Olivier CÔME
Nicolas PRALON
N° étudiant Olivier CÔME: 22110708
N° étudiant Nicolas PRALON: 22110681
olivier.come@etu.umontpellier.fr
nicolas.pralon@etu.umontpellier.fr
Master 2
Statistique et Science des Données
Université de Montpellier

Projet

HAX006X : Modèles à variables latentes
Rédigé le 23 mars 2023 en LaTeX

Sommaire

1 Introduction **2**

2 L'algorithme EM **3**
2.1 Implémentation de la fonction simulation 3
2.2 Implémentation de la fonction EM 3
2.3 Étude sur des données simulées 5
2.3.1 Simulation d'un mélange de deux gaussiennes 5
2.3.2 Simulation d'un mélange à quatre gaussiennes 6

3 Comparaison de la fonction *EM* avec *mixtools* sur des données réelles **7**

4 Conclusion **8**

5 Annexes **9**

6 Bibliographie **10**

1 Introduction

Dans le cadre de ce projet, nous allons modéliser le comportement extrême des vagues dans le golfe du lion à l'aide des méthodes vue à travers l'unité d'enseignement HAX005X "valeurs extrêmes". D'après la source [1], le golf s'étale sur 220 kilomètres de la Camargue à la frontière espagnole. La côte, essentiellement sableuse, a été façonnée par la houle (la mer gagnant souvent les terres par élévation du niveau marin) et l'érosion côtière. L'apport de sédiments en provenance des fleuves a également permis de faire avancer le rivage pendant de longues périodes. Des formations de lagunes "comme les graus" (parfois temporaires) ont pu apparaitre et ont permis de faire communiquer les étangs littoraux avec la mer. Le golf du Lion est donc un milieu naturellement dynamique. C'est dans ce contexte que nous étudions le comportement extrême des vagues à cet endroit, de façon univariée dans un premier temps puis de façon bivariée dans un second temps.

2 L'algorithme EM

Dans cette section nous allons nous intéresser à l'algorithme EM (Expectation-Maximization) afin d'estimer les paramètres μ et σ d'un mélange gaussien. Nous avons implémenté cet algorithme via la fonction *EM* (présente dans notre script *R*) en nous aidant de la source [2]. Afin de tester l'efficacité de notre implémentation, nous allons dans un premier temps l'éxecuter sur des données simulées. En effet, dans notre script, nous avons implémenté une autre fonction que nous avons nommé *simulation*. Cette dernière nous permettra de générer de manière aléatoire, un échantillon issu d'un mélange gaussien. Nous décrirons plus en détaille cette fonction, en aval. Dans un second temps, nous exécuterons notre algorithme sur de vraies données afin de voir s'il est robuste. Pour cela nous utiliserons le jeu de données galaxies de la librairie MASS et nous le décrirons ultérieurement.

2.1 Implémentation de la fonction simulation

Comme il a été mentionné précédemment nous allons réaliser dans un premier temps une étude sur des données simulées à partir de la fonction *simulation*. Décrivons cette dernière, elle prend en argument :

- **dt_param :** Le dataframe contenant les paramètres α, μ et σ
- **n :** La taille de l'échantillon

Elle retourne un vecteur de taille n qui sera l'échantillon du mélange gaussien.
Regardons un plus en détaille comment a été conçue cette fonction.
La partie la plus importante et la plus subtile de ce script est celle dans laquelle nous distribuons aléatoirement les $(X_i)_{i\in 1,\dots,n}$ de l'échantillon de sorte à avoir un bon mélange gaussien.
Afin de simplifier les choses, rien de mieux que de prendre un exemple. Dans celui-ci, l'objectif sera de générer un mélange de quatre gaussiennes, ayant pour paramètres respectifs $\theta_1 = (\alpha_1, \mu_1, \sigma_1)$, $\theta_2 = (\alpha_2, \mu_2, \sigma_2)$, $\theta_3 = (\alpha_3, \mu_3, \sigma_3)$ et $\theta_4 = (\alpha_4, \mu_4, \sigma_4)$.
Les $(\alpha_j)_{j\in\{1,\dots,4\}}$ étant ici des probabilités, nous avons que :

$$\sum_{j=1}^{4} \alpha_j = 1$$

La démarche est la suivante :

- Si $Z < \alpha_1$ alors $X \sim \mathcal{N}(\mu_1, \sigma_1)$
- Sinon si $\alpha_1 < Z < \alpha_1 + \alpha_2$ alors $X \sim \mathcal{N}(\mu_2, \sigma_2)$
- Sinon si $\alpha_1 + \alpha_2 < Z < \alpha_1 + \alpha_2 + \alpha_3$ alors $X \sim \mathcal{N}(\mu_3, \sigma_3)$
- Sinon si $\alpha_1 + \alpha_2 + \alpha_3 < Z < \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ alors $X \sim \mathcal{N}(\mu_4, \sigma_4)$

Notez que notre implémentation marche dans le cas général d'un mélange de J gaussiennes avec $J \in \mathbb{N}^*$.

2.2 Implémentation de la fonction EM

La fonction *EM* est sans aucun doute celle la plus importante de cette section, il est donc primordial de la décrire.
Tout d'abord, pour implémenter cette dernière, nous nous sommes fortement aidés du pseudo-code suivant :

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

Entrée(s) : $N \in \mathbb{N}$, $\widehat{\theta_0} \in \Theta$, un jeu de données $x_1 \dots x_n$;
Initialisation ;
$k := 1$;
Tant que $K < N + 1$ **faire**
  ETAPE E : *Calculer la fonction* $Q(\theta; \widehat{\theta}_{k-1}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\widehat{\theta}_{k-1}}[logf(X_i, Z_i, \theta)|X_i = x_i]$;
  ETAPE M : $\widehat{\theta}_k = argmax\ Q(\theta; \widehat{\theta}_{k-1})$;
  $k \leftarrow k + 1$;
fin du Tant que ;
retourner $\widehat{\theta}_N$;

La fonction *EM* prend en argument :
- data_init, le dataframe contenant les paramètres initiaux (α_{init}, μ_{init}, σ_{init})
- X, les données (réelles ou simulées) issues d'un mélange gaussien
- K le nombre d'itérations souhaitées pour l'algorithme

Elle retourne un dataframe contenant les valeurs des paramètres estimées par l'algorithme, à savoir α, μ et σ.
Les formules que nous avons utilisé pour calculer l'étape E et M et qui sont présentées ci dessous sont issue de la source [2].

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

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

- 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 suivantes :

$$\widehat{\alpha_j} = \frac{1}{n} \sum_{i=1}^{n} (Z = j|X = X_i)$$

$$\widehat{\mu_j} = \frac{\sum_{i=1}^{n} X_i (Z = j|X = X_i)}{\sum_{i=1}^{n} (Z = j|X = X_i)}$$

$$\widehat{v_j} = \frac{\sum_{i=1}^{n} (X_i - \widehat{\mu_j})^2 (Z = j|X = X_i)}{\sum_{i=1}^{n} (Z = j|X = X_i)}$$

Toutefois, il est important de notifier le fait qu'il n'existe pas de convergence de la suite de paramètres établies par l'algorithme EM. En effet, ces derniers peuvent rester bloqués dans des extremas locaux. On comprend donc qu'il est primordial de choisir de bons paramètres initiaux afin de ne pas être confronté à ce problème. Dans la section consacrée à l'étude sur de vraies données, nous expliciterons la procédure qui a été mise en place pour choisir ces paramètres initiaux.

2.3 Étude sur des données simulées

Cette sous-section sera consacrée à l'étude menée sur des données simulées à partir de notre fonction *simulation*.

2.3.1 Simulation d'un mélange de deux gaussiennes

Nous avons ici décidé de générer un échantillon de taille 100 issu d'un mélange de deux gaussiennes de lois respectives $\mathcal{N}(\mu_1, \sigma_1) = \mathcal{N}(50, 11)$ et $\mathcal{N}(\mu_2, \sigma_2) = \mathcal{N}(220, 50)$. La densité associée à cet échantillon a été estimée de manière non paramétrique à partir d'une méthode à noyau et elle a été tracée sur la figure 1.

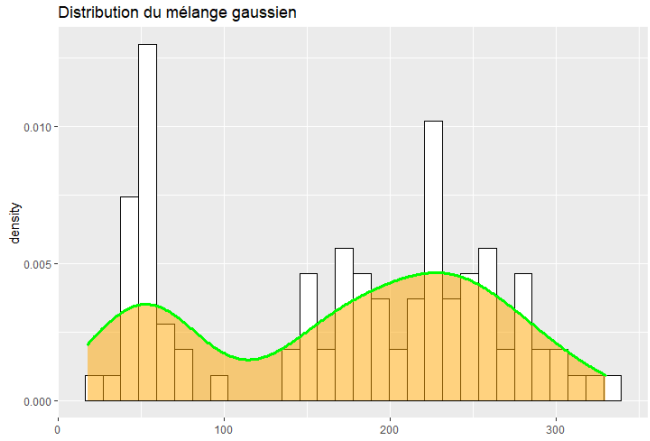

FIGURE 1 – Densité estimée par méthode à noyau.

Nous avons ensuite appliqué notre algorithme (fonction EM) sur cet échantillon. Le tableau 1 contient les valeurs des vrais paramètres de ce mélange simulé. Comme il a été mentionné précédemment, le choix des paramètres initiaux est crucial si l'on veut que l'algorithme estime correctement les paramètres du mélange. Le tableau 2 contient les valeurs des paramètres initiaux utilisés. Comme vous pouvez le voir en observant ces deux tableaux, nous avons choisi des paramètre initiaux assez proches des vrais de manière à ne pas être bloqué dans des extremas locaux. Les résultats obtenus par notre algorithme sont affichés sur la capture d'écran de la figure 2.
Comme on peut le voir sur la figure 2, les valeurs estimées par notre implémentation sont proches de celles des vrais paramètres (voir tableau 1). Ces résultats nous montrent donc que notre fonction *EM* fonctionne correctement, ce qui est rassurant.

```
      mixtureParameters     alpha       mu    sigma
1 parameters of Mixture1 0.3998206  48.86914 10.02995
2 parameters of Mixture2 0.6001794 214.56796 41.67484
```

FIGURE 2 – Paramètres du mélange gaussien estimés par notre fonction *EM*

	α	μ	σ
Paramètres du 1er mélange	0.4	50	11
Paramètres du 2ème mélange	0.6	220	50

TABLE 1 – Vrais paramètres du mélange

	α_{init}	μ_{init}	σ_{init}
Paramètres du 1er mélange	0.2	30	21
Paramètres du 2ème mélange	0.8	280	160

TABLE 2 – Paramètres estimées du mélange

2.3.2 Simulation d'un mélange à quatre gaussiennes

Nous avons ici décidé de générer un échantillon de taille 1000 issu d'un mélange de quatre gaussiennes de lois respectives :

— $\mathcal{N}(\mu_1, \sigma_1) = \mathcal{N}(35, 11)$
— $\mathcal{N}(\mu_2, \sigma_2) = \mathcal{N}(350, 22)$
— $\mathcal{N}(\mu_3, \sigma_3) = \mathcal{N}(720, 32)$
— $\mathcal{N}(\mu_4, \sigma_4) = \mathcal{N}(1198, 55)$

La densité associée à cet échantillon a été estimée de manière non paramétrique à partir d'une méthode à noyau et elle a été tracée sur la figure 3.

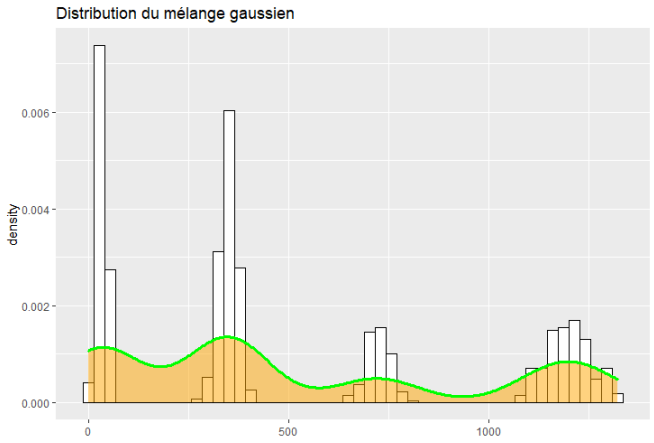

FIGURE 3 – Paramètres initiaux choisis

Nous avons ensuite appliqué notre algorithme (fonction EM) sur cet échantillon. Le tableau 3 contient les valeurs des vrais paramètres de ce mélange simulé. Comme il a été mentionné précédemment, le choix des paramètres initiaux est crucial si l'on veut que l'algorithme estime correctement les paramètres du mélange. Le tableau 4 contient les valeurs des paramètres initiaux utilisés. Comme vous pouvez le voir en observant ces deux tableaux, nous avons choisi des paramètre initiaux assez proches des vrais de manière à ne pas être bloqué dans des extremas locaux. Les résultats obtenus par notre algorithme sont affichés sur la capture d'écran de la figure 4.

Comme on peut le voir sur la figure 4, les valeurs estimées par notre implémentation sont proches de celles des vrais paramètres (voir tableau 3). Ces résultats nous montrent conforte une fois de plus dans l'idée que notre fonction EM fonctionne correctement.

```
> print(paramEst)
      mixtureParameters alpha         mu    sigma
1 parameters of Mixture1 0.284   33.57018 10.95175
2 parameters of Mixture2 0.345  348.98442 21.81027
3 parameters of Mixture3 0.129  722.23235 30.41467
4 parameters of Mixture4 0.242 1200.08647 55.03635
```

FIGURE 4 – Paramètres du mélange gaussien estimés par notre fonction EM

	α	μ	σ
Paramètres du 1er mélange	0.30	35	11
Paramètres du 2ème mélange	0.33	350	22
Paramètres du 3ème mélange	0.15	720	32
Paramètres du 4ème mélange	0.22	1198	55

TABLE 3 – Vrais paramètres du mélange

	α	μ	σ
Paramètres du 1er mélange	0.33	30	10
Paramètres du 2ème mélange	0.30	370	25
Paramètres du 3ème mélange	0.17	717	30
Paramètres du 4ème mélange	0.20	1238	57

TABLE 4 – Paramètres initiaux choisis

3 Comparaison de la fonction *EM* avec *mixtools* sur des données réelles

Dans cette section, l'étude sera menée sur des données réelles. Nous utiliserons le jeu de données *galaxies* provenant de la librairie *MASS* de *R*. Dans un premier temps, nous allons estimer les paramètres du mélange associés à ce dataset via l'utilisation de notre implémentation de l'algorithme EM (la fonction EM). Nous estimerons ensuite une seconde fois les paramètres de ce même mélange mais cette fois-ci, en utilisant la fonction *normalmixEM* prédéfinie de *R* qui est disponible via le package *mixtools*. Nous avons choisi d'utiliser cette librairie car l'algorithme EM y est implémenté et il est utilisé par la fonction *normalmixEM* pour estimer les paramètres d'un mélange. Le fait de comparer les résultats de notre fonction avec ceux obtenus par celle prédéfini de *R* nous permettra d'évaluer la performance et la robustesse de notre implémentation sur de vraies données.
Le jeu de données *Galaxies* est un vecteur numérique qui représente les vitesses en km/s (kilomètres par secondes) de 82 galaxies. La figure 5 est un extrait de ce jeu de données.

```
> galaxies
 [1]  9172  9350  9483  9558  9775 10227 10406 16084 16170 18419 18552 18600 18927
[14] 19052 19070 19330 19343 19349 19440 19473 19529 19541 19547 19663 19846 19856
[27] 19863 19914 19918 19973 19989 20166 20175 20179 20196 20215 20221 20415 20629
[40] 20795 20821 20846 20875 20986 21137 21492 21701 21814 21921 21960 22185 22209
[53] 22242 22249 22314 22374 22495 22746 22747 22888 22914 23206 23241 23263 23484
[66] 23538 23542 23666 23706 23711 24129 24285 24289 24366 24717 24990 25633 26690
[79] 26995 32065 32789 34279
```

FIGURE 5 – Extrait du jeu de données *galaxies*

Comme il a été mentionné dans la section 2.2, si l'on veut obtenir de bonnes estimations pour les paramètres du mélange, il est primordial de sélectionner correctement les paramètres qui serviront de conditions initiales pour l'algorithme. Nous allons donc détailler la stratégie qui a été mise en place pour sélectionner ces derniers. Étant donné que nous ne connaissons pas la vraie densité associée à ces données, nous avons utilisé dans un premier temps une méthode d'estimation non paramétrique (à noyau) afin de d'estimer cette dernière. La figure 6 représente la courbe de densité estimée par la méthode à noyau.
En observant la figure 6, on peut facilement distinguer 3 pics principaux. Un premier vers 9800 sur l'axe des abscisses, un deuxième vers 21000 et un 3ème vers 32000. Il semblerait aussi y avoir une sorte de pic vers 24000 mais celui-ci n'étant pas clairement visible nous ne le considérerons pas. On supposera donc pour la suite de l'étude que nous somme dans le cas d'un mélange à 3 composantes.

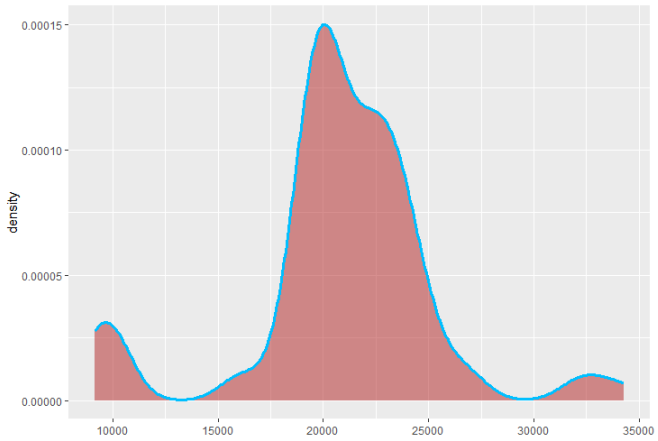

FIGURE 6 – Densité du dataset *galaxies* estimée par une méthode à noyau

4 Conclusion

À travers ce projet, nous avons pu voir qu'au fil des années, nous serons confrontés à des valeurs extrêmes de vagues de plus en plus grandes. Cela est sans aucun doute une conséquence de la hausse des températures et en particulier de la montée du niveau des océans. Nous comprenons donc pourquoi la lutte contre le réchauffement climatique est au cœur des enjeux scientifiques actuels. D'autre part, nous avons pu constater que plus nous prendrons des mesures issues de stations proches et plus les données associées à ces dernières seront fortement corrélées.

5 Annexes

6 Bibliographie

[1] https://reporterre.net/Le-golfe-du-Lion-est-tres-vulnerable-a-la-montee-des-eaux page web rédigée par Alexandre Brun et Benoît Devillers

[2] Frédéric Santos (2015). L'algorithme EM : une courte présentation https://members.loria.fr/moberger/Enseignement/AVR/Exposes/algo-em.pdf