3 0.2065318 16.73281 5.410275
```

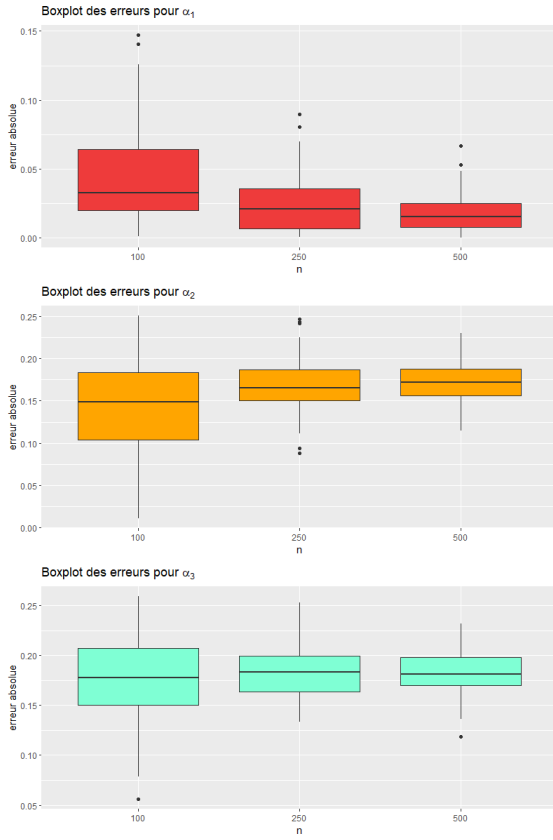
FIGURE 3.5 – Paramètres théoriques VS paramètres estimés

De même que pour l'étude précédente, nous allons maintenant regarder si l'augmentation de la taille de l'échantillon du mélange gaussien généré aléatoirement aura pour effet de diminuer l'erreur absolue entre les paramètres estimés par l'algorithme EM et les valeurs théoriques dans le cas d'un mélange à faibles séparations. Pour cela nous allons de nouveau utiliser notre fonction *Monte-Carlo*.

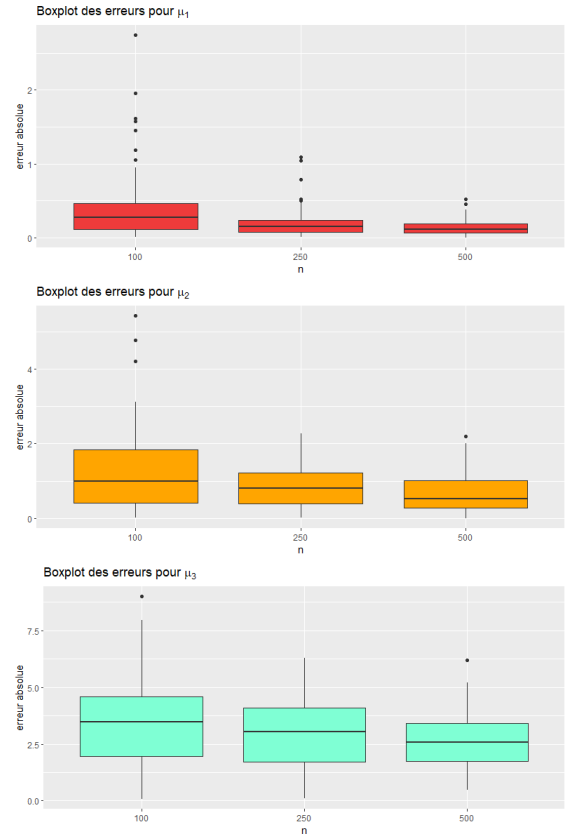
Nous avons effectué 100 itérations de Monte-Carlo pour les différentes tailles d'échantillons suivantes :

- $n = 100$
- $n = 250$
- $n = 500$

Les boîtes à moustaches (boxplot) ci-dessous, modélisent l'erreur absolue entre les paramètres estimés par l'algorithme Em et ceux théoriques dans le cas d'une "faibles séparations".



(a) Boxplot des erreurs absolues pour α_1 , α_2 et α_3



(b) Boxplot des erreurs absolues pour μ_1 , μ_2 et μ_3

FIGURE 3.6 – Boxplot des erreurs absolues de $(\alpha_j)_{j \in \{1 \dots 3\}}$ et $(\mu_j)_{j \in \{1 \dots 3\}}$

Concernant les boxplots des erreurs absolues des paramètres $(\mu_i)_{i \in \{1 \dots 3\}}$ et $(\sigma_i)_{i \in \{1 \dots 3\}}$ de la figure 3.6, les conclusions sont les mêmes que dans le cas d'un mélange gaussien avec "fortes séparations". En effet, leurs erreurs absolues diminuent lorsque la taille n de l'échantillon croît. Nous observons cependant quelque chose d'inhabituelle chez les paramètres α_2 et α_3 . Leurs erreurs absolues augmentent quand n croît alors que leurs variances diminuent. Cela illustre bien que les paramètres estimés par l'algorithme EM ne convergent pas toujours vers les paramètres théoriques.

3.3 Automatisation du choix des paramètres initiaux

Dans le cas éventuel où l'utilisateur ne souhaite pas ou ne peut pas effectuer de décisions quant au choix des paramètres initiaux de l'algorithme EM, nous proposons de présenter trois fonctions automatisant ce choix.

Nous présenterons dans un premier temps les deux fonctions *param_quantile1* et *param_quantile2*. Elles reposent sur une approche plus simpliste et rendent compte, pour le choix des paramètres $(\alpha_j)_{j \in \{1, \dots, J\}}$ (proportions des J espèces observées) et $v_{j \in \{1, \dots, J\}}$ (variances des volumes des J espèces observées), des idées suivantes :

Si l'échantillon est obtenu aléatoirement, sur une zone d'étude tirée aléatoirement, on peut supposer que la proportion de chacune des espèces sur cette zone est approximativement la même. Nous choisirons ainsi des proportions égales pour le choix d'initialisation des paramètres $(\alpha_j)_{j \in \{1, \dots, J\}}$.

De même, il est vraisemblable de ce dire que, pour des espèces semblables, chacune fait des nids de volumes approximativement égaux. Ainsi, nous pourrions choisir comme écarts-types initiaux, des écarts-types égaux.

La distinction de ces deux fonctions vient quant au choix des moyennes des espèces observées. Pour ce faire, nous allons décrire les codes de ces deux fonctions.

3.3.1 Fonction param_quantile1

```
param_quantile1 <- function(X,J){
  N <- length(X)
  Alpha = rep(1/J,J)
  Mean <- rep(NA,J)
  X <- sort(X)

  for(j in 1:J){
    Mean[j] <- X[floor(j*N/J+1)]
  }

  Sd <- sqrt(var(X[1:Mean[1]]))
  Sd <- rep(Sd,J)

  data <- data.frame(init_alpha = Alpha, init_mu = Mean, init_sd = Sd)
  return(data)
}
```

La fonction *param_quantile1* prend en arguments :

- X , l'échantillon observé
- J , le nombre d'espèces observées

Comme décrit en début de section, nous choisirons des proportions égales entre espèces. Nous avons alors

$$\alpha_j = \frac{1}{J}, \forall j \in \{1, \dots, J\}.$$

Puisque que les espèces font (dans l'idée de cette fonction) des nids semblables, nous allons trier notre échantillon.

Pour le choix des moyennes le principe est le suivant :

- Nous découpons l'intervalle des observations uniformément en $J + 1$ intervalles
- Nous prendrons comme moyennes les J premières valeurs associées à ces coupures ; cela correspondra approximativement à prendre les J premiers quantiles.

Le fait de supposer des proportions égales entre les espèces permet de voir chacun des intervalles comme les valeurs associées aux volumes de chaque espèce, nous aurons ainsi pour le j -ème intervalle formé les valeurs des volumes des nids de l'espèce j .

Quant aux choix des écart-types de chaque espèce, nous choisirons l'écart-type de la première espèce.

Décrivons maintenant la seconde fonction.

3.3.2 Fonction param_quantile2

```
param_quantile2 <- function(X,J){
  N <- length(X)
  Alpha = rep(1/J,J)
  Mean <- rep(NA,J)
  Sd <- rep(NA,J)
  X <- sort(X)

  index1_Q <- 1
  index2_Q <- 0
  for(j in 1:J-1){
    index2_Q <- floor(j*N/J)+1
    Mean[j] <- mean(X[index1_Q:index2_Q])
    index1_Q <- index2_Q+1
    Sd[j] <- sqrt(var(X[index1_Q:index2_Q]))
  }
  Mean[J] <- mean(X[index1_Q:N])
  Sd[J] <- sqrt(var(X[index1_Q:N]))
}
```



```

data <- data.frame(init_alpha = Alpha, init_mu = Mean, init_sd = Sd)
return(data)
}

```

De la même façon que la fonction *param_quantile1*, la fonction *param_quantile2* prend comme arguments :

- X , l'échantillon observé
- J , le nombre d'espèces observées

Et nous choisirons comme proportions initiales :

$$\alpha_j = \frac{1}{J}, \forall j \in \{1, \dots, J\}.$$

La distinction qui se fait entre cette fonction et la précédente est le choix des moyennes pour chaque espèce. Le choix ici est le suivant : nous effectuons comme précédemment un partitionnement (uniforme suivant la parité) de l'échantillon. Puisque les intervalles formés correspondent approximativement aux valeurs du volume des nids de chaque espèce (suivant l'idée d'avoir des proportions égales d'espèces) la fonction choisie de prendre comme moyenne d'une espèce j , la moyenne du j -ème intervalle.

Pour les écarts-types, nous choisirons les écarts-types de chaque intervalles.

Remarque 3. Même sous réserve d'avoir des proportions d'espèces proches, il n'est pas assuré que l'une des deux fonctions donne une meilleure initialisation de l'algorithme EM. Nous pouvons envisager dans ce cas là, de considérer les deux fonctions et de tester leurs résultats en appliquant l'algorithme EM. Il s'agira donc d'évaluer et de comparer la log-vraisemblance de la variable X en les valeurs de sortie de l'algorithme EM, avec pour choix de paramètres initiaux, ceux des fonctions présentées. On retiendra comme initialisation, celles dont la log-vraisemblance de X sera la plus grande.

Afin de se donner le plus de chance quant à la détermination des estimateurs du maximum de vraisemblance de X , nous envisageons de considérer la fonction d'initialisation *param_kmeans*, en espérant qu'elle soit plus efficace que les deux précédentes. Par soucis de présentation, celle-ci sera donnée en Annexe.

Décrivons toutefois son code et son principe :

3.3.3 Fonction *param_kmeans*

Cette fonction choisi en argument :

- X , l'échantillon observé
- J , le nombre d'espèce observées

Il s'agit dans cette fonction de choisir une initialisation des paramètres initiaux dans un cadre plus général que celui des deux fonctions précédentes (en supposant égalité des coefficients α_j ou encore des v_j).

La démarche est la suivante :

Etant donné une variable aléatoire Y suivant une loi normale de paramètres μ, σ quelconques, la moyenne de variable Y correspondra au maximum de ses valeurs.

Dans ce cas, et si nous disposons d'un échantillon "plus ou moins bien hétérogène" (quant aux paramètres des lois normales du mélange), il est possible de choisir comme initialisation des moyennes, les moyennes des espèces. Dans le cas d'un mélange plus "homogène", il est possible de relever sur la densité de X moins de maximums que de nombre de lois normales. Il peut y avoir plusieurs raison à cela, des moyennes très similaires, des proportions très faibles d'espèces, des écarts-types très élevés ; tout ceci faisant confondre possiblement les lois normales entre elles.

Pour remédier à ce problème, et pour la suite du choix des paramètres, même dans des cas d'hétérogénéité, nous allons introduire un partitionnement de l'échantillon par méthode des k-means.

La fonction *param_kmeans* va effectuer un partitionnement par méthode des k-means sur le choix obtenu des moyennes (en les maximums de la densité de X). Cette méthode permettra de rassembler les valeurs les plus proches de ces centres ensembles, il sera ensuite aisé de déterminer une approximation de la proportion de chaque espèce ainsi que l'écart-type associé.

Cependant, comme sus-mentionné, il est possible que l'on ait un mélange homogène entre les espèces. Ceci peut amener au fait d'obtenir moins de maximums qu'il y a d'espèces présentes ; et peut biaiser une bonne initialisation des moyennes.

Pour tenter de résoudre ceci, la fonction utilise un k-means sur ces centres obtenus par maximum de la densité de X . Puis elle retient le cluster ayant le plus grand nombre de valeurs et effectue un nouveau k-means sur ce sous-échantillon, avec deux centres initiaux. Ces centres seront choisis automatiquement par la fonction *kmeans()* de *R* parmi un choix de 25 initialisations.

De cette manière nous actualisons le choix des moyennes, la fonction effectuera autant de répétitions que nécessaire, jusqu'à avoir un nombre de moyennes égale au nombre d'espèces. Enfin, elle effectue un nouveau k-means sur ces moyennes et relève les proportions associées aux clusters formés ainsi que leurs écarts-types.

Nous allons maintenant décrire quelques parties du code de la fonction *param_kmeans* :

```
if (size_m_X > J){
  min <- m_X[m_X < mean(X)]
  max <- m_X[m_X > mean(X)]
  min_max <- sort(c(abs(min-mean(X)),abs(max-mean(X))))
  i <- 1
  while (size_m_X > J){
    if (!is.null(min_max)){
      m_X <- m_X[m_X != min_max[i]] #On enlève les max les plus éloignés
      size_m_X <- length(m_X)
    }
    else{
      m_X2 <- sort(m_X)
      m_X <- m_X[m_X != m_X2[i]] #On enlève les max les plus faibles
    }
    i <- i+1
  }
}
```

La densité de la variable X (des volumes des nids) peut faire apparaître, à cause d'un mélange "plus ou moins" homogène ou par l'approximation de la densité de X , des maximums non significatifs, ce qui peut engendrer des problèmes dans notre démarche. Il s'agira alors d'essayer de les supprimer.

Pour ce faire nous relèverons tous les maximums en dehors de l'échantillon, et nous supprimons une part de ceux qui sont les plus éloignés des bornes de l'échantillon. Si cela ne suffit pas, nous supprimons les moyennes les moins significatives (les plus faibles) jusqu'à avoir un nombre de maximums égal au nombre d'espèces. Il s'agira également dans le choix d'initialisation des moyennes de remplacer les maximums par les centres des clusters formés, afin d'éviter le risque d'une moyenne non significative.

Pour finir cette section, il conviendra d'étudier ces trois fonctions comme dans la précédente remarque. Ceci sera effectué en annexe, l'essentiel des scripts y seront décrit.

Chapitre 4

Un modèle pour les nids d'oiseaux

Dans ce quatrième et dernier chapitre, nous nous proposons de mettre en oeuvre ce que nous avons développé dans les précédents chapitres. Nous allons nous placer en situation réelle, et étudier un jeu de données correspondant à un mélange de lois gaussiennes. Notre thématique ne changera pas, il sera toujours question de nids d'oiseaux.

4.1 Préambule

Nous n'avons hélas pu trouver des jeux de données sur les volumes de nids d'oiseaux en libre accès. Toutefois, nous avons récupéré sur le site de l'*Université de Lincoln (Royaume-Uni)* la prépublication [5], contenant des informations fort utiles sur une douzaine d'espèces d'oiseaux. En voici un extrait :

	Female Body Mass (g)	Total mass of nest (g)	Cup diameter parallel to long axis (mm)	Cup diameter perpendicular to long axis (mm)	Nest diameter parallel to long axis (mm)	Nest diameter perpendicular to long axis (mm)	Upper wall thickness (mm)	Base Thickness (mm)	Cup depth (mm)	Nest Height (mm)	Volume (cm ³)
Fringillidae											
European Goldfinch (<i>Carduelis Carduelis</i>) [10]	16.4	8.3 ± 2.4	62.8 ± 12.1	54.8 ± 7.4	91.4 ± 9.3	77.8 ± 7.9	12.8 ± 3.3	15.7 ± 4.3	26.0 ± 5.5	41.6 ± 7.4	38.0 ± 9.1
Common Linnet (<i>Linaria cannabina</i>) [11]	18.0	18.9 ± 5.4	74.7 ± 6.3	59.9 ± 8.6	107.9 ± 8.8	95.1 ± 10.2	16.9 ± 4.9	24.5 ± 8.9	30.6 ± 9.8	55.1 ± 9.2	60.9 ± 20.8
Common Chaffinch (<i>Fringilla coelebs</i>) [11]	21.5	14.5 ± 2.9	63.3 ± 8.1	50.8 ± 8.0	98.7 ± 10.9	90.3 ± 9.8	18.5 ± 3.6	23.6 ± 7.6	34.3 ± 7.8	58.0 ± 7.3	58.3 ± 15.0
European Greenfinch (<i>Chloris chloris</i>) [5]	25.9	22.4 ± 6.2	75.6 ± 7.8	53.9 ± 11.8	128.6 ± 13.7	99.7 ± 16.2	24.9 ± 7.9	29.4 ± 6.0	35.4 ± 5.7	64.9 ± 9.4	74.5 ± 12.2
Eurasian Bullfinch (<i>Pyrrhula pyrrhula</i>) [17]	27.3	12.1 ± 4.6	80.8 ± 12.1	66.4 ± 8.1	129.7 ± 23.4	117.5 ± 19.6	24.8 ± 10.9	24.2 ± 10.7	22.6 ± 4.5	46.8 ± 11.3	45.0 ± 3.8
Hawfinch (<i>Coccothraustes coccothraustes</i>) [4]	52.9	27.4 ± 7.3	102.2 ± 17.9	78.8 ± 25.2	153.4 ± 19.1	131.3 ± 27.1	25.4 ± 5.9	23.3 ± 4.9	31.4 ± 10.9	54.7 ± 11.5	71.6 ± 12.9

FIGURE 4.1 – Caractéristiques des nids (d'après [5])

Parmi elles ce trouve des volumes moyens de nids ainsi que la variance associée.

Afin de se conformer aux hypothèses que nous avons précédemment mentionné, nous supposons ici que les volumes des nids suivent une loi normale. Nous ferons une dernière hypothèse ; nous supposons le nombre d'espèces J connu lors des observations.

Plaçons nous dès à présent dans le cadre d'une étude. Pour ce faire, commençons par établir une méthodologie. Il sera en premier lieu choisi, aléatoirement, plusieurs espèces d'oiseaux parmi celles que nous avons à disposition. Puis nous allons générer, avec R , les données dont nous avons besoin. Cet n -échantillon sera créé grâce à notre fonction de simulation. Notre jeu de données étant prêt, nous pourrons commencer l'étude.

Le premier problème qui se posera sera celui du choix des paramètres initiaux, notre algorithme nécessitant des valeurs initiales. Nous avons diverses solutions à notre portée.

- ✂ La première est de faire une exploration des données ; via une représentation graphique de la densité de l'échantillon. Pour ce faire nous utilisons la fonction pré-implémentée dans *ggplot*. Cette dernière se base sur une méthode d'estimation non-paramétrique, la méthode d'estimation par noyau. Décrivons brièvement son fonctionnement. C'est une méthode qui estime point par point la densité de l'échantillon, via la fonction $\widehat{f}_n(x)$ définie comme suit

$$\widehat{f}_n(x) = \frac{1}{n \times h} \sum_{i=1}^n K\left(\frac{x - X_i}{h}\right)$$

où h est la taille de la fenêtre et n la taille de l'échantillon. Il s'agit, en quelque sorte, d'une moyenne "locale" en x avec les points de l'échantillon X_i qui appartiennent à la fenêtre $[x - h, x + h]$. K est une fonction de noyau. Souvent, elle est choisie comme une densité gaussienne centrée réduite.

Nous pourrions dès lors récupérer des informations bien utiles ; si des pics bien distincts se présentent, cela nous permettra de régler judicieusement les moyennes et variances initiales. Nous détaillerons ceci ultérieurement.

- ✂ Une seconde solution est d'utiliser les quantiles pour l'estimation des moyennes, et l'écart-type empirique pour les variances initiales. Nous détaillerons également cela lors de l'étude.

Une fois cela fait, nous pourrions exécuter notre algorithme et obtenir son estimation des paramètres. Nous pourrions dès lors émettre les conclusions appropriées.

Nous partons, pour ainsi dire, à l'aveugle, mais il sera gardé en mémoire les vraies valeurs, afin de vérifier si nous avons abouti aux bonnes conclusions.

Nous avons, à partir du tableau de données sus-mentionné, construit le data-frame *nest_data* reprenant les noms des espèces, ainsi que les volumes moyens et variances associées. Ce dernier nous servira à créer les échantillons.

4.2 Etude d'un mélange à deux lois

Dans cette section, nous allons étudier le cas d'un mélange de deux lois. Commençons par sélectionner aléatoirement deux espèces, ainsi que les proportions associées. Pour ce faire, nous utiliserons notre fonction, la fonction *random_species*.

```
df <- random_species(nest_data, 2)
```

Nous pouvons dès lors générer l'échantillon, à l'aide de notre fonction *simulation*.

```
data <- simulation(df, 500)
```

Nous commençons par tracer la courbe de densité associée à l'échantillon.

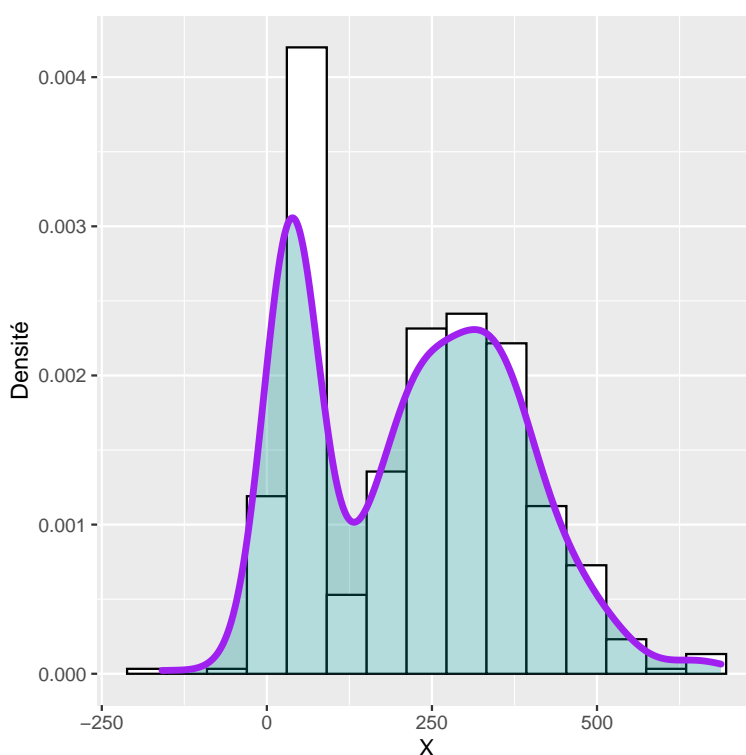


FIGURE 4.2 – Densité de l'échantillon estimée par une méthode à noyau

Le mélange des deux lois se distingue au premier coup d'oeil ; nous distinguons nettement deux pics. L'"étalement" des pics n'étant pas les mêmes, nous pouvons supposer sans craintes que les variances des deux lois sont différentes. Il est cependant difficile d'émettre des hypothèses quant aux proportions.

Nous proposons dans ce qui suit deux sous-sections. Dans la première, nous déterminerons les paramètres initiaux via la première méthode décrite précédemment. Nous pourrons dès lors réaliser une première exécution de notre implémentation de l'algorithme EM. Dans la seconde section, nous ferons de même, en considérant cette fois-ci la seconde méthode de détermination des paramètres initiaux.

4.2.1 Première réalisation de l'étude

✂ Commençons par déterminer les paramètres initiaux via une méthode graphique. Cette méthode sera basée sur des heuristiques très approximatives, mais nous permettra d'obtenir des valeurs initiales plutôt correctes.

Pour les moyennes, nous considérerons l'abscisse correspondant au point culminant des pics ; et pour les écarts-types σ , nous tenterons de déterminer la valeur entre la moyenne précédemment déterminée et le point d'inflexion du pic. Après plusieurs essais sur R , nous obtenons le graphique suivant :

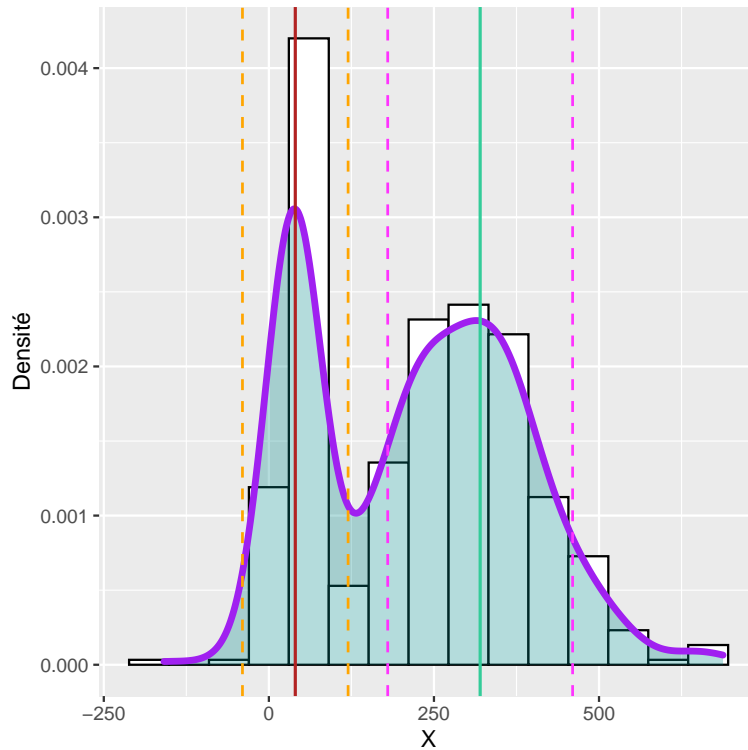


FIGURE 4.3 – Détermination graphique des paramètres initiaux

Nous avons tracé des lignes pleines aux abscisses $X = 40$ et $X = 320$, qui correspondent approximativement au point culminant des pics. Ceci nous donne donc nos moyennes initiales

$$\mu_{1_{init}} = 40 \text{ et } \mu_{2_{init}} = 320.$$

Pour les écarts-types, nous avons tracé des lignes en pointillées aux abscisses $\mu_{1_{init}} \pm 80$ et $\mu_{2_{init}} \pm 140$. Nous prendrons donc comme écarts-types initiaux

$$\sigma_{1_{init}} = 80 \text{ et } \sigma_{2_{init}} = 140$$

Nous n'avons pas représenté les valeurs des abscisses sur le graphique pour des raisons de lisibilité.

Pour les proportions initiales, nous nous proposons de les prendre égales ;

$$\alpha_{1_{init}} = \alpha_{2_{init}} = 0.5.$$

Nous pouvons dès lors construire le dataframe des paramètres initiaux ;

```
param_init_1 = data.frame(bird_names = c("European Goldfinch", "Ring Ouzel"),
                           alpha_init = c(0.5, 0.5),
                           mean_init = c(40, 320),
                           sd_init = c(80, 140))
```

✂ Nous avons ainsi tout les éléments nécessaires à l'exécution de notre algorithme. Comme nous l'avons vu dans le précédent chapitre, une dizaine d'itérations suffisent pour obtenir de bons résultats.

```
algo_EM(param_init_1, data, 10)
```

	bird_names	alpha	mu	sigma
1	European Goldfinch	0.2910832	37.76285	9.512478
2	Ring Ouzel	0.7089168	302.51936	125.951894

Nous obtenons ainsi nos paramètres estimés. Nous pouvons dès lors transmettre les résultats aux ornithologues qui émettront les conclusions appropriées.

✂ Nous avons en notre possession les valeurs théoriques, comparons les avec les valeurs estimées. Ces dernières sont contenues dans le dataframe *df*. Affichons le :

```
bird_names2 proportion_alpha mean_volume sd_volume
1 European Goldfinch      0.2878713      38.0      9.1
12      Ring Ouzel        0.7121287     298.6     125.1
```

Les proportions sont toutes deux très bien estimées, les erreurs sont de l'ordre de 1%. Il en est de même pour les moyennes, les erreurs d'estimations sont de l'ordre de 1%, ce qui est plus que satisfaisant. Les variances sont de même très bien estimées.

4.2.2 Seconde réalisation de l'étude

✂ La deuxième méthode à notre disposition pour déterminer les paramètres initiaux est d'utiliser le jeu de données en tant que tel. Pour les deux moyennes initiales, nous allons utiliser les quantiles ; et plus précisément le premier et le troisième. En travaillant sur *R*, nous pouvons aisément obtenir ces derniers :

```
quantile(data)
```

```
      0%      25%      50%      75%     100%
-44.28714  45.15146 235.38743 352.69611 618.65897
```

Nous obtenons ainsi nos moyennes initiales :

$$\mu_{1_{init}} = 45.15146 \text{ et } \mu_{2_{init}} = 352.69611.$$

Concernant les écarts-types, nous prendrons des valeurs d'initialisations égales ; et nous utiliserons l'écart-type empirique divisé par le nombre de mélanges :

$$\frac{\hat{\sigma}}{J} := \frac{1}{J} \sqrt{\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2}$$

pour déterminer cette valeur. *J* est le nombre de mélanges et *n* la taille de l'échantillon. Ainsi,

$$\sigma_{1_{init}} = \sigma_{2_{init}} = \frac{\hat{\sigma}}{2}$$

Une fois de plus, *R* nous permet de calculer cela aisément :

```
sqrt(var(data))/2
```

```
80.29425
```

Ainsi,

$$\sigma_{1_{init}} = \sigma_{2_{init}} = 80.29425.$$

Pour les proportions initiales, nous proposons de nouveau de les prendre égales, ainsi

$$\alpha_{1_{init}} = \alpha_{2_{init}} = 0.5$$

Par suite, nous construisons le dataframe des paramètres initiaux :

```
param_init_2 = data.frame(bird_names = c("European Goldfinch", "Ring Ouzel"),
                           ,
                           alpha_init = c(0.5, 0.5),
                           mean_init = c(45.15146, 352.69611),
                           sd_init = c(80.29425, 80.29425))
```

✂ Nous exécutons maintenant l'algorithme avec ce deuxième dataframe de paramètres initiaux.

```
algo_EM(param_init_2, data, 10)
```

	bird_names	alpha	mu	sigma
1	European Goldfinch	0.2936202	37.79956	9.690775
2	Ring Ouzel	0.7063798	303.45499	125.198078

✂ Comme pour le cas précédent, comparons les avec les valeurs théoriques. Nous rappelons une fois de plus ces dernières :

	bird_names2	proportion_alpha	mean_volume	sd_volume
1	European Goldfinch	0.2878713	38.0	9.1
12	Ring Ouzel	0.7121287	298.6	125.1

Les proportions sont toutes deux très bien estimées, les erreurs sont de l'ordre de 1%. Il en est de même pour les moyennes, les erreurs d'estimations sont de l'ordre de 1%, , ce qui est formidable. Les variances sont elles aussi remarquablement bien estimées.

Les deux méthodes de déterminations des paramètres initiaux nous donnent des résultats très satisfaisant. Le cas étudié est très favorable, le mélange étant, en quelques sortes, à "fortes séparations".

Nous ferons en annexe une brève étude du package *mclust*, qui contient une implémentation de l'algorithme EM. Nous l'exécuterons sur le même échantillon que celui que nous venons de présenter. Les paramètres estimés par *mclust* seront d'une plus grande précision ; toutefois notre implémentation produit des résultats tout aussi satisfaisant.

Conclusion

Dans le cadre de ce projet, nous nous sommes confrontés à une situation nouvelle pour nous, bien que fréquente dans les problématiques réelles; la non existence d'expressions analytiques des estimateurs du maximum de vraisemblance. Cette difficulté s'est présentée dès le début de l'étude théorique, dès l'instant où nous explorions le problème s'offrant à nous. Ce dernier est issu d'une problématique concrète empruntée à l'ornithologie, celui de l'observation des nids d'oiseaux.

L'impossibilité théorique sus-mentionnée a trouvé solution dans une méthode numérique, l'algorithme EM.

Nous avons choisi d'implémenter cette méthode de nous-même, afin de mieux cerner son fonctionnement, et de ne pas utiliser des « boîtes-noires » de R . Par suite nous avons étudié les performances, en terme de justesse de précisions, de notre implémentation; et ce au travers de simulations. Les résultats obtenus sont satisfaisants; toutefois nous restons critique envers eux, la preuve de convergence de la suite de paramètres produite vers ceux du maximum de vraisemblance n'existant pas.

Ceci nous a permis de proposer dans un dernier chapitre, un modèle pour les nids d'oiseaux; et d'entrevoir ce qu'est une étude de données, dans une situation favorable. En effet, les données étant générées par R , nous n'avons pas eu à nous soucier des problèmes usuels en science de données; à savoir les problèmes concernant un échantillon de taille trop faible ou bien de données aberrantes.

L'algorithme EM est un point d'accroche aux solutions dont nous disposons pour résoudre ce type de problème. Il a été rapidement généralisé, et plusieurs variantes existent; nous pouvons citer l'algorithme Generalized EM, qui propose une amélioration de EM; ou bien le Stochastic Em, qui utilise des méthodes stochastiques afin de diminuer les risques d'obtenir un maximum local.

Bibliographie

- [1] Dempster A.P., Laird N. M., Rubin D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society, Series B*, Vol. 39, 1, 1-38
- [2] Chafai D., Malrieu F. (2018). Recueil de modèles aléatoires, 105-11, *Prépublication*
<https://hal.archives-ouvertes.fr/hal-01897577v3>
- [3] Frédéric Santos (2015). L'algorithme EM : une courte présentation, *Document de cours*
<https://members.loria.fr/moberger/Enseignement/AVR/Exposes/algo-em.pdf>
- [4] Michael Collins (1997). The EM algorithm, *Document de cours*
http://faculty.washington.edu/fxia/courses/LING572/EM_collins97.pdf
- [5] Biddle L.E., Broughton R.E., Goodman A.M., Deeming D.C (2018). Composition of Bird Nests is a Species-Specific Characteristic, *Avian Biology Research*, Vol. 11, 2, 132-153
<https://core.ac.uk/download/pdf/155777956.pdf>
- [6] Fraley C., Raftery A.E., Scrucca L., Murphy T.B, Fop M. (2020). Package 'mclust', *Documentation du package mclust*
<https://cran.r-project.org/web/packages/mclust/mclust.pdf>

Annexes

Annexe A

Le package *mclust*

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. Nous reprendrons ici les mêmes espèces étudiées lors du dernier chapitre ; les divers paramètres seront donc conservés, de même que l'échantillon généré. Nous nous sommes appuyés sur [6] afin d'obtenir les éléments nécessaires à l'utilisation de ce package.

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

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

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 deux mélanges ; un mélange à deux lois et un autre à trois lois. Nous reprenons les mêmes proportions que lors de chapitre 4 pour le mélange à deux lois.

```
df_2= data.frame(bird_names = c("European Goldfinch", "Ring Ouzel"),
                 ,proportion_alpha = c(0.2878713, 0.7121287), 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))
```

Nous reprenons ici notre propre fonction de simulation

```
simulation = function(data_th, n=100)
```

Les prérequis étant posés, nous simulons un échantillon $X2$ de deux espèces d'oiseaux et un autre $X3$ de trois espèces d'oiseaux :

```
set.seed(1907)
X2 <- data # Ce dataframe est celui créé au chapitre 4.
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 ne permet pas de régler manuellement des valeurs initiales pour les paramètres à estimer.

A.1 Un exemple sur un mélange à deux lois

Commençons par exécuter la fonction *densityMclust* sur notre exemple de mélange à deux lois, contenu dans le dataframe $X2$:

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

La première sortie affichée est celle du graphe de la densité du mélange de lois.

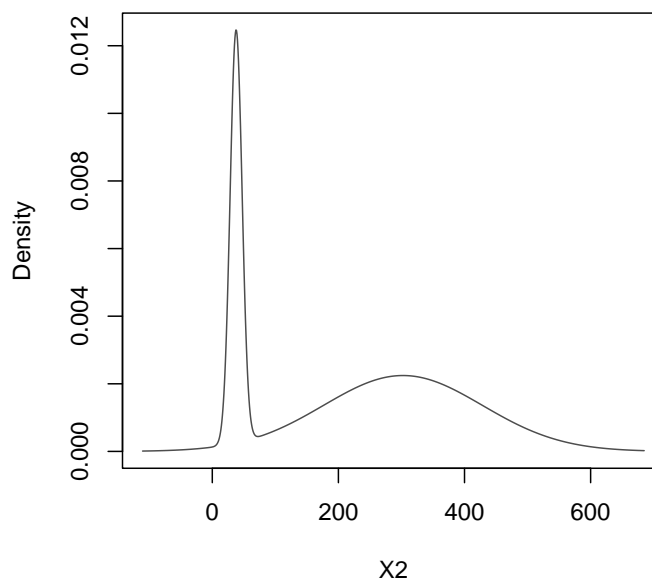


FIGURE A.1 – Densité du mélange à deux lois

Nous pouvons nettement distinguer les deux "pics", correspondant aux deux gaussiennes mélangées. Intéressons-nous maintenant à l'objet créé *est_2*.

```
est_2
```

```
'densityMclust' model object: (V,2)
```

Available components:

[1]	"call"	"data"	"modelName"	"n"	"d"
[6]	"G"	"BIC"	"loglik"	"df"	"bic"
[11]	"ic1"	"hypvol"	"parameters"	"z"	"classification"
[16]	"uncertainty"	"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 : 0.2878713 et 0.7121287 en proportions ; 38 et 298.6 pour les moyennes ; et 9.1 et 125.1 en écart-types.

```
print("Proportions estimées:")
est_2[13]$parameters$pro
print("Moyennes estimées:")
est_2[13]$parameters$mean
print("Ecart-types estimés:")
(est_2[13]$parameters$variance$sigma_sq)^(1/2)
```

```
[1] "Proportions estimées:"
[1] 0.2906965 0.7093035
[1] "Moyennes estimées:"
   1      2
[1] 37.75991 302.37625
[1] "Ecart-types estimés:"
[1]  9.48752 126.06755
```

Ici, le nombre de mélange est exact. Les proportions sont très bien estimées, l'erreur la plus importante est de l'ordre de 1%. Il en va de même pour les moyennes, les erreurs sont d'ordres inférieures à 3%. Les erreurs sur les variances sont de l'ordre de 1%, ce qui est également très bon.

A.2 Un exemple sur un mélange à trois lois

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)
```

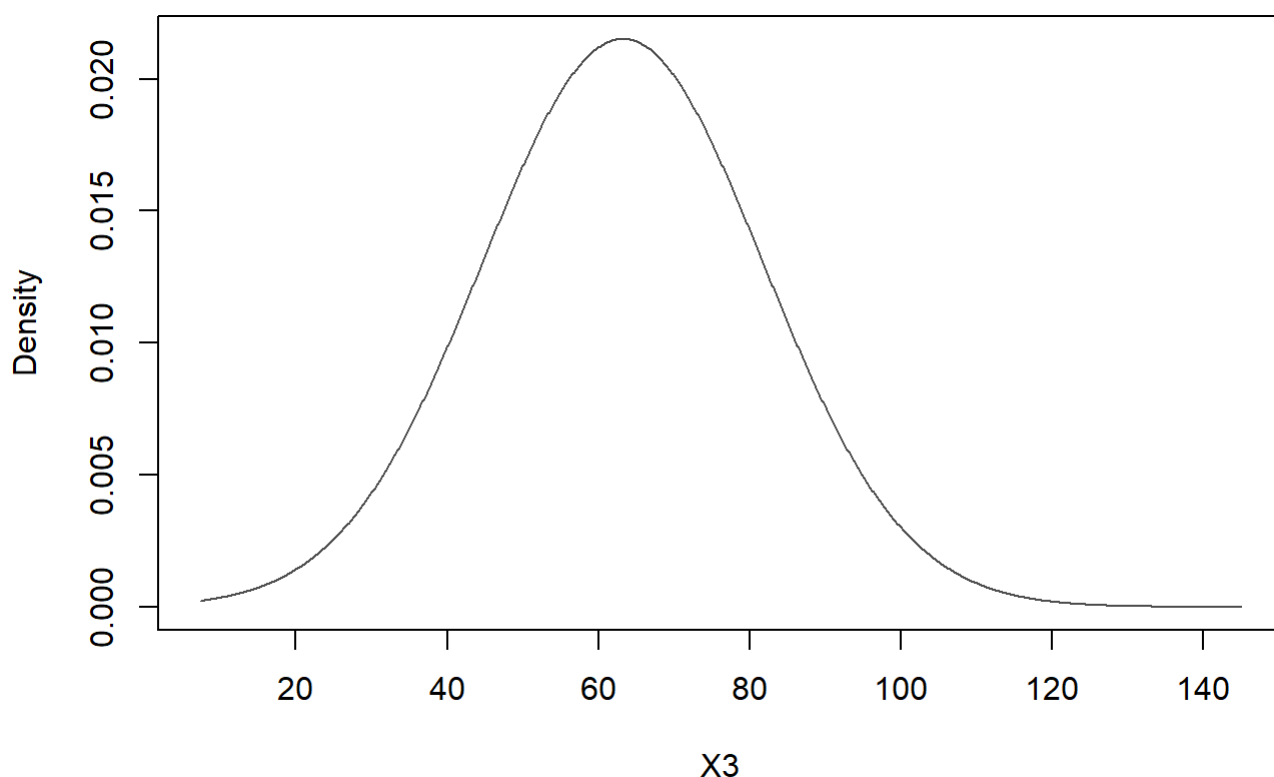


FIGURE A.2 – Densité du mélange à trois lois

Nous obtenons ici quelque chose d'intéressant ; 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...

Il est ici pertinent d'observer la structure des données ; plus précisément, nous allons effectuer un test de *Shapiro*.

```
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 (\mathcal{H}_0) de normalité. Nous sommes ici dans une situation ambiguë.

Observons maintenant comment *densityMclust* se défend face à cette situation.

Rappelons que les divers paramètres de ce mélange sont : 0.6, 0.3 et 0.1 en proportions ; 60.9, 58.3 et 71.6 en moyenne ; et 20.8, 15 et 12.9 en écart-types.

```

print("Proportions estimées:")
est_3[13]$parameters$pro
print("Moyennes estimées:")
est_3[13]$parameters$mean
print("Ecart-types estimés:")
(est_3[13]$parameters$variance$sigmasq)^(1/2)

```

```

[1] "Proportions estimées:"
[1] 1
[1] "Moyennes estimées:"
[1] 63.20547
[1] "Ecart-types estimés:"
[1] 18.54033

```

Le premier élément notable est que l'algorithme échoue à établir le nombre correct de lois. L'unique moyenne et écart-type estimés ne sont quant à eux pas absurdes.

Nous allons relancer la fonction sur le même jeu de données, en précisant cette fois-ci le nombre de lois.

```

est_3b <- densityMclust(X3, G = 3)
print("Proportions estimées:")
est_3b[13]$parameters$pro
print("Moyennes estimées:")
est_3b[13]$parameters$mean
print("Ecart-types estimés:")
(est_3b[13]$parameters$variance$sigmasq)^(1/2)

```

```

[1] "Proportions estimées:"
[1] 0.2552686 0.5642272 0.1805042
[1] "Moyennes estimées:"
      1      2      3
56.63179 62.36268 75.13635
[1] "Ecart-types estimés:"
[1] 17.51051

```

Les proportions sont plutôt bien estimées, quoique légèrement surestimées pour deux d'entre elles, mais les erreurs restent faibles. Il en est étonnant de même pour les moyennes, qui sont très bien estimées. Les erreurs sont au plus de l'ordre de 5%. Ceci est surprenant au vue de l'allure de la densité. Cependant, il n'est estimé qu'un unique écart-type, ce qui n'est guère étonnant. Notons que celui-ci est à peu près égale à la moyenne des écart-types des différentes lois.

Ce cas ambiguë met en exergue les limites de l'algorithme implémenté dans ce package.

Annexe B

Etudes des fonctions

Revenons aux choix des paramètres initiaux. Nous disposons désormais de trois fonctions permettant chacune de déterminer un choix quant à l'initialisation de l'algorithme EM.

Il convient alors d'étudier les résultats de chacune d'elles, en appliquant l'algorithme EM sur les initialisations qu'elles proposent et en comparant la valeur de la log-vraisemblance de l'échantillon (X_1, \dots, X_n) .

Nous présentons alors les deux fonctions suivantes :

Celle-ci permettent de déterminer la valeur de la log-vraisemblance de l'échantillon.

```
log_Vrais_X <- function(data_param,X){  
  J <- length(data_param[,1])  
  logVrai_X <- 0  
  for(i in 1:length(X)){  
    sum_j <- 0  
    for(j in 1:J){  
      sum_j <- sum_j + data_param[j,2]*dnorm(X[i],mean = data_param[j,3],  
      sd = data_param[j,4])  
    }  
    logVrai_X <- logVrai_X + log(sum_j)  
  }  
  return(logVrai_X)  
}
```

Elle prend comme arguments :

- `data_param`, le tableau des résultats obtenus, le nom des espèces étant ajouté en première colonne
- `X`, l'échantillon observé

Elle retourne la valeur de la log-vraisemblance de (X_1, \dots, X_n) pour les paramètres issus de `data_param`.

Nous avons également la fonction *param_init*, effectuant le choix des paramètres en comparant les valeurs des log-vraisemblance obtenues :

```
param_init <- function(data,X){  
  J <- length(data[,1])  
  col <- length(data[1,])-1  
  res <- -Inf  
  data_res <- data.frame()  
  for(i in seq(1,col,by = 3)){  
    data_i <- data.frame(espèces = data[,1], alpha = data[,i+1],  
      moyennes = data[,i+2], sd = data[,i+3])  
    log_Vrai <- log_Vrais_data_i,X)  
    if (res <= log_Vrai){  
      res <- log_Vrai  
      data_res <- data_i  
    }  
  }  
  return(data_res)  
}
```

Elle prend comme arguments :

- data, le tableau de tous les résultats obtenus à partir des fonctions d'initialisation, le nom des espèces étant ajouté en première colonne
- X, l'échantillon

Elle retourne le tableau d'initialisation ayant permis d'obtenir le meilleur résultat.