Université de Montréal

Application des méthodes de partitionnement de données fonctionnelles aux trajectoires de voiture

par

Alexandre Paul

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Statistique

31 Août 2020

Université de Montréal

Faculté des études supérieures et postdoctorales

Ce mémoire intitulé

Application des méthodes de partitionnement de données fonctionnelles aux trajectoires de voiture

présenté par

Alexandre Paul

a été évalué par un jury composé des personnes suivantes :

Pierre Duchesne
(président-rapporteur)

Alejandro Murua
(directeur de recherche)

Aurélie Labbe
(codirecteur)

Manuel Morales
(membre du jury)

Résumé

La classification et le regroupement des données fonctionnelles longitudinales ont fait beaucoup de progrès dans les dernières années. Plusieurs méthodes ont été proposées et ont démontré des résultats prometteurs. Pour ce mémoire, on a comparé le comportement des algorithmes de partitionnement sur un ensemble de données décrivant les trajectoires de voitures dans une intersection de Montréal. La motivation est qu'il est coûteux et long de faire la classification manuellement et on démontre dans cet ouvrage qu'il est possible d'obtenir des prédictions adéquates avec les différents algorithmes.

Parmi les méthodes utilisées, la méthode *distclust* utilise l'approche des K-moyennes avec une notion de distance entre les courbes fonctionnelles. On utilise aussi une classification par mélange de densité gaussienne, *mclust*. Ces deux approches n'étant pas conçues uniquement pour le problème de classification fonctionnelle, on a donc également appliqué des méthodes fonctionnelles spécifiques au problème : *fitfclust*, *funmbclust*, *funclust* et *funHDDC*.

On démontre que les résultats du partitionnement et de la prédiction obtenus par ces approches sont comparables à ceux obtenus par ceux basés sur la distance. Les méthodes fonctionnelles sont préférables, car elles permettent d'utiliser des critères de sélection objectifs comme le AIC et le BIC. On peut donc éviter d'utiliser une partition préétablie pour valider la qualité des algorithmes, et ainsi laisser les données parler d'elles-mêmes. Finalement, on obtient des estimations détaillées de la structure fonctionnelle des courbes, comme sur l'impact de la réduction de données avec une analyse en composantes principales fonctionnelles multivariées.

Mot clés : Donnée fonctionnelle, partition, B-splines, analyse en composantes principales, modèles probabilistes, Bayes, courbes de trajectoire.

Abstract

The study of the clustering of functional data has made a lot of progress in the last couple of years. Multiple methods have been proposed and the respective analysis has shown their efficiency with some benchmark studies. The objective of this Master's thesis is to compare those clustering algorithms with datasets from traffic at an intersection of Montreal. The idea behind this is that the manual classification of these data sets is time-consuming. We show that it is possible to obtain adequate clustering and prediction results with several algorithms.

One of the methods that we discussed is *distclust* : a distance-based algorithm that uses a K-means approach. We will also use a Gaussian mixture density clustering method known as *mclust*. Although those two techniques are quite effective, they are multi-purpose clustering methods, therefore not tailored to the functional case. With that in mind, we apply four functional clustering methods : *fitfclust*, *funmbclust*, *funclust*, and *funHDDC*.

Our results show that there is no loss in the quality of the clustering between the aforementioned functional methods and the multi-purpose ones. We prefer to use the functional ones because they provide a detailed estimation of the functional structure of the trajectory curves. One notable detail is the impact of a dimension reduction done with multivariate functional principal components analysis. Furthermore, we can use objective selection criteria such as the AIC and the BIC, and avoid using cluster quality indices that use a pre-existing classification of the data.

Keywords: Functional data, clustering, B-splines, principal component analysis, probabilistic models, Bayes, trajectory curves.

Table des matières

Résumé 5

Abstract 7

Liste des tableaux 11

Liste des figures 13

Remerciements 15

Chapitre 1. Introduction 17

1.1. Présentation de la problématique 18

Chapitre 2. Analyse des données fonctionnelles 21

2.1. Définition des données fonctionnelles 21

2.1.1. L'espace fonctionnel 21

2.1.2. Quelques exemples 23

2.2. Expansions par base de fonctions 24

2.2.1. Bases par splines 25

2.2.2. B-splines 27

2.3. Analyse en composante principale 29

2.3.1. Analyse en composante principale 30

2.3.2. Analyse en composante principale fonctionnelle 31

2.3.3. Simplification de l'ACPF 35

Chapitre 3. Méthodes de partitionnement 37

3.1. Définition du partitionnement 37

3.2. Méthodes basées sur la distance 38

3.2.1. K plus-proche-voisins 38

3.2.2. K-Moyennes 40

3.2.3. K-moyennes par noyaux 40

3.3. Méthodes probabilistes ... 42
3.3.1. Estimation de densités par mélanges de densités ... 42
3.3.1.1. Mélanges gaussiens ... 44
3.3.2. Modèles bayésiens de mélange gaussien ... 44
3.3.3. Estimation des paramètres : l'algorithme EM ... 45
3.3.4. Sélection d'un modèle : les critères d'information bayésien et d'Akaike ... 47

Chapitre 4. Partitionnement des données fonctionnelles ... 49

4.1. Méthode basée sur des distances ... 49
4.1.1. distclust ... 49

4.2. Méthodes générales basées sur des mélanges de densités ... 52
4.2.1. mclust ... 52

4.3. Méthodes fonctionnelles ... 55
4.3.1. fitfclust ... 56
4.3.2. funmbclust ... 58
4.3.3. funclust ... 61
4.3.4. funHDDC ... 64

Chapitre 5. Simulation : résultats du partitionnement ... 69

5.1. Préparation des données ... 69
5.1.1. Interpolation LOESS ... 70
5.1.2. Construction des échantillons ... 71
5.1.3. Indice ajusté de Rand ... 72

5.2. Choix de l'échantillon et des paramètres ... 74

5.3. Analyse des résultats ... 74
5.3.1. Résultats de l'analyse en composante principale fonctionnelle ... 74
5.3.2. Résultats des partitionnements ... 76

5.4. Prédictions ... 83

Chapitre 6. Conclusion ... 87

Références bibliographiques ... 89

Annexe A. Annexe ... 91

A.1. Identité de Parseval ... 91

Liste des tableaux

4.1 Liste des différentes paramétrisations de la matrice de covariance 54

5.1 Résultats du partitionnement avec l'algorithme *distclust*. 77
5.2 Résultats du partitionnement avec l'algorithme *mclust*. 79
5.3 Résultats du partitionnement avec l'algorithme *fitfclust*. 80
5.4 Résultats du partitionnement avec l'algorithme *funmbclust*. 80
5.5 Résultats du partitionnement avec l'algorithme *funclust* 82
5.6 Résultats du partitionnement avec l'algorithme *funHDDC* 84
5.7 Résultats de la prédiction selon le résultat avec le meilleur ARI, BIC et AIC pour chaque méthode. 85

Liste des figures

1.1 Interprésentation graphique des trajectoires de voitures 20

2.1 Grandeur moyenne (en cm) des enfants entre 1 et 18 ans et Température moyenne (en °C) par jour dans les différentes villes canadiennes pour une année 23

2.2 Impact du nombre q de fonctions de base a sur le lissage des données. 26

2.3 Impact de la continuité sur une régression cubique locale, sur un échantillon cyclique 28

2.4 Graphique des fonctions de base unidimensionnelles d'un B-splines d'ordre 3 sur l'intervalle [0,1] séparé en 10 sous-intervalles. 29

2.5 Composantes principales d'un échantillon normal bivarié. 31

2.6 Analyse en composante principale fonctionnelle sur la base de données *Weather*. . 34

3.1 Exemple de la classification K-plus-proches-voisins. 39

3.2 Partitionnement avec l'algorithme des K-moyennes avec et sans noyau gaussien . 41

3.3 Densité obtenue par le mélange de trois distributions gaussiennes 45

5.1 Les moyennes des sommes des erreurs aux carrés selon le nombre de splines utilisés 75

5.2 Analyse en composantes principales sur les coordonnées 76

5.3 Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *distclust*. 77

5.4 Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *mclust*. 78

5.5 Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *fitfclust*. 81

5.6 Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *funmbclust*. 82

5.7 Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *funclust*. 83

5.8 Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *funHDDC*. 83

Remerciements

Je voudrais d'abord remercier mon directeur Alejandro Murua et ma co-directrice Aurélie Labbe pour leur soutien et pour avoir cru en moi à travers le parcours de mon mémoire. Malgré les difficultés rencontrées, ils ont continué à me supporter et m'offrir l'aide dont j'avais besoin pour compléter ce chapitre de mon parcours académique.

Je dois remercier tous mes collègues de l'Université de Montréal, notamment Gabriel Boisvert pour sa relecture et ses commentaires, qui m'ont été d'une aide incroyable, mon collègue de bureau Gabriel Lemyre, pour avoir toujours été à mes côtés dans les moments difficiles, et aussi un grand remerciement à tous les amis que j'ai rencontrés durant mon parcours, dont Thomas, Pierre-Alexandre, Rosalie, Camille, Jonathan et Alexis. Ensuite, je veux saluer le support apporté par plusieurs personnes lors de mon parcours. François et Lory-Ann pour la superbe colocation que nous avons eue ensemble. À Antoine, Fabrice, Justin, Raphaël, Julie et Charles pour être des personnes exceptionnelles avec qui j'ai eu la chance de faire mon parcours universitaire avec. Puis à Julie et Philippe, deux personnes si importantes dans ma vie et qui m'ont aidé à passer au travers du monde étrange des mathématiques et de la statistique. J'aimerais souligner le travail de Léo qui a eu le courage de corriger mes fautes de français.

Enfin, je veux remercier ma famille. D'abord mes parents, Monique et Michel, qui ont toujours été à l'écoute dans les moments les plus sombres. Mes sœurs Élizabeth et Catherine pour avoir été formidables et avoir cru en moi et enfin mon frère Nichollas qui, je l'espère, sait qu'il aura toujours une place spéciale dans mon cœur.

Je voudrais remercier une dernière personne, sans qui tout ça n'aurait pas été possible. Monsieur Manuel Prégent, c'est grâce à votre aide professionnelle que j'ai pu devenir la personne que je suis aujourd'hui. Vous m'avez fait comprendre l'importance de prendre soin de ma santé mentale, de faire la paix avec mon passé et de ne pas lâcher prise. Avec toute sincérité, merci.

Chapitre 1

Introduction

L'analyse des données fonctionnelles s'intéresse à comprendre le comportement d'une quantité expliquée par une fonction dépendante d'une variable continue. Dans le cas unidimensionnel, ces fonctions sont représentées par des courbes, comme l'évolution de la température au fil des jours. Ce type de données se distingue surtout par leur caractère longitudinal, défini par la variation d'un phénomène dans le temps. C'est un domaine de la statistique qui a évolué dans ces dernières années [**13**], dans lequel de nombreux chercheurs se sont intéressés à la modélisation des données fonctionnelles, à leur prédiction et leur partitionnement.

Cet ouvrage s'intéressera principalement au partitionnement des données et à la comparaison des multiples méthodes existantes. L'objectif de l'analyse des partitions est d'identifier les regroupements homogènes des observations fonctionnelles issues d'une variable aléatoire. Supposons que les grandeurs de plusieurs enfants ont été collectées, mais que la variable de sexe n'est pas disponible même s'il est connu qu'il y a une différence entre les garçons et les filles. Une analyse des partitions permettrait de déceler cette différence et de classifier les courbes de grandeur par le sexe de l'individu. Cet outil est souvent utilisé lors de la pré-analyse et de l'exploration des données, pour ainsi voir si les observations peuvent être partitionnées en groupes homogènes.

Plusieurs chercheurs tels que Jacques, James, Preda et Sugar ([**15**], [**14**]) ont tenté de déterminer la meilleure approche pour résoudre le problème de partitionnement et, si possible, de modélisation. Certains ont décidé d'utiliser une notion de distance fonctionnelle pour appliquer la méthode des «k-moyennes», populaire dans le monde de l'apprentissage machine [**20**], et ont créé l'approche *distclust*. D'autres ont montré qu'une méthode par mélange gaussien est suffisante, avec la méthode *mclust* [**10**]. Cependant, ces deux techniques ne modélisent pas les courbes d'un point de vue fonctionnel et il est donc souhaitable d'aller chercher plus d'information. C'est pour cela que de nombreux écrits

proposent des méthodes qui vont non seulement modéliser les données fonctionnelles, mais aussi réduire leur dimension et partitionner les courbes en groupes homogènes. Les modèles explorés dans ce mémoire seront *fitfclust* [**15**], *funmbclust* [**1**], *funclust* [**14**] et *funHDDC* [**3**]. À l'exception de la méthode par distance, ces méthodes possèdent des paramètres qui doivent être estimés. Ces estimations seront calculées usuellement par l'algorithme EM [**9**].

Ainsi, on souhaite appliquer ces méthodes dans le domaine du transport routier et déterminer la plus appropriée. La compagnie de développement de technologies de l'information *Brisk Synergie* nous a fourni un ensemble de données qui décrivent les déplacements des voitures au travers d'une intersection à trois branches.

Ce qui nous amène aux deux questions suivantes : est-ce que les méthodes de partitionnement fonctionnelles s'appliquent efficacement sur les trajectoires de véhicules automobiles? Avec les paramètres obtenus du partitionnement, est-ce que ces modèles peuvent bien classifier un nouvel ensemble de trajectoires? C'est à ces questions que l'on souhaite répondre dans cet ouvrage.

On présentera le problème en plus de détails, ainsi que la structure des données dans la sous-section suivante. Comme n'importe quelle analyse statistique, le nettoyage et la pré-analyse des données sont une étape primordiale, puisqu'elle permette de mieux saisir la nature de l'information disponible. Une définition formelle des données fonctionnelles sera présentée dans le chapitre 2, ainsi qu'une explication des outils théoriques. Ces outils sont l'analyse en composantes principales et les modèles fonctionnels paramétriques, qui vont permettre la réduction de dimension et la reconstruction des fonctions respectivement.

Le chapitre 3 présentera une vision globale du partitionnement de n'importe quel type de données, fonctionnel ou non. Ceci permettra d'expliquer clairement les concepts comme la méthode des k-moyennes et l'algorithme EM. Ce dernier est utile lors de l'estimation de paramètres. Par la suite, le cœur de la rédaction se trouvera dans le chapitre 4, où les méthodes de partitionnement fonctionnel seront détaillées. Un total de six méthodes seront explorées : quatre fonctionnelles, une basée sur la notion de distance et une autre sur le mélange de densité. Par la suite, le chapitre 5 portera sur les résultats obtenus lors des évaluations numériques. Dans un premier temps, les résultats du partitionnement brut seront présentés, ceux de la classification automatique seront à leur tour présentés.

1.1. Présentation de la problématique

Brisk Synergie, maintenant connu sous le nom de *Transoft Solutions*, est une entreprise qui se spécialise notamment dans l'ingénierie routière. Ils ont développé des capteurs

vidéo permettant de collecter les données des trajectoires de véhicules. Ils l'utilisent principalement pour faire de la prédiction de collision, c'est-à-dire déterminer les points dans une intersection où les risques de collision sont plus grands. Ces capteurs sont aussi capables d'identifier le type de véhicule (voitures, autobus, cyclistes et même les piétons). Pour notre cas, on a les données d'une caméra installée à une intersection à trois branches. Du point de vue de la caméra, elle a la forme d'un «T»à l'envers.

Avec la vidéo prise par la caméra, leurs logiciels calculent la position d'un véhicule en coordonnées cartésiennes (x,y) à chaque fréquence (la caméra prend 60 images par seconde). Ensuite, on a procédé à un nettoyage des données. Les trajectoires des voitures ont été préservées, puisqu'elles sont le centre de notre étude. Ensuite, la variable de temps a été standardisée puisque les voitures traversent l'intersection à différente vitesse. La variable de temps brute est selon l'apparition du véhicule dans la vidéo. On a ajusté cette variable sur l'intervalle [0,1] tel que 0 est le moment où la voiture entre dans le cadre du capteur vidéo, et 1 celui où elle en sort.

On a donc des trajectoires définies par les coordonnées $(x(t), y(t))$ selon la variable de temps t. Avec cette interprétation mathématique, on souhaite analyser les données et les regrouper selon leur direction. Cette direction est définie par la branche d'entrée et de sortie que la voiture utilise dans l'intersection. Puisqu'on a trois branches, il y a donc six groupes possibles. Une présentation de l'interprétation mathématique des trajectoires se retrouve dans 1.1. Il faut comprendre qu'il y a une déformation entre l'information prise par la caméra et la réalité, mais les trajectoires restent valides. Avant de regrouper les données des trajectoires, il faut comprendre l'espace fonctionnel dans lequel elles résident.

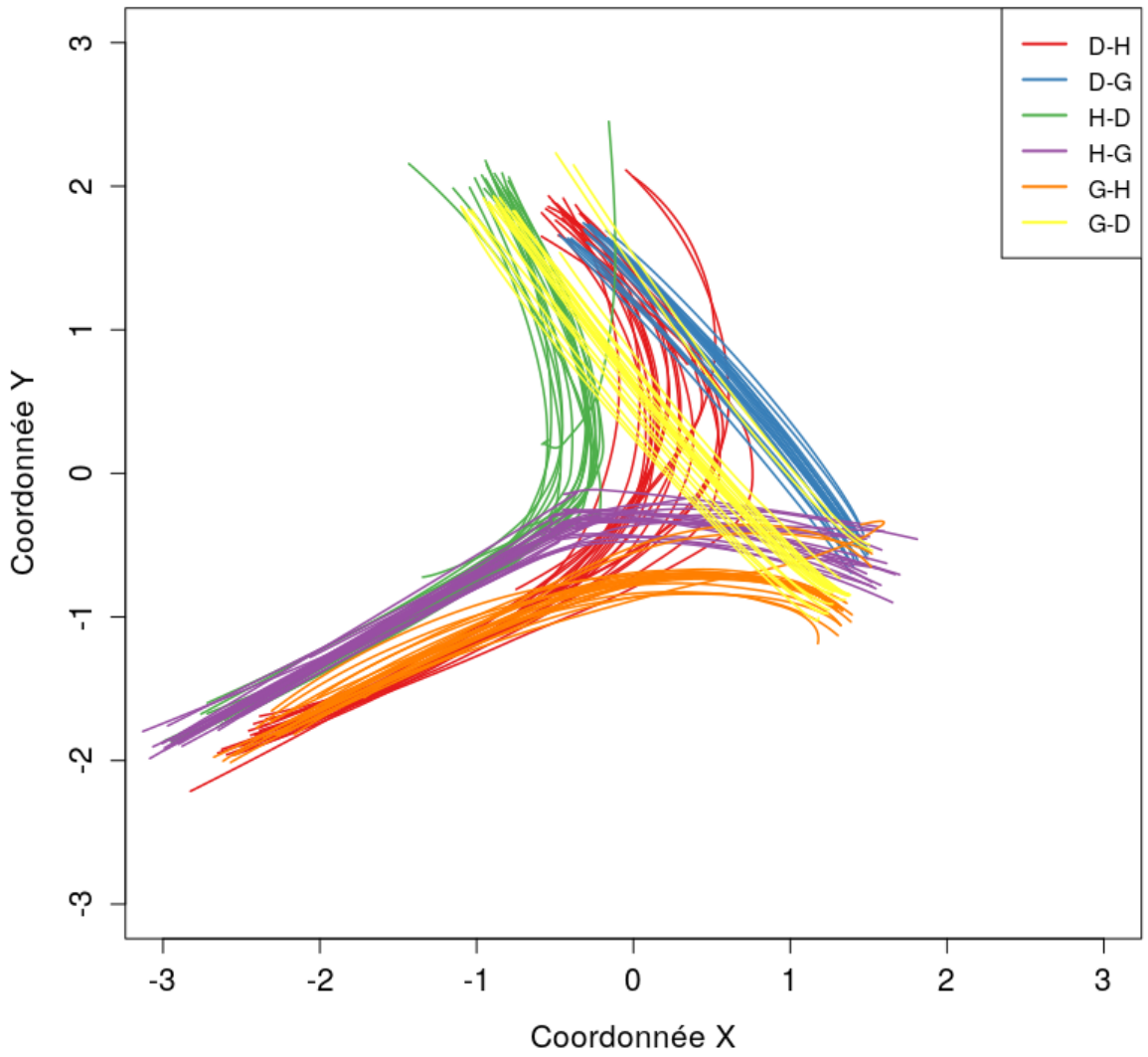

Fig. 1.1. Interprésentation graphique des trajectoires de voitures. Selon le point de vue de la caméra, les branches sont notées par G (Gauche), D (Droite) et H (Haut).

Chapitre 2

Analyse des données fonctionnelles

Il faut d'abord comprendre le comportement de l'espace fonctionnel. Les modèles statistiques multivariés s'appliquent difficilement dans ce contexte. L'espace fonctionnel est de dimension infinie et on traite des données longitudinales, c'est-à-dire une évolution au travers du temps. C'est pour cela on établira dans ce chapitre le contexte de l'analyse fonctionnelle, des problèmes confrontés et des solutions existantes.

La première partie est consacrée à la définition théorique des données fonctionnelles et de l'espace dans lequel elles résident. Des opérations mathématiques et des calculs statistiques seront définis. Par la suite, des techniques de reconstruction des données par des modèles paramétriques fonctionnels seront proposées, après quoi l'analyse en composantes principales permettra de réduire la dimension de l'espace fonctionnel. L'ensemble de ce chapitre suit la théorie établie par Ramsay et Silverman dans [**24**] et de Friedman et Tibshirani [**11**].

2.1. Définition des données fonctionnelles

2.1.1. L'espace fonctionnel

Comme décrit dans l'introduction, les données fonctionnelles sont des valeurs quantitatives qui évoluent selon une autre variable, comme le temps par exemple. Un cas classique est l'évolution de la taille de plusieurs enfants au travers des années. L'évolution de chacun représente une courbe, et l'ensemble de ces courbes forme un échantillon de données fonctionnelles.

Une variable aléatoire fonctionnelle $Y(t)$ est une quantité aléatoire, univariée ou multivariée, telle que ses valeurs dépendent d'une variable univariée continue t définie sur une durée $[0,T]$, où T est la durée totale du phénomène observé. Lorsque univariée, $Y(t)$ s'interprète bien par une courbe. Sinon, les données ont p composantes telles que

$Y(t) = (Y_1(t), Y_2(t), \ldots, Y_p(t))$. L'ensemble de données qu'on possède pour l'étude est un ensemble d'observations $y_1(t), \ldots, y_N(t)$. Sous forme matricielle, on a

$$\mathbf{Y}(t) = \mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Np} \end{pmatrix}.$$

Elles peuvent provenir d'une même distribution statistique $f_Y(y_i; \Theta)$ avec θ l'ensemble des paramètres et $i = 1, \ldots, N$. Cependant, cette analyse se portera sur des données longitudinales, c'est-à-dire qu'on observe des phénomènes qui évoluent dans le temps. Ceci implique une dépendance entre les mesures observées dans le temps. Notre espace va donc se restreindre à des fonctions qui sont des processus stochastiques L^2 continus. Sommairement, les carrés des fonctions doivent être intégrables pour omettre des cas où la fonction diverge en un point. Par exemple la fonction $f(t) = t^{-1/2}$ n'a pas un carré intégrable sur [0,1]. Ensuite, cet espace de probabilités continues amène que

$$\forall\ t \in \mathcal{T},\ \ \lim_{h \to 0} \mathbb{E}\left[\|Y(t-h) - Y(t)\|^2\right] = \lim_{h \to 0} \int_{\mathcal{T}} \sum_{j=1}^{p} \mathbb{E}\left[(Y_j(t-h) - Y_j(t))^2\right] dt = 0.$$

Les fonctions aléatoires possèdent des probabilités continues tout le long des points $t \in \mathcal{T}$.

Avec un espace bien défini, on peut définir des statistiques de base ainsi que des opérations élémentaires. D'abord, la fonction moyenne et la variance fonctionnelle empiriques sont respectivement

$$\mu(t) = \mathbb{E}\left[Y(t)\right] \quad \text{et} \quad \mathbb{V}(t) = \mathbb{E}\left[(Y(t) - \mu(t))\,(Y(t) - \mu(t))'\right], \quad t \in \mathcal{T}.$$

Il est à noter que l'apostrophe désigne la transposée matricielle. Puisque les fonctions sont multivariées, $\mu(t) \in \mathbb{R}^p$ et $\mathbb{V}(t) \in \mathbb{R}^{p \times p}$, il évident que ces quantités ne sont pas disponibles et devront être estimées. Ces estimateurs sont identiques au cas multivarié et sont

$$\bar{y}(t) = \frac{1}{N} \sum_{i=1}^{N} y_i(t) \quad \text{et} \quad v(t) = \frac{1}{N-1} \sum_{i=1}^{N} \left(y_i(t) - \bar{y}(t)\right)^2 \ , \quad t \in \mathcal{T}$$

Il est à noter que le calcul de $\bar{y}(t)$ et $v(t)$ se simplifie lorsque les données sont centrées. Le produit scalaire entre deux fonctions en L^2 est :

$$\langle x, y \rangle = \int_{\mathcal{T}} x(t) y(t) dt.$$

La norme d'un vecteur devient simplement

$$\|y(t)\|^2 = \int_{\mathcal{T}} y^2(t) dt$$

Tous ces outils seront utilisés au travers de ce mémoire et permettront de définir des concepts élaborés comme l'analyse en composantes principales et l'estimation de paramètres lors de la modélisation.

Deux exemples sont présentés pour aider à non seulement comprendre l'espace fonctionnel qu'on vient d'établir, mais aussi afin de mieux visualiser l'impact des deux autres outils présentés dans ce chapitre. Ces deux exemples s'obtiennent grâce au logiciel **R** et de la librairie **fda**.

2.1.2. Quelques exemples

Le premier exemple est le jeu de données *Growth* [**29**], contenant la grandeur de plusieurs enfants (en cm) de 1 à 18 ans prises en 1954. Il y a 31 mesures disponibles pour chacun, mais elles sont observées irrégulièrement dans le temps. Il y a eu quatre mesures dans la première année, des mesures annuelles entre deux et huit ans, puis bisannuelles pour le reste. L'échantillon est constitué de 93 enfants: 39 garçons et 54 filles. Une visualisation est présentée dans le graphique 2.1. La particularité de ces données est que la poussée de croissance n'est pas la même pour tous, puisqu'elle diffère entre les deux sexes. Le défi de l'analyse fonctionnelle sera de déterminer cette variation de croissance, malgré la différence de sexe.

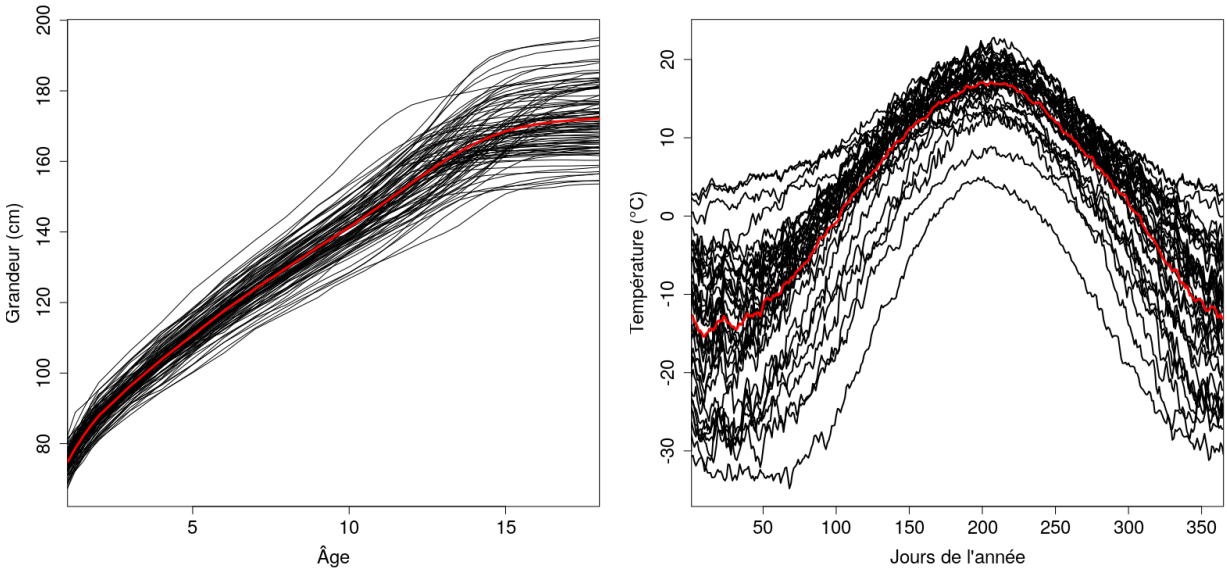

Fig. 2.1. À Gauche : Grandeur moyenne (en cm) des enfants entre 1 et 18 ans. À Droite : Température moyenne (en °C) par jour dans les différentes villes canadiennes pour une année. La courbe rouge représente la moyenne de chacun des échantillons.

Le deuxième est le jeu de données nommé *Weather* [**23**], qui décrit la température moyenne de chaque jour dans plusieurs villes au Canada. Les données ont été observées entre 1960 et 1994. Contrairement aux données *Growth*, les mesures sont prises à des temps équidistants. Les données sont dites *cycliques*, puisqu'elles décrivent un phénomène qui se répète dans le temps. L'hiver revient toujours chaque année avec de basses températures. De plus, le Canada n'a pas un climat constant dans tout le pays. C'est pour cela qu'on observe une grande variation de température entre les villes les jours d'hiver. L'analyse fonctionnelle devra capter le phénomène cyclique et cette grande variation.

Les ensembles de données précédents sont des échantillons, alors la variable de temps est donc discrète, de sorte que $t = (t_1,t_2,\ldots,t_n)$. Il est possible de reconstruire les fonctions à l'aide de modèles paramétriques, à partir des données observées.

2.2. Expansions par base de fonctions

Dans le cas de l'espace $\mathbb{R}^p$, il est possible de définir chacun de ces vecteurs par une combinaison linéaire des vecteurs de la base. Il serait possible de faire la même chose pour notre espace de fonctions, or cet espace est de dimension infinie. En d'autres mots, la suite infinie de fonctions linéairement indépendantes $\{\phi_1, \phi_2, \ldots\} = \{\phi_i\}_{i\geq 1}$ forme la base de L^2. C'est pour cela qu'on se limitera à un sous-ensemble de fonction qui générera un sous-ensemble $H \subset L^2$, contenant les fonctions que l'on souhaite analyser. Prenons $\{\phi_1(t), \ldots, \phi_q(t)\}$, la base qui génère le sous-espace de fonction $H \subset L^2$. Cette base possède des éléments linéairement indépendants et, si possible, orthogonaux entre-eux pour alléger le calcul. Pour un q assez grand, il est possible d'exprimer une fonction avec une combinaison linéaire de la façon suivante:

$$y_i(t) = \sum_{k=1}^{q} c_{ik}\phi_k(t) \tag{2.2.1}$$

pour un certain $q \in \mathbb{N}$ avec des coefficients $c_{ik} \in \mathbb{R}$, $i = 1,\ldots N,\ k = 1,\ldots,q$. Sous forme matricielle, les coefficients c_{ik} peuvent être exprimés par une seule matrice $\mathbf{C}$ de dimension $N \times q$ et les fonctions de la base par le q-vecteur $\boldsymbol{\phi}(t)$. Ainsi, on a

$$\mathbf{Y}(t) = \mathbf{C}\boldsymbol{\phi}(t) = \begin{pmatrix} c_{11} & c_{12} & \ldots & c_{1q} \\ c_{21} & c_{22} & \ldots & c_{2q} \\ \vdots & & \ddots & \vdots \\ c_{N1} & c_{N2} & \ldots & c_{Nq} \end{pmatrix} \begin{pmatrix} \phi_1(t) \\ \phi_2(t) \\ \vdots \\ \phi_q(t) \end{pmatrix}.$$

Il est possible d'exprimer $\mathbf{Y}(t) = \boldsymbol{\phi}(t)\mathbf{C}$, tant que les éléments $\boldsymbol{\phi}(t)$ et $\mathbf{C}$ sont définis de telle sorte que l'égalité (2.2.1) reste valide. Comme de fait, la matrice $\mathbf{C}$ est un paramètre

qui doit être estimé. On reviendra sur les méthodes d'estimation de ce paramètre, comme l'algorithme EM, dans le chapitre 3. Cette section est consacrée au choix de la base $\boldsymbol{\phi}(t) = (\phi_1(t), \ldots, \phi_q(t))'$.

La valeur q n'est pas fixe : c'est un hyper-paramètre à estimer tel qu'il minimise l'erreur entre la fonction originale et sa combinaison linéaire. Une faible valeur de q entraîne une sous-paramétrisation des données, et le contraire entraîne une sur-paramétrisation. La figure 2.2 présente l'impact du choix de q lors de la reconstruction des fonctions. Une courbe aléatoire $Y(t)$ a été générée telle que $Y(t) = \sin(2\pi t) + \varepsilon(t)$, et on l'a évaluée sur 45 points entre les temps 0 et 3. Le terme d'erreur ε est un vecteur aléatoire avec 45 composantes telles que chacune est identiquement et indépendamment distribuée selon une loi gaussienne centrée en 0 et de variance 0,5 notée par $\mathcal{N}(0; 0{,}5)$. La courbe est ensuite reconstruite à l'aide de la base de *B-splines* [**11**] avec 5, 15 et 25 fonctions de base, à partir de l'échantillon de points. Ces fonctions seront définies sous peu dans la section 2.2.2. Ces reconstructions se retrouvent dans les trois graphiques de la figure 2.2. La vraie fonction $\sin(2\pi t)$ est représentée par les courbes bleues et l'ensemble de l'échantillon par les points noirs. Il est évident que l'approximation est faible lorsque 5 fonctions sont utilisées. Avec 25, le bruit affecte le lissage davantage ce qui mène à une sur-paramétrisation, donc les courbes estimées sont sensibles aux variations des observations. C'est pour cela qu'il faut déterminer l'hyper-paramètre q avec parcimonie pour obtenir une approximation sensée, comme le démontre le graphique avec 15 bases. C'est avec des critères de sélection comme le BIC [**26**] et l'AIC [**2**], discutés dans le chapitre 3, que l'on pourra déterminer le choix final de q.

Maintenant, un choix de la base de fonctions doit être fait. Il n'y a pas une seule bonne réponse à cette question, mais il est judicieux de choisir des fonctions qui représentent bien le phénomène observé. Un choix classique est de prendre une série de puissances $\{1,t,t^2,t^3,\ldots\}$ ou les séries de Fourier $\{1, \sin(\omega t), \cos(\omega t), \sin(2\omega t), \cos(2\omega t) \ldots\}$, qui sont couramment utilisées pour les données chronologiques. Il en existe plusieurs autres, mais dans certains cas, il est préférable de construire les fonctions par morceaux par la méthode des *splines*.

2.2.1. Bases par splines

Les fonctions par splines sont populaires pour l'approximation de données fonctionnelles non-cycliques. Elles permettent un calcul numérique rapide de polynôme avec grande flexibilité. Dans un premier temps, l'anatomie d'un spline sera présentée, suivie de la populaire base de fonction *B-splines*, utilisée dans la modélisation fonctionnelle.

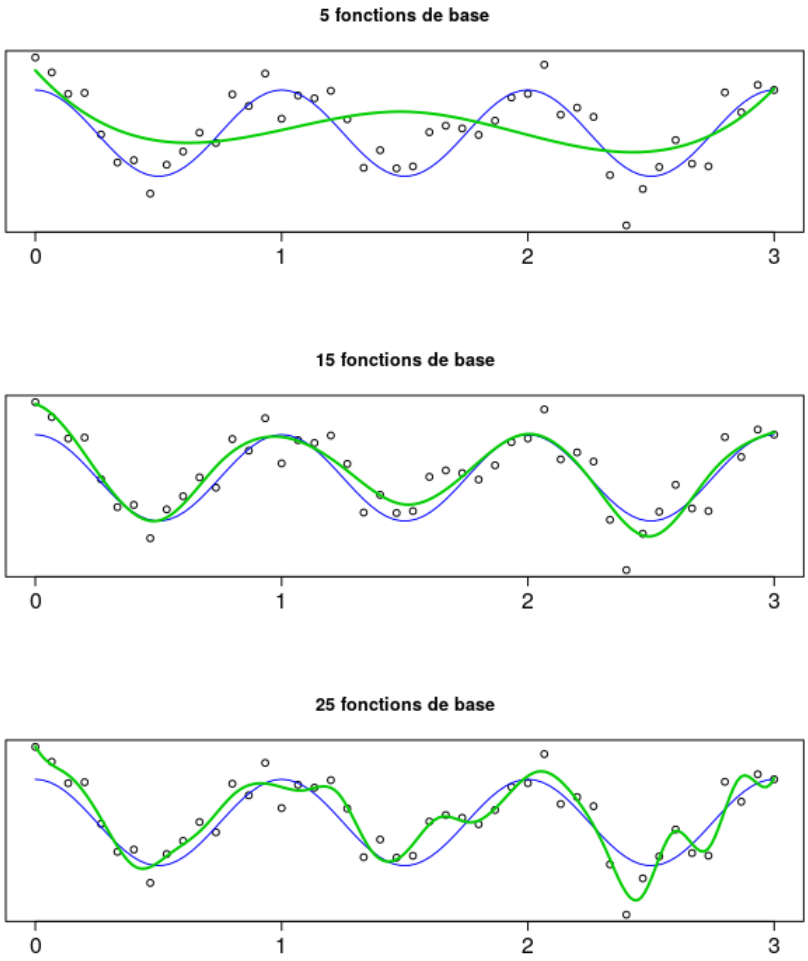

Fig. 2.2. Impact du nombre q de fonctions de base a sur le lissage des données.

Le calcul des splines est principalement centré sur le concept de *nœud*. Avec l'intervalle de temps $[0,T]$, les nœuds $\xi_0, \xi_1, \ldots, \xi_q$ séparent l'intervalle $[0,T]$ en q sous intervalles, préférablement de même longueur. Les points aux extrémités de l'intervalle sont définis par $\xi_0 = 0$ et $\xi_q = T$. L'idée est de faire une régression polynômiale dans chaque sous-intervalle pour que le somme de ces régressions forment une courbe lisse, qui approxime la densité d'une population. Chacun de ces polynômes forme un spline de degré $M-1$ et d'ordre M. L'*ordre* est le nombre de coefficients nécessaire pour définir chacun de ces polynômes. Cette approche permettra une plus grande flexibilité, tout en limitant la puissance du polynôme pour éviter d'augmenter la taille des calculs.

Il est important que ces bases de fonctions forment une courbe qui possède des dérivées continues aux nœuds. Les graphiques de la figure 2.3 montrent l'importance de ce problème. Un échantillon a été généré selon des variables aléatoires $X_i(t) = \sin(2\pi t/3) + \varepsilon_i$ avec

$\varepsilon_i \sim \mathcal{N}(0; 0.5)$ et $i = 1 \ldots, 45$. Les deux nœuds $\xi_1 = 1$ et $\xi_2 = 2$ séparent cet échantillon en trois part égales sur lesquelles une régression cubique locale est effectuée. Le résultat de cette régression sera donnée par la fonction f. Le premier montre les résultats des régressions sans aucune contrainte de continuité aux nœuds $\xi_1 = 1$ et $\xi_2 = 2$. Il est clair que la somme de ces courbes n'est pas adéquate. On peut rajouter des contraintes de continuité aux nœuds. Avec ξ_j^- la valeur la plus proche et inférieure à ξ_j et ξ_j^+ la valeur la plus proche et supérieure à ξ_j, on impose que $f(\xi_1^-) = f(\xi_1^+)$ de même que $f(\xi_2^-) = f(\xi_2^+)$. Ceci assure que la fonction f obtenue par la régression est continue, comme le démontre le graphique au coin supérieur droit de la figure 2.3. En ajoutant des contraintes de continuité sur les dérivées de f, on arrive à une courbe qui possède une dérivée seconde aux nœuds et qui est lisse pour l'œil humain, comme le démontre le graphique au coin inférieur droit. C'est aussi pour cette raison que les polynômes de degrés 3 sont populaires dans la reconstruction par expansion de base.

Il nous faut donc une base de splines qui remplisse ces conditions. Il en existe plusieurs, mais les méthodes de partitionnement présentées dans cet ouvrage utiliseront les *B-splines*.

2.2.2. B-splines

Les B-splines forment une base de fonction polynômiale par intervalles et elles sont construites par morceaux et par itérations. Elles sont reconnues pour leur simplicité de calcul et leur flexibilité. Pour bien définir ces splines, il faut augmenter le nombre d'éléments dans la suite de nœuds $\xi_0, \ldots, \xi_q$.

Avec les points aux extrémités ξ_0 et ξ_q, les nœuds $\tau_1, \ldots, \tau_{q+2M}$ sont rajoutés de sorte que

- $\tau_1 \leq \ldots \leq \tau_M \leq \xi_0$,
- $\tau_{i+M} = \xi_i$, pour tout $i = 1, \ldots, q$,
- $\xi_{q+1} \leq \tau_{q+M+1} \leq \ldots \leq \tau_{q+2M}$.

On a donc une suite de nœuds augmentés $\{\tau_1, \ldots, \tau_{q+2M}\}$. Il n'est pas nécessaire que les nœuds situés au-delà de la limite prennent des valeurs hors de l'intervalle [0,T]. En pratique, leurs valeurs sont $\tau_0 = \ldots = \tau_M = 0$ et $\tau_{q+M+1} = \ldots = \tau_{q+2M} = T$. Le reste des nœuds $\tau_{M+1}, \ldots, \tau_{q+M}$ sont placés de sorte que l'intervalle [0,T] soit divisé en sous-intervalles de mêmes longueurs. À présent, les B-splines seront définis pour chacun de ces sous-intervalles $[\tau_i, \tau_{i+1}]$, $i = 1, \ldots, q + 2M$. Ensemble, ces splines forment une base polynomiale par intervalles de degré $M-1$ et d'ordre M. Soit $B_{i,m}(t)$ la $i^{ième}$ fonction de base du B-spline d'ordre $m \leq M$ pour la suite de nœuds $\tau = \{\tau_1, \ldots, \tau_{q+2M}\}$. Ils sont définis récursivement comme suit :

$$B_{i,1}(t) = \begin{cases} 1 & \text{si } \tau_i \leq t < \tau_{i+1} \\ 0 & \text{sinon} \end{cases},$$

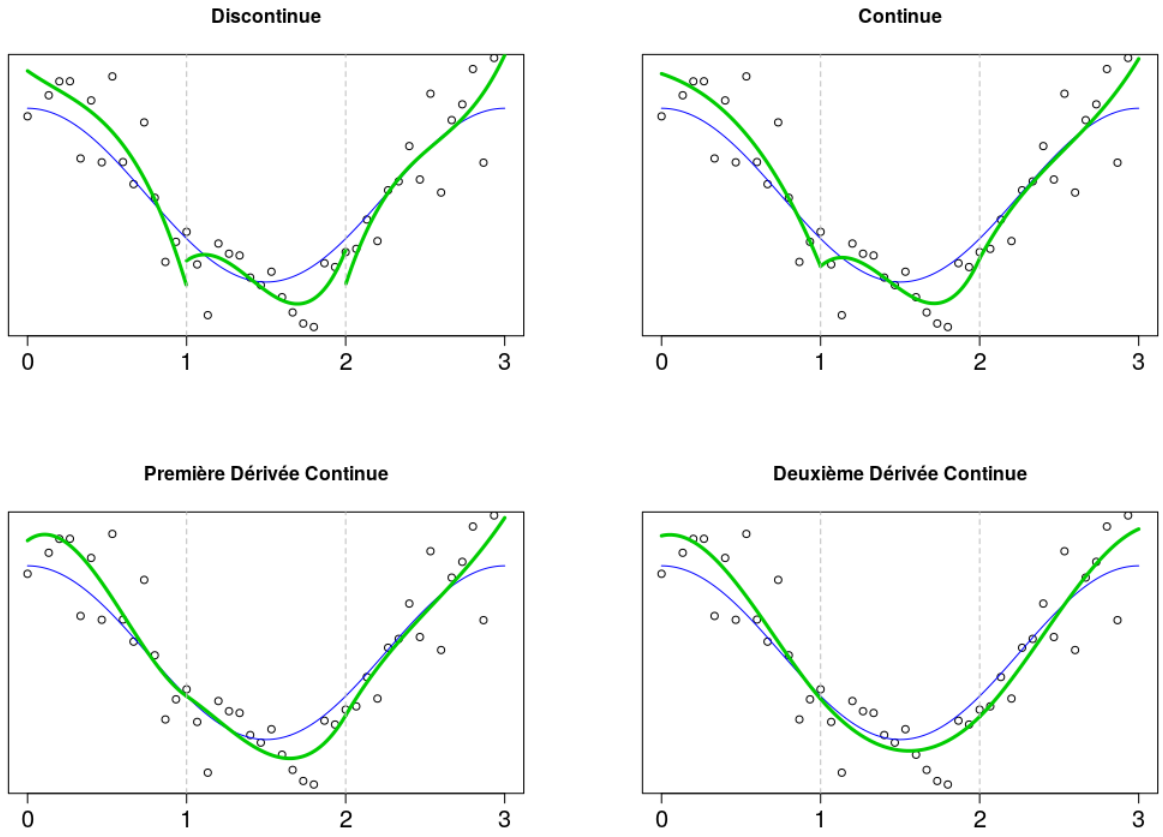

Fig. 2.3. Impact de la continuité sur une régression cubique locale, sur un échantillon cyclique. De gauche à droite et de haut en bas, on augmente le degré de continuité en commençant par une régression discontinue.

pour $i = 0, \ldots, q$, et

$$B_{i,m}(t) = \frac{t - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(t) + \frac{\tau_{i+m} - t}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(t)$$

pour $i = 1, \ldots, q$ et pour $m = 2, \ldots, M$.

La duplication des nœuds joue un rôle dans la continuité des fonctions. Supposons que le nœud ξ_i est répété k fois. Alors les B-splines d'ordre M définis en ce point auront alors leurs $M - k$-ième dérivées discontinues. Dans la construction initiale des B-splines, les nœuds aux extrémités ξ_0 et ξ_q sont répétés M fois, donc les fonctions sont discontinues aux extrémités. Cette situation est plus représentative de la réalité puisque les observations sont définies dans un intervalle de temps fixé.

Trois exemples de base par B-splines sont présentés sur la figure 2.4. De là, on remarque le changement du degré des polynômes selon l'ordre des splines. L'ordre 4 est le plus

populaire puisque les polynômes sont cubiques et ils ont donc une seconde dérivée continue.

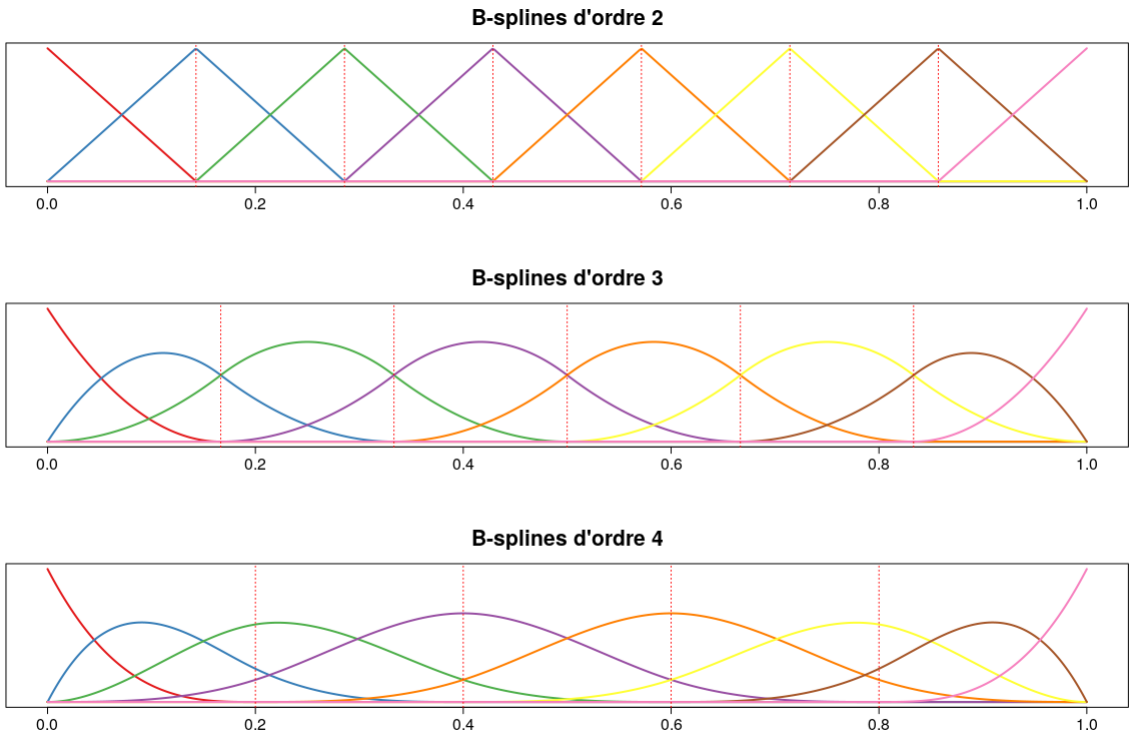

Fig. 2.4. Graphique des fonctions de base unidimensionnelles d'un B-splines d'ordre 3 sur l'intervalle [0,1] séparé en 10 sous-intervalles.

Cette base n'est pas lourde en calcul numérique. En effet, avec N observations et $q+M$ fonction de base, le nombre d'opérations par seconde du calcul de l'estimation de la matrice des coefficients $\mathbf{C}$ par moindres carrées est de $O(N(q+M)^2+(q+M)^3)$.

2.3. Analyse en composantes principales

Une fois que la reconstruction des données en fonctions est faite, les données sont représentées dans un sous-espace fonctionnel de dimension q. Cependant, une grande valeur de q va nécessairement alourdir la charge de calcul numérique. La même problématique survient dans le cas multivarié : des données qui résident dans un espace $\mathbb{R}^d$ tel que d est élevée entraîneront aussi de lourds calculs. Or, il existe une solution dans le cas multivarié qui permet de réduire la dimension de l'espace : c'est l'analyse en composantes principales. Cet outil peut être ajusté au cas fonctionnel.

L'analyse en composantes principales (ACP) permet de réduire la dimension des données en les projetant sur un sous-espace linéaire de dimension inférieure. Il existe plusieurs formes et approches de l'ACP, dont une adaptée pour le cas fonctionnel utilisé dans les modèles de partitionnement. L'ACP classique sera d'abord présentée, suivie de sa version fonctionnelle. On démontrera ensuite une possibilité de résoudre l'analyse en composante fonctionnelle par une ACP classique, ce qui simplifiera la charge de travail.

2.3.1. Analyse en composantes principales

Commençons par le cas où les données sont dans un espace $\mathbb{R}^p$. Le but de l'ACP est de projeter chaque variable $\mathbf{x} \in \mathbb{R}^p$ dans un sous-espace de dimension inférieure $\mathbb{R}^k$ avec la base $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$, $k < p$. Par contre, il y aura nécessairement une perte d'information puisque la dimension du nouvel espace est inférieure à celle de l'original. Lorsqu'on projette sur un espace généré par la base de vecteurs $\mathbf{u}_1 \ldots, \mathbf{u}_k$, cette perte d'information est la distance entre les points originaux et les vecteurs de la base. Les vecteurs qui minimisent cette distance sont aussi ceux qui maximisent la variance des points projetés dans le sous-espace généré par chacun des vecteurs. Si on regarde l'exemple dans la figure 2.5, la droite verte la plus longue représente la direction où la variabilité des données est la plus prononcée. Les distances entre les points et cette droite sont en moyenne inférieures à celles entre les points et l'autre droite. C'est pour cette raison que la variabilité des données doit être préservée le plus que possible lorsque qu'un point $\mathbf{x}$ est projeté sur un élément $\mathbf{u}_i$ de la base du nouveau sous-espace.

Avec un ensemble de points $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ dans $\mathbb{R}^p$, nous considérons la moyenne et la matrice de covariance empirique donnée par

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{et} \quad \mathbf{S} = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})'.$$

La variance des points projetés dans la direction d'un premier vecteur $\mathbf{u}_1$ est

$$\frac{\langle \mathbf{u}_1, \mathbf{S} \rangle}{\langle \mathbf{u}_1, \mathbf{u}_1 \rangle}\mathbf{u}_1 = \frac{\mathbf{u}_1'\mathbf{S}}{\|\mathbf{u}_1\|^2}\mathbf{u}_1 = \mathbf{u}_1'\mathbf{S}\mathbf{u}_1.$$

On souhaite donc trouver $\mathbf{u}_1$ qui maximise la projection $\mathbf{u}_1'\mathbf{S}\mathbf{u}_1$ sous la contrainte $\|\mathbf{u}_1\|^2 = 1$ pour restreindre les distances entre les points originaux et leur projection, en plus d'assurer l'unicité de la solution. En utilisant le multiplicateur de Lagrange, on a

$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \iff \mathbf{u}_1'\mathbf{S}\mathbf{u}_1 = \lambda_1,$$

c'est-à-dire, le vecteur $\mathbf{u}_1$ correspond au premier vecteur propre de $\mathbf{S}$ associé à la valeur propre λ_1. Soient $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k$ les valeurs propres de $\mathbf{S}$, et soient $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ leur vecteurs propres. Le vecteur propre $\mathbf{u}_1$ explique la plus grande variabilité parmi tous les autres vecteurs. Ce résultat est avantageux, car la base de l'espace propre formé par $\mathbf{u}_1, \ldots, \mathbf{u}_k$ doit être orthonormée puisque la matrice $\mathbf{S}$ est symétrique. La contrainte

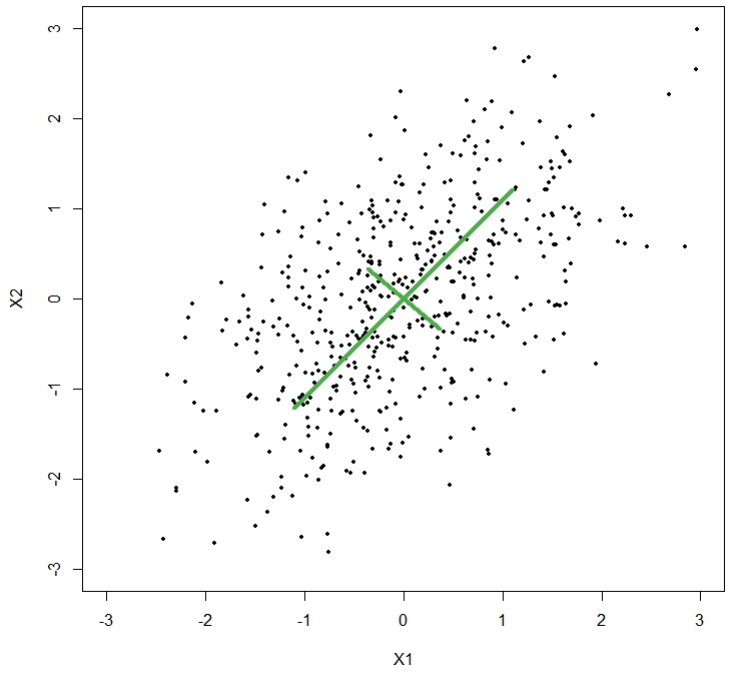

Fig. 2.5. Composantes principales d'un échantillon normal bivarié.

$\|\mathbf{u}_1\|^2 = 1$ est respectée et l'orthogonalité implique que chaque projection sur un vecteur propre représente une information non expliquée par les autres. Donc l'ACP est réduite au problème algébrique de l'équation propre dans $\mathbb{R}^p$.

La figure 2.5 présente un exemple d'une ACP effectuée sur une simulation. Un échantillon a été généré selon une normale bivariée avec une corrélation modérée ($Corr(X_1,X_2) \approx 0{,}577$). Les segments verts représentent les vecteurs propres dilatés par leurs valeurs propres. Ces vecteurs sont les composantes principales. La plus grande composante est sur la direction de la plus grande variabilité du nuage de points. L'autre vecteur, nettement plus petit, représente la variabilité restante et il est orthogonal au premier.

Dans ce mémoire, l'espace que l'on traite est fonctionnel et ne suit donc pas les mêmes règles que l'espace $\mathbb{R}^p$. Une variante fonctionnelle est obtenue facilement en changeant quelques concepts.

2.3.2. Analyse en composantes principales fonctionnelles

Le problème de dimension prédomine l'espace fonctionnel puisque cet espace est de dimension infinie, d'où l'importance de réduire l'espace. L'ACP devient ainsi un outil

important dans l'analyse fonctionnelle. Cette section s'inspire fortement du chapitre 8 de Ramsay et Silverman[**24**] et cette théorie se retrouve aussi dans l'article [**3**].

Le problème reste le même : trouver un espace, cette fois fonctionnel, sur lequel les observations $Y_1(t), \ldots, Y_N(t)$ peuvent y être projetées tout en préservant un maximum de variabilité possible. La solution est analogue à celle de l'ACP multivariée, mais il faut définir l'*opérateur de la variance* $\mathcal{V}$. D'abord, en supposant que les données sont centrées et réduites, la matrice de covariance empirique pour un échantillon de taille N $v_N(s,t)$ est définie par

$$v(s,t) = \frac{1}{N}\sum_{j=1}^{N} Y_j(t)Y_j(s).$$

Pour $f : \mathcal{T} \to \mathbb{R}^p$, une fonction à valeur en $\mathbb{R}^p$, l'opérateur de la covariance est défini par

$$\mathcal{V}f = \int_{\mathcal{T}} v(\cdot,t)f(t)dt.$$

Cet opérateur permet d'établir l'analogie entre l'ACP et l'ACP fonctionnelle. Dans le cas multivarié, on cherchait un ensemble de vecteurs orthogonaux qui maximisait la quantité $\mathbf{u}_i'\mathbf{S}\mathbf{u}_i$ qui est équivalente au produit scalaire $\langle \mathbf{u}_i, \mathbf{S}\mathbf{u}_i \rangle$, le tout sous la contrainte $\|\mathbf{u}_i\|^2 = 1, \forall i$. L'analogue fonctionnelle est

$$\max \langle \varphi_i, \mathcal{V}\varphi_i \rangle \quad \text{tel que} \quad \|\varphi_i\|^2 = 1, \forall i = 1,...,q. \tag{2.3.1}$$

Mais en quoi (2.3.1) est équivalent au problème de l'ACP multivariée? On veut minimiser les distances au carrés entre les points de l'espace original et leurs projections sur le nouvel espace, c'est-à-dire

$$\min_{\|\varphi_i\|^2=1} \mathbb{E}\left[\|Y - (Y \cdot \varphi_i)\varphi_i\|^2\right]. \tag{2.3.2}$$

Par souci de notation, $Y \cdot \varphi_i \equiv \langle Y, \varphi_i \rangle$. En développant l'espérance,

$$\begin{aligned}\min_{\|\varphi_i\|^2=1} \mathbb{E}\left[\|Y - (Y \cdot \varphi_i)\varphi_i\|^2\right] &= \min_{\|\varphi_i\|^2=1} \left(\mathbb{E}\left[\|Y\|^2\right] - \mathbb{E}\left[(Y \cdot \varphi_i)^2\right]\right) \\ &= \min_{\|\varphi_i\|^2=1} \left(\mathbb{E}\left[\|Y\|^2\right] - \mathbb{V}\left[Y \cdot \varphi_i\right]\right).\end{aligned}$$

Puisque les observations Y sont fixées, alors $\mathbb{E}\left[\|Y\|^2\right]$ l'est aussi. Ainsi, minimiser la distance revient à maximiser la variance du produit scalaire $Y \cdot \varphi$. Or, celle-ci peut s'exprimer comme

suit

$$\begin{aligned}
\mathbb{V}[Y \cdot \varphi_i] &= \frac{1}{N}\sum_{j=1}^{N}(Y_j \cdot \varphi_i)^2 \\
&= \frac{1}{N}\sum_{j=1}^{N}\left(\int_{\mathcal{T}} Y_j(t)\varphi_i(t)dt\right)^2 \\
&= \frac{1}{N}\sum_{j=1}^{N}\int_{\mathcal{T}}\int_{\mathcal{T}} Y_j(t)\varphi_i(t)Y_j(s)\varphi_i(s)dtds \\
&= \int_{\mathcal{T}}\int_{\mathcal{T}} v(s,t)\varphi_i(t)\varphi_i(s)dtds \\
&= \int_{\mathcal{T}} \mathcal{V}\varphi_i(s)ds = \langle \varphi_i, \mathcal{V}\varphi_i \rangle
\end{aligned}$$

Par ce fait, le problème (2.3.2) est équivalent au problème (2.3.1). Ramsay et Silverman ([**24**]) expliquent que ce problème est résolu par l'analyse des valeurs et fonctions propres

$$\mathcal{V}\varphi_i = \lambda_i\varphi_i, \ \forall i, \tag{2.3.3}$$

où $\varphi_1, \ldots, \varphi_q$ sont ce qu'on appelle des *fonctions propres* qui satisfont l'équation (2.3.3). Par définition, ils ne peuvent pas être nuls. Les fonctions φ_i possèdent les mêmes propriétés que les vecteurs propres trouvés dans l'ACP: chacune explique différentes variations dans les données puisqu'elles sont orthogonales entre elles. Malgré le fait que l'espace fonctionnel est de dimension infinie, le nombre de fonctions propres sera limité par la taille N de l'échantillon. Puisque les données sont centrées, la matrice des observations $\mathbf{Y}$, qui est $p \times N$, aura un rang d'au plus $N-1$. Ceci implique que l'opérateur fonctionnel $\mathcal{V}$ aura $\min(N-1,p)$ valeurs propres.

Les graphiques de 2.6 représentent les quatre premières composantes fonctionnelles des données *Weather*. Le pourcentage dans le titre de chacun correspond à la proportion de variabilité expliquée par chaque fonction principale. Les graphiques comparent les jours de l'année contre la valeur de la composante, c'est-à-dire le poids λ_i. Si on observe la première, elle explique 88,8% de la variabilité des données et les poids sont plus importants pour les jours d'hiver. Cette information est cohérente avec les températures, des fois chaotiques, de l'hiver canadien. Les autres composantes sont orthogonales avec la première et expliquent la variabilité restante. La deuxième composante possède aussi des poids importants pour les jours d'hiver et une importante contribution négative des jours d'été. Les troisième et quatrième composantes représentent ensemble 2,4% de la variabilité, ce qui peut sembler négligeable. Or, il se peut que ces composantes expliquent d'autres aspects importants de la donnée, comme les cas extrêmes par exemple. Il sera préférable d'analyser davantage les composantes principales, afin de mieux les interpréter. Ramsay et Silverman proposent certaines stratégies à cet égard dans [**23**]. Pour notre cas, le nombre de composantes

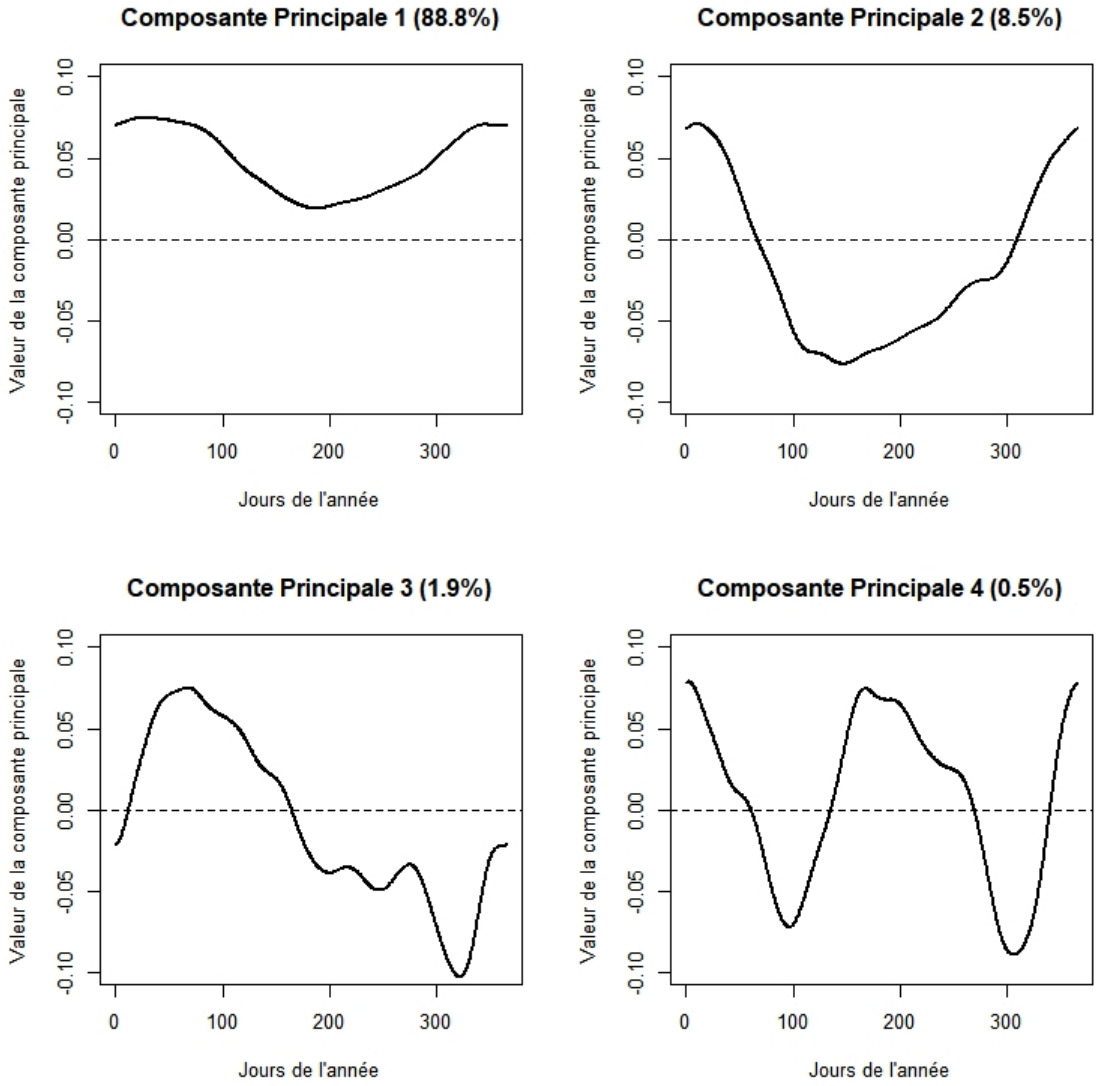

Fig. 2.6. Analyse en composantes principales fonctionnelles sur la base de données *Weather*.

retenues sera en lien avec le BIC et l'AIC.

Concrètement, on travaille avec des fonctions reconstruites telles que $\mathbf{Y} = \mathbf{C}\boldsymbol{\phi}(t)$. Les fonctions propres peuvent posséder une expansion par fonction de base telle que $\varphi_j(t) = \boldsymbol{\phi}(\boldsymbol{t})' B_j$ avec $j = 1, \ldots, q$ et B_j un vecteur de coefficients. Cette interprétation des fonctions propres permettra d'écrire l'équation (2.3.3) de façon matricielle.

2.3.3. Simplification de l'ACPF

On va à présent démontrer que le problème propre fonctionnel peut être simplifié en un problème propre matriciel.

Proposition 2.3.1. *Soit l'ensemble de données* $\mathbf{Y} = (y_1, \ldots, y_N)'$ *qui peut s'exprimer par la combinaison linéaire* $\mathbf{C}\boldsymbol{\phi}$*, où* $\mathbf{C}$ *est une matrice* $N \times q$ *de coefficients et* $\boldsymbol{\phi}$ *est le vecteur* $q \times 1$ *des fonctions de base.*

De plus, soit $\mathcal{V}$ *l'opérateur de covariance et les fonctions propres* $\varphi_1, \ldots, \varphi_q$ *qui satisfont l'égalité*

$$\mathcal{V}\varphi_j = \lambda_j \varphi_j, \quad j = 1, \ldots, q.$$

avec $\lambda_1 \geq \ldots \geq \lambda_q$ *les valeurs propres associées. On suppose que les fonctions propres possèdent une expansion par fonction de base tel que* $\varphi_j(t) = \boldsymbol{\phi}(t)'\mathbf{B}_j$, $j = 1, \ldots, q$*. Alors les mêmes valeurs propres* $\lambda_1 \geq \ldots \geq \lambda_q$ *satisfont aussi l'égalité*

$$N^{-1}\mathbf{W}^{1/2}\mathbf{C}'\mathbf{C}\mathbf{W}^{1/2}\mathbf{u}_j = \lambda_j \mathbf{u}_j, \quad j = 1, \ldots, q,$$

où $\mathbf{W} = \int_0^T \boldsymbol{\phi}(t)\boldsymbol{\phi}(t)'dt$ *est une matrice non-singulière,* $\mathbf{u}_j = \mathbf{W}^{1/2}\mathbf{B}_j$*, et* $\mathbf{W}^{1/2}$ *est la matrice racine carré de* $\boldsymbol{W}$*, c'est-à-dire* $\mathbf{W} = \mathbf{W}^{1/2}\mathbf{W}^{1/2}$*.*

Démonstration. D'abord, la covariance fonctionnelle prend la forme matricielle

$$v(s,t) = N^{-1}\boldsymbol{\phi}(s)'\mathbf{C}'\mathbf{C}\boldsymbol{\phi}(t).$$

L'opérateur de la covariance appliquée à la fonction propre $\varphi_j = \boldsymbol{\phi}'\mathbf{B}_j$ donne

$$\begin{aligned} \mathcal{V} = \varphi_j(t) \int_0^T v(s,t)\varphi_t(t)dt &= \int_0^T N^{-1}\boldsymbol{\phi}(s)'\mathbf{C}'\mathbf{C}\boldsymbol{\phi}(t)\boldsymbol{\phi}(t)'\mathbf{B}_j dt \\ &= \boldsymbol{\phi}(s)'N^{-1}\mathbf{C}'\mathbf{C}\mathbf{W}\mathbf{B}_j. \end{aligned} \tag{2.3.4}$$

De ce fait, l'équation propre fonctionnelle s'écrit sous la forme suivante :

$$\boldsymbol{\phi}(s)'N^{-1}\mathbf{C}'\mathbf{C}\mathbf{W}\mathbf{B}_j = \lambda_j \boldsymbol{\phi}(s)'\mathbf{B}_j$$

Cette égalité peut se simplifier en multipliant par la gauche chaque partie par $\boldsymbol{\phi}(s)$ et en intégrant chaque termes selon la variable s par la suite. Ceci donne

$$\begin{aligned} \int_0^T \boldsymbol{\phi}(s)\boldsymbol{\phi}(s)'N^{-1}\mathbf{C}'\mathbf{C}\mathbf{W}\mathbf{B}_j ds &= \int_0^T \lambda_j \boldsymbol{\phi}(s)\boldsymbol{\phi}(s)'\mathbf{B}_j ds \\ N^{-1}\mathbf{C}'\mathbf{C}\mathbf{W}\mathbf{B}_j &= \lambda_j \mathbf{B}_j \end{aligned}$$

En posant $\mathbf{u}_j = \mathbf{W}^{1/2}\mathbf{B}_j$, la dernière équation devient

$$N^{-1}\mathbf{W}^{1/2}\mathbf{C}'\mathbf{C}\mathbf{W}^{1/2}\mathbf{W}^{1/2}\mathbf{B}_j = \lambda_j\mathbf{W}^{1/2}\mathbf{B}_j$$

$$N^{-1}\mathbf{W}^{1/2}\mathbf{C}'\mathbf{C}\mathbf{W}^{1/2}\mathbf{u}_j = \lambda_j\mathbf{u}_j, \text{ pour } j = 1,\ldots,q. \tag{2.3.5}$$

□

Cette équivalence permettra de simplifier le calcul des valeurs propres par un calcul matriciel. En calculant $\mathbf{B}_j = \mathbf{W}^{-1/2}\mathbf{u}_j$ pour chaque $j = 1,\ldots,q$, on retrouve les fonctions propres à l'aide de la combinaison linéaire établie plus haut. Il est à noter que si la base de fonction est orthonormée, alors $\mathbf{W} = \mathbf{I}_q$, l'équation (2.3.5) se simplifie davantage. Il faudrait alors seulement faire une analyse des vecteurs propres de la matrice symétrique $N^{-1}\mathbf{C}'\mathbf{C}$ d'ordre q.

La structure et les outils de l'analyse fonctionnelle forment ensemble la première partie du partitionnement fonctionnel. Comme le titre du chapitre suivant le suggère, la prochaine étape est d'établir le concept de partitionnement au sens général.

Chapitre 3

Méthodes de partitionnement

Un objectif courant en statistique, et notamment dans l'apprentissage machine, est de regrouper les données en sous-groupes pour ainsi les classifier. On rappelle que cet ouvrage se concentre sur la classification et la prédiction de trajectoires de voitures. Il est donc impératif que les concepts liés aux partitionnements soient établis. Pour ce faire, une définition du partitionnement des données sera mise en place, puis deux approches seront explorées : la première se base sur la distance entre les objets pour établir un critère de classification. La seconde, elle, tente de modéliser les données pour ainsi calculer les probabilités d'appartenance à un groupe. Enfin, on présentera le cadre de la modélisation en introduisant le mélange de densités, d'estimation de paramètres par l'algorithme EM et de critère de sélection.

3.1. Définition du partitionnement

Lorsqu'on fait face à un jeu de données hétérogène, il est parfois possible de regrouper des individus en sous-ensembles homogènes formant des *partitions*. On peut aussi dire qu'ils appartiennent à la même classe ou au même groupe. Les individus se retrouvant dans chacune de ces partitions partagent des caractéristiques communes. Dans l'exemple *Growth*, il est possible de partitionner les courbes selon le sexe des enfants. Dans le cas des trajectoires des voitures, les classes seront leurs directions. La grande question dans l'analyse en partition est de déterminer le ou les caractéristiques, si elles existent, qui permettent de regrouper des observations en des groupes homogènes. En d'autre mots, on souhaite trouver des regroupements tels que les observations dans chacun d'entre eux possèdent des comportements similaires.

Selon la théorie des ensembles de [**27**], on définit l'ensemble des données par $\mathcal{Y} = \{y_1, y_2 \ldots, y_N\}$ non-vide. Une partition $\mathcal{P}$ de $\mathcal{Y}$ est un regroupement de sous-ensembles $P_1, P_2, \ldots, P_G$ tels qu'ils respectent les trois conditions suivantes :

(1) $\forall g \in 1, \ldots, G, \quad P_g \neq \varnothing$
(2) $\bigcup_{g=1}^{G} P_g = \mathcal{Y}$
(3) $P_i \bigcap P_j = \varnothing, \quad \forall i \neq j.$

Cette définition ne remplit pas tous les critères nécessaires pour la partition statistique. En effet, on souhaite trouver les partitions telles que les éléments de chaque sous-ensemble P_g ont une caractéristique commune, chose qui n'est pas définie ici. En d'autres mots, on désire des partitions homogènes. Il faut donc définir le concept de variable latente: des variables sous-jacentes dans les données statistiques. Par exemple, même si les données sur les trajectoires de voitures ne comprennent pas la direction de chaque voiture, c'est une variable qui a un impact sur celles-ci et qu'on souhaite déterminer.

Formellement, soient $z_1, z_2, \ldots, z_N$ les réalisations des variables latentes d'un ensemble de données telles que $z_i \in \{1, \ldots, G\}$. L'ensemble des paires $\{(y_1,z_1), (y_2,z_2), \ldots, (y_N,z_N)\}$ forme les données complètes. On notera chacune de ces paires par $x_i = (y_i,z_i)$ pour tout $i = 1, \ldots,N$ et l'ensemble des données complètes par $\mathbf{X}$. Avec l'exemple de *Growth*, les variables latentes seraient le sexe tel que $z_i = 0$ lorsque l'individu i est un homme, $z_i = 1$ si c'est une femme.

Ainsi, on souhaite obtenir des partitions $P_1, P_2, \ldots, P_G$ telles que $P_g = \{y_i : z_i = g\}$. Il y a deux grandes approches du partitionnement: soit les points dans l'espace multidimensionnel sont regroupés soit selon leur distance entre eux, soit selon la probabilité qu'ils appartiennent à un groupe. Ces deux approches vont être à présent discutées, dans cet ordre.

3.2. Méthodes basées sur la distance

Ce type de partitionnement tente de regrouper les observations ayant une distance minime entre elles ou selon un point central commun. Elles sont excessivement populaires dans le monde de l'intelligence artificielle, de par leur simplicité et leur efficacité. Il est possible de les adapter à plusieurs situations, tant que la notion de distance existe dans l'espace des éléments en question.

3.2.1. K plus-proche-voisins

Aussi connu sous le nom *K-nearest-neighbours*, l'algorithme des K-plus-proches-voisins est certainement une des plus simples parmi toutes les méthodes basées sur la distances. En revanche, elle nécessite un ensemble d'entraînement.

Soit l'ensemble d'entraînement $\mathbf{X}_T$ contenant les d'éléments $(y_1, z_1), \ldots, (y_{N_T}, z_{N_T})$, où $N_T \in \mathbb{N}$, avec G groupes possibles. Les variables latentes $z_1, \ldots, z_{N_T}$ sont connues. Maintenant, la nouvelle paire d'éléments (y^*, z^*) est introduite et Z^* est inconnue. L'approche des K-plus-proches-voisins consiste à observer les K éléments y_i les plus proches du point y^*, selon une distance $\|\cdot\|$, puis à sélectionner la classe la plus fréquente parmi ces éléments.

La figure 3.1 présente un exemple de la classification avec cette méthode. Le point en noir représente la donnée à classifier, alors que les points rouges et bleus proviennent de l'ensemble d'entraînement. On veut déterminer la classe (dans l'exemple, elle est représentée par les couleurs bleu et rouge) du point noir. Si on se fie au trois premiers voisins, la classe bleue est la plus fréquente. Or, lorsque les cinq premiers voisins sont observés, la classe rouge l'emporte. Même si l'exemple est simple, il montre que le choix du nombre de voisins K, qui est un hyper-paramètre, affecte directement la classification.

Cette méthode s'entraîne avec un ensemble de données dont les partitions sont connues, c'est ce qu'on appelle l'*apprentissage supervisé*. En revanche, cette information n'est pas toujours accessible. L'algorithme peut être ajusté lorsque les données ne sont pas complètes, pour ainsi «découvrir» les partitions. Il s'agit de l'algorithme des *K-moyennes*, et on parlera alors d'apprentissage *non-supervisé*.

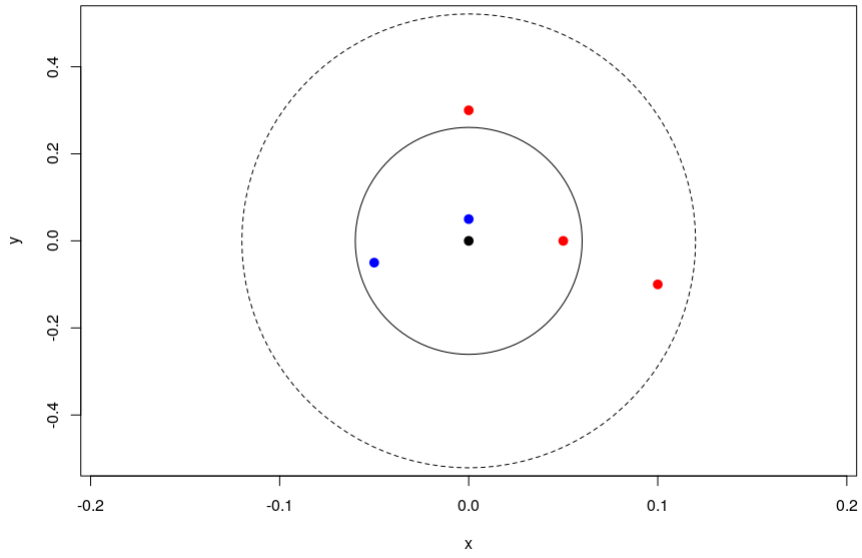

Fig. 3.1. Exemple de la classification K-plus-proches-voisins.

3.2.2. K-Moyennes

La technique des k-moyennes consiste à faire bouger des points centraux, (qu'on appelle *centroïdes*), afin que ceux-ci finissent dans le cœur de chaque partition. Pour bien expliquer les détails de cet algorithme, supposons que l'on a un ensemble de données $y_1, \ldots, y_N$ à N éléments, et que l'on souhaite partitionner cet ensemble en G partitions avec n_g éléments pour $g = 1, \ldots, G$. Les données peuvent être univariées, multivariées ou même fonctionnelles, il faut surtout qu'une mesure de distance soit bien définie. Avec la partition $\mathcal{P} = \{P_1, P_2, \ldots, P_G\}$, la fonction $p : \{1, \ldots, N\} \to \{1, \ldots, G\}$ est définie telle que $p(i) = g$ si $y_i \in P_g$. Finalement, on a $m_1, \ldots, m_G$ les moyennes de chaque groupe. En se référant à Tibshirani [**11**], les étapes de l'algorithme des k-moyennes sont les suivantes:

(1) Les valeurs $m_1, \ldots, m_G$ sont fixées soit aléatoirement, soit de façon arbitraire.

(2) Pour une partition $\mathcal{P}$ donnée, on trouve les moyennes $m_1, \ldots, m_G$ qui minimisent la variance totale d'une partition

$$\min_{m_1,\ldots,m_k} \sum_{g=1}^{G} \sum_{i:p(i)=g} n_g \|y_i - m_g\|^2$$

(3) Chaque élément $y_1, \ldots, y_N$ est assigné au sous-ensemble avec la moyenne la plus proche, c'est-à-dire

$$p(i) = \underset{1 \le g \le G}{\operatorname{argmin}} \|y_i - m_g\|^2.$$

Ce regroupement aboutit à une nouvelle partition $\mathcal{P}^*$.

(4) Les étapes 2 et 3 sont répétées avec $\mathcal{P}^*$ jusqu'à ce qu'il n'y ait plus aucun changement entre les moyennes, ou que la différence ne dépasse plus un certain seuil. Si $\mathbf{m} = (m_1, \ldots, m_G)'$ est le vecteur des moyennes et $\mathbf{m}^*$ est celui de la dernière itération, l'algorithme se termine lorsque $\|\mathbf{m} - \mathbf{m}^*\|^2 < \delta$, où δ est un seuil choisi arbitrairement.

Cette méthode a l'avantage de ne pas avoir recours à un ensemble d'entraînement, et d'être facile à programmer. Cependant, il faut connaître la nature des données pour savoir combien de partitions il y a. De plus, elle s'applique seulement aux espaces linéairement séparables. Ce sont des espaces $\mathbb{R}^d$ tels que chaque paire de partitions est séparable par un hyperplan. Dans le cas de $\mathbb{R}^2$, chaque partition doit être séparable par une droite. Il est évident que ce type de partitions n'est pas la norme. Il est possible d'ajuster l'algorithme des K-moyennes pour qu'il soit applicable à des partitions plus complexes.

3.2.3. K-moyennes par noyaux

Le problème du dernier algorithme est la notion de distance, qui est une approche linéaire. Il est possible de changer cette notion de distance, avec l'aide de noyaux. À l'étape 2 du

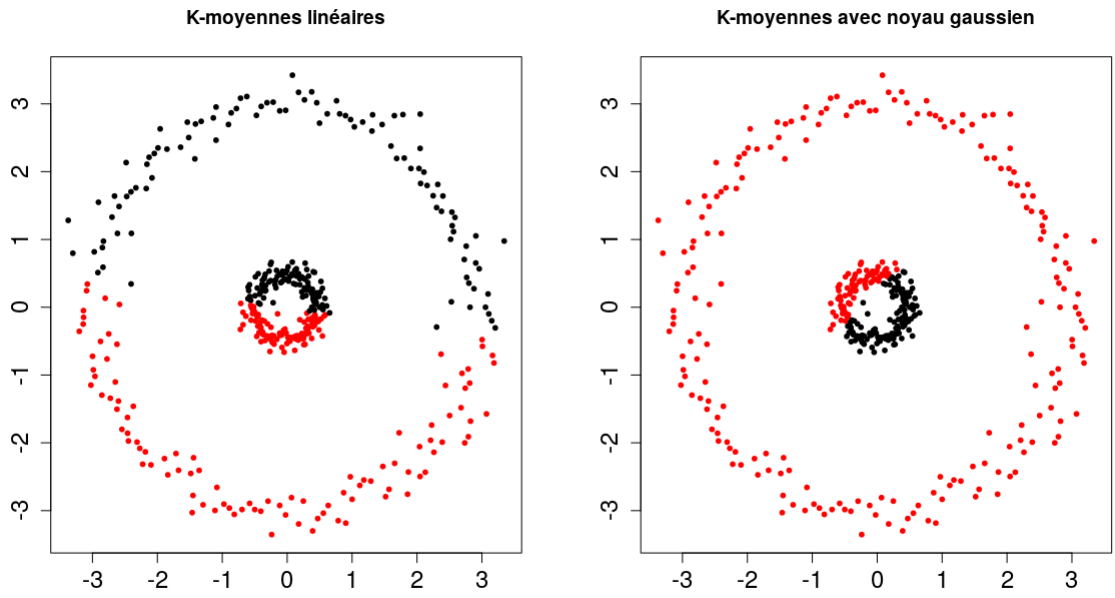

Fig. 3.2. À gauche : Partitionnement avec l'algorithme des K-moyennes. À droite : Partitionnement avec l'algorithme des K-moyennes avec un noyau gaussien.

dernier algorithme, on avait

$$\min_{m_1,\ldots,m_k} = \sum_{g=1}^{G} \sum_{p(i)=g} n_g \|y_i - m_g\|^2.$$

En développant la norme, on trouve que

$$\|y_i - m_g\|^2 = y_i \cdot y_i - 2y_i \cdot m_g + m_g \cdot m_g.$$

Les produits scalaires sont remplacés par ce qu'on appelle une fonction noyau $\kappa(y_i, m_g)$. Ces fonctions attribuent un poids à y_i selon sa distance avec m_i dans la région définie par κ. Il existe plusieurs fonctions noyaux, comme les suivantes :

- Polynômiale : $\kappa(x,y) = (x \cdot y + c)^p$, pour un $c \in \mathbb{R}^+$ et un $p \in \mathbb{N}$
- Gaussienne : $\kappa(x,y) = \frac{1}{\sigma} \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$, pour une variance $\sigma^2 \in \mathbb{R}^+$.

Une fois la fonction noyau choisie, on procède comme avec l'algorithme des K-moyennes, en substituant la norme des étapes 2 et 3 par

$$\|y_i - m_g\|^2 = \kappa(y_i, y_i) - 2\kappa(y_i, m_g) + \kappa(m_g, m_g).$$

La figure 3.2 est un exemple de partitions qui ne sont pas linéairement séparables. Les points ont été générés par simulation. Les algorithmes des K-moyennes, réguliers et avec noyaux, sont appliqués à l'ensemble de données et les résultats sont affichés dans les graphiques 3.2. Les deux classes sont définies par les couleurs noir et rouge. Il est évident

que la méthode par noyaux s'adapte mieux aux partitions, et donne une classification parfaite. La méthode régulière a tenté de séparer l'ensemble par une droite linéaire visible dans le graphique. Comme de fait, cette partition n'est pas adéquate pour cet ensemble de données.

En bref, les méthodes de classification par distance sont efficaces et simples à implémenter. En revanche, les résultats finaux sont une partition et leurs centroïdes respectifs. Ces informations peuvent être insuffisantes selon l'étude, il faudra alors considérer différentes approches pour établir une densité entre les variables. Cette information est accessible lorsqu'on applique une méthode probabiliste.

3.3. Méthodes probabilistes

L'idée principale des méthodes probabilistes est d'établir une série de modèles statistiques et d'évaluer les probabilités qu'une observation provienne d'un de ces modèles. La complexité de cette approche est nettement supérieure à celle par distance puisqu'il faut établir le modèle, estimer ses paramètres et calculer les probabilités. En revanche, il sera possible d'analyser en profondeur la structure fonctionnelle des données pour mieux comprendre, par exemple, les impacts de la reconstruction par expansion de base et de l'ACPF. De plus, on pourra établir des critères de sélection de modèle rigoureux.

Les modèles établis seront en fait des mélanges de densités, car les données changent de comportement selon leur partition. Même s'il existe une infinité de densités, la loi gaussienne est privilégiée et elle est le cœur d'une des méthodes de partition utilisées dans la classification des trajectoires de voitures. Le calcul de la probabilité d'appartenir à un groupe est possible à l'aide d'une structure bayésienne. Or, les vrais paramètres resteront inconnus et seront estimés grâce à un algorithme très populaire, appelé «algorithme EM». Finalement, le meilleur modèle est déterminé à l'aide de certains critères de sélection.

Pour alléger la notation, la densité de Y sera dénotée par $f_Y(y) = f(Y)$. Elle sera réutilisée lorsqu'on parlera de la densité des variables latentes Z, des variables complètes X et des paramètres Θ dans l'approche bayésienne.

3.3.1. Estimation de densités par mélanges de densités

Le but est de concevoir une nouvelle densité $f(Y|\Theta)$ par une somme pondérée de plusieurs densités connues $f(Y|\Theta_g)$. Puisque l'objectif de cet ouvrage est de partitionner des

données, le nombre de densités utilisé pour le mélange est le même que le nombre de partitions G. Cependant, le principe de mélanges de densités n'est pas du tout limité à ce nombre.

Soit $w_1, \ldots, w_G$ des poids positifs attribués à chacune des densités $f_1, \ldots, f_G$ avec les paramètres associés $\Theta_1, \ldots, \Theta_G$ respectivement, alors la densité par mélange devient simplement

$$f(Y|\mathbf{\Theta}) = \sum_{g=1}^{G} w_g f_g\left(Y|\Theta_g\right)$$

où $\mathbf{\Theta} = \{\Theta_1, \ldots, \Theta_G\}$ et $\sum_{g=1}^{G} w_g = 1$. Les poids, pour notre cas, seront la probabilité d'appartenance à une partition $p_{ig} = \mathbb{P}\left(\Theta = \Theta_g | Y = y_i\right)$. Ils assurent que

$$\int_{\mathcal{T}} f(Y|\mathbf{\Theta}) dy = \sum_{g=1}^{G} p_{ig} \int_{\mathcal{T}} f_g\left(Y|\Theta_g\right) dy = 1$$

pour que $f(Y|\mathbf{\Theta})$ soit une densité.

Ces poids seront estimés lors de l'estimation des paramètres avec l'algorithme-EM, et ils permettront d'ajuster le mélange de densité sur l'échantillon de données. En plus, ils donnent l'information sur l'importance des densités lors de la modélisation. Les $f_1, \ldots, f_G$ peuvent être de même loi de probabilité.

Soient μ_g et σ_g^2 les moyennes et variances associées à chaque densité $f_g(Y)$. Alors

$$\mathbb{E}\left[f(Y|\mathbf{\Theta})\right] = \sum_{g=1}^{G} w_g \mathbb{E}\left[f_g\left(Y|\Theta_g\right)\right] = \sum_{g=1}^{G} w_g \mu_g$$

et

$$\begin{aligned}
\mathbb{V}\left[f(Y|\mathbf{\Theta})\right] &= \mathbb{E}\left[f(Y|\mathbf{\Theta})^2\right] - \mathbb{E}\left[f(Y|\mathbf{\Theta})\right]^2 = \sum_{g=1}^{G} w_g \mathbb{E}\left[f_g(Y|\Theta)^2\right] - \left(\sum_{g=1}^{G} w_g \mu_g\right)^2 \\
&= \sum_{g=1}^{G} w_g \left(\mathbb{V}\left[f_g(Y|\Theta)\right] + \mathbb{E}\left[f_g(Y|\Theta)\right]^2\right) - \left(\sum_{g=1}^{G} w_g \mu_g\right)^2 \\
&= \sum_{g=1}^{G} w_g \left(\sigma_g^2 + \mu_g^2\right) - \left(\sum_{g=1}^{G} w_g \mu_g\right)^2 .
\end{aligned}$$

Cette approche s'adapte à n'importe quelle densité. Cependant, le mélange de densités gaussiennes est le plus populaire.

3.3.1.1. *Mélanges gaussiens*

Dans cette situation, il est supposé que les densités des éléments y_i de la partition g sont générés par une distribution gaussienne multivariée $\mathcal{N}(\mu_g,\Sigma_g)$ de dimension d où $\mu_g = \mathbb{E}[y_i]$ et $\Sigma_g = \mathbb{V}[y_i]$, $g = 1,\ldots,G$. Le mélange de densités est

$$f_Y(y|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{g=1}^{G} w_g (2\pi)^{-\frac{d}{2}} |\Sigma_g|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(y-\mu_g)' \Sigma_g^{-1}(y-\mu_g)\right),$$

où $\boldsymbol{\mu} = \{\mu_1,\ldots,\mu_G\}$ et $\boldsymbol{\Sigma} = \{\Sigma_1,\ldots,\Sigma_G\}$.

Le graphique de la figure 3.3 présente l'exemple d'une densité obtenue par le mélange de distributions gaussiennes. La pondération attribuée à chacune des densités est, de gauche à droite, de $1/2, 1/4$ et $1/4$. Le mélange est représenté par la courbe noire, et les trois densités gaussiennes sont représentées par trois couleurs distinctes. Les aires sous les courbes ont aussi été pondérées. On peut voir facilement qu'un poids élevé a une grande influence sur la nouvelle densité obtenue.

Il faudra déterminer ces poids. Ils sont, dans le cas du partitionnement, intrinsèquement liés à la probabilité d'appartenance à un groupe. Ces estimations seront obtenues à l'aide d'une structure bayésienne.

3.3.2. Modèles bayésiens de mélange gaussien

Le fameux théorème de Bayes s'applique aisément lorsqu'on désire faire de la classification probabiliste. D'abord, on suppose à nouveau que les G partitions sont chacune déterminées par une distribution $f(Y|\Theta_g)$ de paramètre Θ_g. Au lieu de déterminer la probabilité $\mathbb{P}(Y|\Theta_g)$, il est préférable de calculer $f(\Theta_g|Y)$ à l'aide du théorème de Bayes. Pour y arriver, on a que

$$f(\Theta_g|Y) = \frac{f(Y|\Theta_g)f(\Theta_g)}{f(Y)},$$

où $f(\Theta_g)$ est la densité a priori de Θ_g, $f(\Theta_g|Y)$ la densité a posteriori de Θ_g et $f(Y)$ la densité de Y.

La densité proposée pour $f(\Theta_g)$ n'est rien d'autre que la proportion de points dans le groupe g et donc $f(\Theta_g) = {}^{n_g}/_N$, $g = 1,\ldots,G$. La densité a posteriori $f(Y|\Theta_g)$ est équivalente à la vraisemblance $\mathcal{L}(\Theta_g|Y)$. Avec le principe de la probabilité totale, $f(Y) = \sum_{g=1}^{G} f(Y|\Theta_g)f(\Theta_g)$. Alors, la probabilité de Θ_g sachant Y est directement

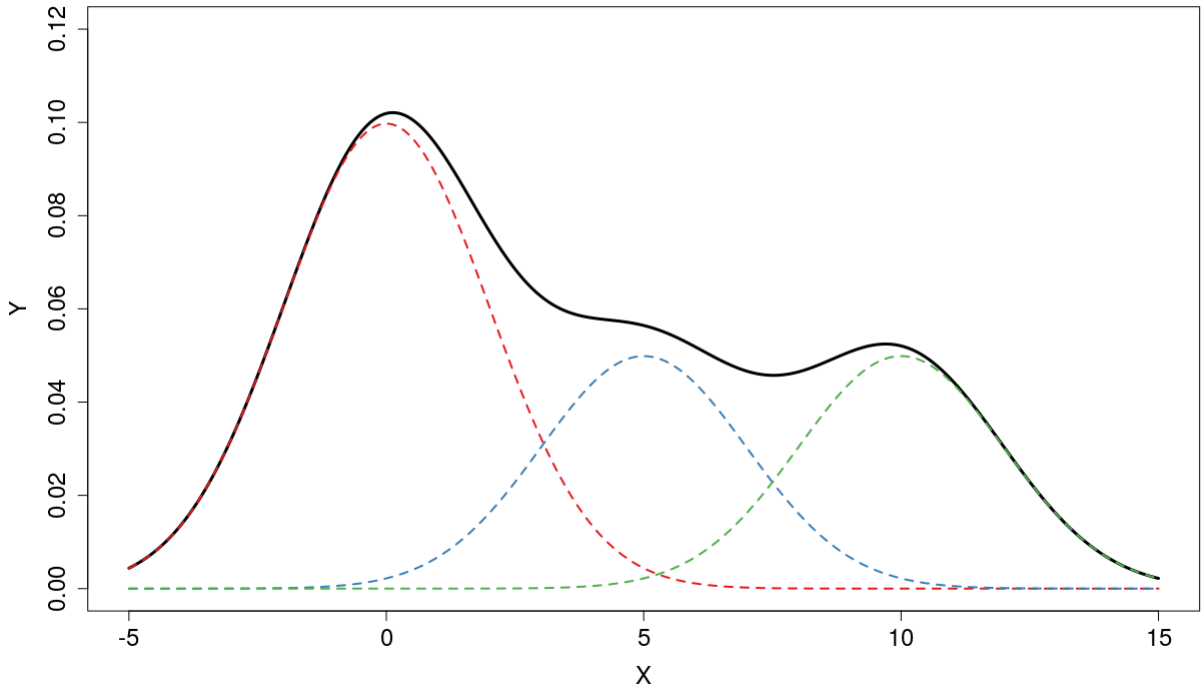

Fig. 3.3. Densité obtenue par le mélange de trois distributions gaussiennes de moyenne 0,5 et 10 et de même variance 2. Les poids et les aires sous la courbe sont $1/2, 1/4$ et $1/4$.

$$f(\Theta_g|Y) = \frac{f(Y|\Theta_g)f(\Theta_g)}{\sum_{l=1}^{G} f(Y|\Theta_l)f(\Theta_l)} = \frac{\mathcal{L}(\Theta_g|Y)\frac{n_g}{N}}{\sum_{l=1}^{G} \mathcal{L}(\Theta_l|Y)\frac{n_l}{N}}.$$

Ainsi, on obtient la probabilité qu'un modèle soit approprié pour une variable aléatoire Y puisque $\mathbb{P}(\Theta_g|Y) = f(\Theta_g|Y)$. Par contre, ce calcul nécessite l'ensemble des paramètres $\boldsymbol{\Theta} = \{\Theta_1, \ldots, \Theta_G\}$ et la taille des partitions $n_1, \ldots, n_G$. Ces valeurs seront estimées à l'aide d'une méthode itérative.

3.3.3. Estimation des paramètres : l'algorithme EM

L'algorithme *EM*, proposé par Dempster, Laird et Rubin dans [**9**], est un outil primordial pour obtenir des estimations par itération. Son nom vient du principe de l'*Expectation-Maximization* en anglais. L'espérance conditionnelle du logarithme de la vraisemblance est calculée (*Expectation*), pour ensuite être maximisée selon leur paramètre (*Maximization*).

La structure des observations reste la même : elles forment un ensemble de paires d'informations $\mathbf{X} = (y_1, z_1), \ldots, (y_N, z_N)$, les premiers termes sont les données observées issues d'une même distribution $f(Y|\Theta)$ et les deuxièmes sont les variables latentes. Les notations suivantes seront utilisées : $\log(\mathcal{L}(\mathbf{Y},\mathbf{Z}|\Theta)) = \ell(\Theta; \mathbf{Y}, \mathbf{Z})$, $\mathbf{Y} = (y_1, \ldots, y_N)'$ et

$\mathbf{Z} = (z_1, \ldots, z_N)'$.

L'algorithme *EM* est composé des étapes suivantes :

1. En commençant avec $j = 0$, choisir une valeur initiale $\hat{\Theta}^{(j)}$ pour les paramètres Θ du modèle. Le choix peut être arbitraire.
2. Calculer l'espérance conditionnelle du logarithme de la vraisemblance sachant $\mathbf{Y}$ et $\hat{\Theta}^{(j)}$, i.e. $Q(\Theta,\hat{\Theta}^{(j)}) = \mathbb{E}\left[\ell(\Theta;\mathbf{X}|\mathbf{Y},\hat{\Theta}^{(j)})\right]$.
3. Déterminer la nouvelle estimation $\hat{\Theta}^{(j+1)}$ qui maximise la fonction $Q(\Theta|\hat{\Theta}^{(j)})$ sur Θ.
4. Répéter les étapes 2 et 3 avec la nouvelle estimation $\hat{\Theta}^{(j+1)}$ jusqu'à la convergence de celle-ci.

Cette approche permet de maximiser le logarithme de la vraisemblance complète, en plus d'assurer une convergence des estimateurs obtenue par itération. La raison, démontrée dans [**19**], en est la suivante. Par le théorème de Bayes,

$$\mathbb{P}\left(\mathbf{Y}|\Theta\right) = \frac{\mathbb{P}\left(\mathbf{Y},\mathbf{Z}|\Theta\right)}{\mathbb{P}\left(\mathbf{Z}|\mathbf{Y},\Theta\right)}.$$

La même chose peut être dite pour $f(\mathbf{Y}|\Theta)$. Les vraisemblances résultantes sont

$$\begin{aligned}\mathcal{L}\left(\Theta;\mathbf{Y}\right) &= \prod_{i=1}^{N} f_Y\left(y_i|\Theta\right) = \prod_{i=1}^{N} \frac{f_Y\left(y_i,z_i|\Theta\right)}{f_Z\left(z_i|Y,\Theta\right)} \\ \Rightarrow \ell\left(\Theta;\mathbf{Y}\right) &= \sum_{i=1}^{N} \log f_Y\left(y_i,z_i|\Theta\right) - \sum_{i=1}^{N} \log f_Z\left(z_i|Y,\Theta\right) = \ell(\Theta;\mathbf{Y},\mathbf{Z}) - \ell(\Theta;\mathbf{Z}|\mathbf{Y}).\end{aligned}$$

Ensuite, on calcule l'espérance conditionnelle de $\mathbf{Z}$ sachant $\mathbf{Y}$ et l'estimation $\hat{\Theta}$ sur la dernière équation. Puisque $\ell\left(\Theta;\mathbf{Y}\right)$ est constante selon $\mathbf{Y}$ et qu'elle n'a aucun terme en $\hat{\Theta}$, elle n'est pas affectée par l'espérance conditionnelle et on obtient

$$\begin{aligned}\ell(\Theta;\mathbf{Y}) &= \mathbb{E}_Z\left[\ell(\Theta;\mathbf{Y},\mathbf{Z})|\mathbf{Y},\hat{\Theta}\right] - \mathbb{E}_Z\left[\ell(\Theta;\mathbf{Z}|\mathbf{Y})|\mathbf{Y},\hat{\Theta}\right] \\ &= Q(\Theta|\hat{\Theta}) - R(\Theta|\hat{\Theta}) \qquad (3.3.1)\end{aligned}$$

avec $R(\Theta|\hat{\Theta}) = \mathbb{E}_Z\left[\ell(\Theta;\mathbf{Z}|\mathbf{Y})|\mathbf{Y},\hat{\Theta}\right]$. L'équation (3.3.1) reste valide pour toute valeur de Θ, de même que pour $\Theta = \hat{\Theta}$. Alors l'affirmation suivante reste vraie:

$$\ell(\hat{\Theta};\mathbf{Y}) = Q(\hat{\Theta}|\hat{\Theta}) - R(\hat{\Theta}|\hat{\Theta}).$$

En soustrayant cette dernière à l'équation (3.3.1), on trouve que

$$\ell(\Theta;\mathbf{Y}) - \ell(\hat{\Theta};\mathbf{Y}) = Q(\hat{\Theta}|\hat{\Theta}) - Q(\Theta|\hat{\Theta}) + R(\Theta|\hat{\Theta}) - R(\hat{\Theta}|\hat{\Theta}).$$

Le logarithme de la vraisemblance est une fonction concave, on peut donc appliquer l'inégalité de Jensen [**16**] sur la fonction R et la borner par $R(\hat{\Theta}|\hat{\Theta}) \geq R(\Theta|\hat{\Theta})$. On obtient donc

$$\ell(\Theta; \mathbf{Y}) - \ell(\hat{\Theta}; \mathbf{Y}) \geq Q(\Theta|\hat{\Theta}) - Q(\hat{\Theta}|\hat{\Theta})$$

Ceci démontre que lorsque Θ améliore la quantité $Q(\Theta|\hat{\Theta})$, $\ell(\Theta; \mathbf{Y})$ augmentera d'au moins autant. Puisque le logarithme de la vraisemblance est une fonction concave définie sur un intervalle fermé et borné, elle a une borne supérieure, sauf dans le cas extrême où la densité n'est pas bornée. Par ce fait, l'algorithme convergera presque toujours.

Cet algorithme va permettre d'obtenir les estimations des modèles de partitionnement probabilistes, puisqu'ils utilisent le mélange de densité et les variables latentes. Avec l'approche bayésienne établie plus tôt, on est en mesure d'obtenir l'estimation de $\hat{\boldsymbol{\Theta}} = \left\{\hat{\Theta}_1, \ldots, \hat{\Theta}_G\right\}$. Cependant, ceci ne permet pas de déterminer les hyper-paramètres comme la dimension de l'espace des fonctions de base, un facteur qui affecte les estimations des paramètres. Il faudra faire appel aux critères de sélection pour ce problème.

L'algorithme EM pour des mélanges gaussiens est implémenté dans le logiciel **R** dans le module *mclust* [**10**].

3.3.4. Sélection d'un modèle : les critères d'information bayésien et d'Akaike

Les critères d'information attribuent une valeur à l'exactitude d'un modèle statistique dans son ensemble. Ceci est pratique lorsqu'on souhaite comparer des modèles qui diffèrent dans le nombre de paramètres ou tout simplement dans leur structure.

On suppose qu'on doit faire un choix entre les G modèles $\mathcal{M}_1, \ldots, \mathcal{M}_G$ où $G \in \mathbb{N}$. Ici, on reprend l'indice G car on doit faire un choix entre plusieurs tailles de partition. Or, le choix de modèle n'est pas limité à cette situation. L'idée de base est d'avoir un critère d'information qui trouve le modèle $\mathcal{M}_g$ avec la plus grande probabilité $\mathbb{P}(\mathcal{M}_g|\mathbf{Y})$. Avec le théorème de Bayes, $\mathbb{P}(\mathcal{M}_g|\mathbf{Y}) \propto \mathbb{P}(\mathbf{Y}|\mathcal{M}_g)\,\mathbb{P}(\mathcal{M}_g)$, où $\mathbb{P}(\mathcal{M}_g)$ est la probabilité a priori du modèle $\mathcal{M}_g$. Maintenant, on peut calculer la probabilité $\mathbb{P}(\mathbf{Y}|\mathcal{M}_g)$ par

$$\mathbb{P}(\mathbf{Y}|\mathcal{M}_g) = \int \mathbb{P}(\mathbf{Y}|\Theta_g, \mathcal{M}_g)\,\mathbb{P}(\Theta_g|\mathcal{M}_g)\,d\Theta_g.$$

Cette dernière quantité s'appelle la *vraisemblance intégrée* et notre critère d'information va se baser sur son approximation avec l'estimation du paramètre $\hat{\Theta}_g$. On souhaite maximiser la vraisemblance intégrée, puisqu'elle est associée à une probabilité $\mathbb{P}(\mathcal{M}_j|\mathbf{Y})$ élevée. On va donc tester pour chaque modèle $\mathcal{M}_1, \ldots, \mathcal{M}_G$.

Schwarz [**26**] propose l'approximation de la vraisemblance intégrée suivante :

$$2\log\mathbb{P}\left(\mathbf{Y}|\mathcal{M}_j\right) \approx 2\log\mathbb{P}\left(\mathbf{Y}|\hat{\Theta}_g, \mathcal{M}_g\right) - d_g\log(N).$$

Cette approximation s'appelle le *Bayesian Information Criterion* (BIC) et d_g est le nombre de paramètres à estimer par le modèle $\mathcal{M}_g$. C'est le maximum du logarithme de la vraisemblance pénalisé par le nombre de paramètres. On retient le modèle $\mathcal{M}_g$ associé au BIC le plus élevé. Certaines études ont démontré que ce critère d'information est approprié lors du choix du nombre de partitions [**18**]. Il faut remarquer que puisqu'on tente de maximiser la probabilité $\mathbb{P}\left(\mathcal{M}_g|\mathbf{Y}\right)$, qui est bornée par 1, alors les BIC tentent de trouver le «vrai» modèle empirique. Ceci n'est pas une tâche réaliste, mais permet néanmoins de sélectionner un modèle qui soit proche de la réalité. Comme dit le dicton en statistiques : «Tout les modèles sont faux, mais certains sont utiles. »[**4**].

Un autre critère populaire est le *Akaike Information Criterion* (AIC) [**2**]. Dans sa forme il est très similaire au BIC. Il est défini par

$$AIC = 2\log\mathbb{P}\left(\mathbf{Y}|\hat{\Theta}_g, \mathcal{M}_g\right) - 2d_g.$$

Son objectif est de maximiser le logarithme de la vraisemblance en le pénalisant par le nombre de paramètres. La motivation derrière sa conception est que le principe que l'estimateur de vraisemblance est asymptotiquement efficace. Akaike explique que la pénalité permet d'éviter les modèles qui surestiment ou sous-estiment les données. Dans le cas de la surestimation, le modèle est plus complexe, alors la pénalité due au nombre de paramètres sera plus sévère. Sinon, une sous-estimation implique un logarithme de la vraisemblance plus faible. Par sa construction, le AIC tente de choisir le meilleur modèle parmi $\mathcal{M}_1, \ldots, \mathcal{M}_G$.

Lors du partitionnement fonctionnel, on analysera les AIC et BIC de chacun et on retiendra celui qui correspond au meilleur BIC ou AIC. Ces modèles seront ensuite être utilisés pour l'étape de prédiction.

Ceci conclut non seulement ce chapitre sur le partitionnement probabiliste, mais aussi toutes les définitions nécessaires pour discuter des méthodes de partitionnement fonctionnelles. On est à présent en mesure d'approcher ces données et de les regrouper en sous-groupes homogènes. Effectivement, les techniques qui seront discutées utilisent aussi les outils fonctionnels.

Chapitre 4

Partitionnement des données fonctionnelles

Les dernières années ont démontré une grande avancée dans le domaine de la classification des données fonctionnelles. La majorité des méthodes présentées dans ce mémoire ont été développées dans le nouveau millénaire. Jacques et Preda ont fait une excellente revue littéraire à ce sujet [**13**] où ils comparent les différentes approches. C'est ce que nous allons faire ici même, en discutant de la structure théorique des modèles.

On commencera par un modèle basé sur les distances, **distclust** (*distance-based clustering* [**22**]), et sur le mélange de densités gaussiennes **mclust** (*model-based clustering* [**10**]). Les méthodes fonctionnelles **fitfclust** (*clustering for sparsely sampled functional data* [**15**]), **funmbclust** (*Bayesian model-based functional clustering* [**1**]), **funclust** (*model-based clustering for functional data* [**14**]) et **funHDDC** (*functional high-dimensional data clustering* [**3**]).

4.1. Méthode basée sur des distances

4.1.1. distclust

Proposé par Peng et Müller [**22**], la méthode *Distclust* utilise l'algorithme des K-moyennes adapté aux données fonctionnelles. Leur motivation première est de trouver une notion de distance applicable aux données longitudinales irrégulières, voire incomplètes. Avec cette distance, ils procèdent avec l'algorithme des K-moyennes sur une dimension réduite.

Des données longitudinales sont dites irrégulières si les temps de mesure ne sont pas les mêmes pour chaque observation. En d'autres mots, les vecteurs de temps t_i et t_j des individus y_i et y_j ne sont pas égaux et peuvent être de dimension différente. Il est possible de traiter l'information pour qu'elle soit régulière à l'aide d'interpolation, cependant il n'est pas toujours possible de le faire lorsque le nombre de mesures est très

petit pour certaines courbes. Il faut donc trouver une notion de distance adaptée pour ce cas.

D'abord, il faut définir la structure des données. Il est possible d'exprimer les courbes par une combinaison linéaire de fonctions de base. On suppose que $Y_i(t) = X_i(t) + \varepsilon_i(t)$ tel que $X_i(t)$ est la fonction issue du modèle de la courbe $Y_i(t)$ et $\varepsilon_i(t)$ est la variable aléatoire du terme d'erreur. Puisque ces courbes sont issues d'un processus stochastique résidant dans un espace L^2, supposition faite dans le chapitre 2, il existe un noyau semi-défini positif $C(\cdot,\cdot)$ de sorte que $Cov(X(s), X(t)) = C(s,t)$. Il est donc possible d'exprimer $X(t)$ comme une combinaison linéaire de fonctions propres comme suit:

$$X_i(t) = \mu(t) + \sum_{k=1}^{\infty} \lambda_{ik}\varphi_k(t),\ i = 1\ldots, N.$$

De cette égalité, on a $\mu(t) = \mathbb{E}(X_i(t))$ la fonction moyenne, les valeurs propres $\lambda_{i1} \geq \lambda_{i2} \geq \ldots \geq 0$ de $C(\cdot,\cdot)$ et φ_k les fonctions propres orthonormées. Puisque ces fonctions forment une base orthonormée de l'espace L_2 et que celui-ci est un espace de Hilbert, on peut appliquer l'identité de Parseval (voir A.1) sur la distance habituelle entre deux courbes aléatoires $X_i(t)$ et $X_j(t)$ pour obtenir

$$D(i,j) = \left(\int_0^T \left(X_i(t) - X_j(t)\right)^2 dt\right)^{\frac{1}{2}} = \left(\sum_{k=1}^{\infty} \left(\lambda_{ik} - \lambda_{jk}\right)^2\right)^{\frac{1}{2}}$$

Comme de fait, ce calcul nécessite une connaissance complète des données X_i et X_j sur l'intervalle $t \in [0, T]$. Une distance alternative est possible en prenant la distance D et en la conditionnant sur l'information disponible y_i et y_j.

$$\tilde{D}(i,j) = \left(\mathbb{E}\left(D^2(i,j)|y_i, y_j\right)\right)^{\frac{1}{2}} = \left(\mathbb{E}\left(\sum_{k=1}^{\infty}(\lambda_{ik} - \lambda_{jk})^2|y_i, y_j\right)\right)^{\frac{1}{2}},\ i,j = 1,\ldots, N. \quad (4.1.1)$$

L'article de Peng et Müller [**22**] démontre que cette distance est une métrique dans un espace de variables aléatoires suivant un processus stochastique. Cependant, il n'est pas possible d'utiliser l'infinité de fonctions qui se retrouvent dans la base de l'espace fonctionnel. Une solution simple est de garder les K premières fonctions propres. La distance tronquée est donnée par

$$\begin{aligned}\tilde{D}^{(K)}(i,j) &= \left(\mathbb{E}\left(\sum_{k=1}^{K}(\lambda_{ik} - \lambda_{jk})^2|y_i, y_j\right)\right)^{\frac{1}{2}} \\ &= \left(\sum_{k=1}^{K} \mathbb{V}(\lambda_{ik}|y_i) + \mathbb{V}(\lambda_{jk}|y_j) + \left(\mathbb{E}(\lambda_{ik}|y_i) - \mathbb{E}(\lambda_{jk}|y_j)\right)^2\right)^{\frac{1}{2}} \quad (4.1.2)\end{aligned}$$

Les espérances et les variances de l'équation ci-dessus restent à calculer pour obtenir la distance. On suppose que les termes y_i et $\lambda_i = (\lambda_{i1}, \ldots, \lambda_{iK})'$ suivent des distributions normales $\mathcal{N}(0,\Sigma_i)$ et $\mathcal{N}(0,\Lambda_i)$. Avec la matrice des fonctions propres $\boldsymbol{\varphi_i}$, le travail dans [**22**]

montre que l'équation 4.1.2 devient

$$\left(\tilde{D}^{(K)}(i,j)\right)^2 = tr\left(\Lambda - \Lambda\boldsymbol{\varphi}_i'\Sigma_i^{-1}\boldsymbol{\varphi}_i\Lambda\right) + tr\left(\Lambda - \Lambda\boldsymbol{\varphi}_j'\Sigma_j^{-1}\boldsymbol{\varphi}_j\Lambda\right) + \|\Lambda\boldsymbol{\varphi}_i'\Sigma_i^{-1}(y_i - \mu_i) - \Lambda\boldsymbol{\varphi}_j'\Sigma_j^{-1}(y_j - \mu_j)\|_2^2. \quad (4.1.3)$$

Même si cette distance est applicable, les valeurs μ_i et Σ_i restent inconnues pour tout $i = 1, \ldots, N$ ainsi que Λ et $\boldsymbol{\varphi}_i$. La technique utilisée passe par des opérateurs de lissage sur l'ensemble des données observées. Les détails sont présentés dans l'article de Yao, Müller et Wang [**30**]. Ils supposent que la moyenne, la matrice de covariance et les fonctions propres possèdent des dérivées secondes continues. Cette régression permet de prendre l'information disponible pour une courbe, d'estimer une courbe locale de dimension 1 pour l'estimation de la fonction moyenne et un plan local de dimension 2 pour l'estimation du noyau de la covariance. Avec ces opérateurs de lissages, on peut maintenant obtenir les estimateurs $\hat{\mu}$ et $\hat{C}(\cdot,\cdot)$. Les estimateurs des valeurs et fonctions propres sont calculés en trouvant la solution de l'équation propre

$$\int_0^T \hat{C}(s,t)\hat{\varphi}_k(s)ds = \hat{\lambda}_k\hat{\varphi}_k(t).$$

Avec $\hat{\Lambda} = diag(\hat{\lambda}_1, \ldots, \hat{\lambda}_K)$ et $\hat{\boldsymbol{\varphi}}_i = (\hat{\boldsymbol{\varphi}}_{i1}, \ldots, \hat{\boldsymbol{\varphi}}_{iK})$, l'estimation de la distance tronquée est alors

$$\left(\hat{D}^{(K)}(i,j)\right)^2 = tr\left(\hat{\Lambda} - \hat{\Lambda}\hat{\boldsymbol{\varphi}}_i'\hat{\Sigma}_i^{-1}\hat{\boldsymbol{\varphi}}_i\hat{\Lambda}\right) + tr\left(\hat{\Lambda} - \hat{\Lambda}\hat{\boldsymbol{\varphi}}_j'\hat{\Sigma}_j^{-1}\hat{\boldsymbol{\varphi}}_j\hat{\Lambda}\right) + \|\hat{\Lambda}\hat{\boldsymbol{\varphi}}_i'\hat{\Sigma}_i^{-1}(Y_i - \hat{\mu}_i) - \hat{\Lambda}\hat{\boldsymbol{\varphi}}_j'\hat{\Sigma}_j^{-1}(Y_j - \hat{\mu}_j)\|_2^2. \quad (4.1.4)$$

En raison de la grande dimension des données fonctionnelles, une réduction de la dimension de l'espace sera nécessaire pour faciliter les calculs numériques et l'interprétation visuelle. Malgré que l'ACP fonctionnelle soit un outil de prédilection pour ce type de problème, l'approche *distclust* utilisera plutôt un positionnement multidimensionnel (*multi-dimentional scaling* en anglais). Le but de cette méthode est de retrouver les coordonnées d'un ensemble d'objets (dans notre cas ce sont les courbes fonctionnelles) dans un espace à p-dimension pour que les distances dans l'espace projeté, notées par d_{ij}, correspondent aux distances originales notées par δ_{ij} selon un critère de sélection. Si on veut faire de l'interprétation visuelle, on prend une dimension $p = 2$ ou 3. Dans les écrits, les δ_{ij} sont aussi connues sous le nom de *termes de dissemblance*. La méthode *distclust* utilise un positionnement multidimensionnel métrique qui traite-les δ_{ij} directement par une distance Euclidienne. Il y a une équivalence entre l'ACP et le positionnement multidimensionnel lorsque les distances δ_{ij} sont réellement des distances euclidiennes. Dans notre situation, la distance originale sera l'estimation de la distance conditionnelle L_2 qui est $\hat{D}^{(K)}$ dans

l'équation (4.1.4).

Les nouvelles distances entre les objets dans la projection doivent minimiser une fonction de perte. Deux fonctions populaires sont *stress* et *s-stress*, qui prennent la forme

$$\sum_{i \neq j} w_{ij} \left((d_{ij}^2)^r - (\delta_{ij}^2)^r \right)^2,$$

couramment utilisé avec $w_{ij} = 1$. La fonction *stress* correspond au cas $r = 1/2$ proposé par Kruskal [**17**], et *s-stress* correspond au cas $r = 1$ qui a été proposé par Takane, Young et DeLeeuw [**28**]. Le second cas entraîne une minimisation plus lisse que le premier et c'est le critère implémenté dans *distclust*. Finalement, c'est dans cet espace à dimension réduite que l'algorithme des k-moyennes est utilisé pour déterminer les partitions. Le nombre de partitions G est estimé avec les critères de sélection AIC et BIC comme mentionnés dans la section 3.3.4.

La méthode *distclust* apporte une notion de distance applicable aux courbes irrégulières. Malgré que ça ne soit pas notre cas ici, cela permet de pallier le problème de données manquantes dans la cueillette d'information. Le positionnement multidimensionnel permet une interprétation visuelle des données, mais cet algorithme de classification par distance procure peu d'informations sur leur structure probabiliste des données. On va donc considérer des algorithmes qui en procurent davantage, en commençant avec *mclust*.

4.2. Méthodes générales basées sur des mélanges de densités

4.2.1. mclust

Pour faire le pont entre la méthode par distance et celle par modélisation fonctionnelle, on discutera de celle par mélange de densités gaussiennes. Cette approche offrira un modèle statistique, mais celui n'est pas aussi riche que ce qu'on peut obtenir avec les méthodes qui vont suivre après. On commence avec le partitionnement par mélange de densités qui est intuitif, sensiblement simple à implémenter et applicable à plusieurs structures de données différentes.

Avec ce qui a été dit dans la section 3.3.1, on sait que chaque donnée y_i est une réalisation d'une variable aléatoire Y_i qui suit une mixture de densités gaussiennes $f_{Y_i}(y_i) = \sum_{g=1}^{G} w_g f_{Y_i}^g(y_i)$. On va maintenant y appliquer les recommandations de Fraley et Raftery [**10**].

D'abord, les poids w_g seront les probabilités qu'une observation quelconque fasse partie de la partition P_g qu'on notera par π_g. Ensuite, les densités caractérisent les vecteurs des mesures observées en chaque temps $t_1, \ldots, t_n$. Autrement dit, chaque courbe observée sera traitée comme un n-vecteur $y_i(t) = (y_i(t_1), \ldots, y_i(t_n))'$. Les densités $f^g_{Y_i}$ sont modélisées par des densités gaussiennes multivariées de moyenne μ_g et avec une matrice de covariance Σ_g. Ainsi, la vraisemblance des données avec N courbes $\mathbf{Y} = (y_1, \ldots, y_N)$ est

$$\mathcal{L}\left(\mu_1, \ldots, \mu_G, \Sigma_1, \ldots, \Sigma_G | \mathbf{Y}\right) = \prod_{i=1}^{N} \sum_{g=1}^{G} \pi_g \left(2\pi\right)^{-\frac{n}{2}} \left|\Sigma_g\right|^{-\frac{1}{2}} \exp\left(\frac{1}{2}\left(y_i - \mu_g\right)' \Sigma_g^{-1} \left(y_i - \mu_g\right)\right) \tag{4.2.1}$$

Avec le nombre de partitions fixé, les paramètres du modèle peuvent être estimés par un algorithme EM.

La densité de chacun des groupes est principalement caractérisée par les moyennes μ_g qui définissent leurs centres. La région autour de ce centre sera déterminée par la forme de la matrice de covariance Σ_g. Cette région peut être constante au travers des groupes, et elle peut prendre une forme sphérique ou ellipsoïdale. Tout va dépendre de la modélisation de Σ_g.

Les deux premières paramétrisations, les plus simples et souvent utilisées pour leurs courts temps de calcul, sont $\Sigma_g = \lambda I$ et $\Sigma_g = \lambda_g I$. Ces paramétrisations ne supposent aucune covariance entre les composantes des vecteurs y_i. La première est constante entre chaque groupe, i.e. $\Sigma_g = \Sigma$ pour $g = 1, \ldots, G$ ce qui n'est pas toujours souhaitable. C'est pour cela que la deuxième, qui permet d'obtenir des matrices de covariance distinctes pour chaque groupe tout en préservant une variance commune entre toutes les composantes, suppose que celles-ci sont indépendantes entre elles. Il est possible de complexifier la paramétrisation de Σ_g, mais il y aura plus de paramètres à estimer.

Il n'est pas surprenant qu'il existe des modèles plus complexes et plus riches. Le travail de Fraley et Raftery [**10**] propose une structure générale pour la paramétrisation des matrices de covariance dans le cas d'une densité de mixture gaussienne. Ils utilisent la décomposition matricielle par valeurs propres, ce qui donne l'expression suivante :

$$\Sigma_g = \lambda_g Q_g D_g Q_g'. \tag{4.2.2}$$

Ici, Q_g est une matrice orthogonale formée des vecteurs propres de Σ_g, D_g est une matrice diagonale contenant des valeurs proportionnelles aux valeurs propres de Σ_g, et λ_g est le vecteur constant de proportionnalité. Chacune de ces composantes sera calculée indépendamment. Elles peuvent être fixes entre chaque groupe, ou non. De plus, chacun de des paramètres suivant détermine un aspect de la forme de la densité $f^g_{Y_i}$: D_g son orientation,

Tableau 4.1. Liste des différentes paramétrisations de la matrice de covariance

Étiquette	Modèle	Nombre de paramètres	Forme de la Distribution
EII	λI	1	Sphérique
VII	$\lambda_g I$	G	Sphérique
EEI	λD	n	Diagonale
VEI	$\lambda_g D$	$G + (n-1)$	Diagonale
EVI	λD_g	$1 + G(n-1)$	Diagonale
VVI	$\lambda_g D_g$	Gn	Diagonale
EEE	$\lambda PDP'$	$n(n+1)/2$	Ellipsoïdale
EEV	$\lambda P_g D P_g'$	$1 + (n-1) + G(n(n-1)/2)$	Ellipsoïdale
VEV	$\lambda_g P_g D P_g'$	$G + (n-1) + G(n(n-1)/2)$	Ellipsoïdale
VVV	$\lambda_g P_g D_g P_g'$	$G(n(n+1)/2)$	Ellipsoïdale

Q_g sa forme et λ_g son volume.

La librairie **mclust** dans **R** permet de tester les différentes paramétrisations possibles. On les retrouvent dans le tableau 4.1. Les différentes formes possibles de Σ_g sont caractérisées par la forme géométrique de la densité. Le programme teste les différentes paramétrisations possibles et retient celui avec un AIC et un BIC optimaux. Chaque paramétrisation de Σ_g est étiquetée par un code à trois lettres, qui représentent le volume (λ_g), la forme (Q_g) et l'orientation (D_g) de la densité. Ces caractéristiques peuvent soit être égales entre les groupes (E), soit variées entre les groupes (V) dans lesquels la matrice de covariance peut être un multiple de l'identité (I).

Comme il a été mentionné, les paramètres seront estimés par l'algorithme EM. Il traite chaque courbe observée comme étant une paire d'informations (y_i, z_i). Cette fois-ci, la variable latente z_i sera un vecteur de variable indicatrice $(z_{i1}, \ldots, z_{iG})'$ tel que

$$z_{ig} = \begin{cases} 1 & \text{si } y_i \in P_g \\ 0 & \text{sinon} \end{cases}$$

On commence l'algorithme EM en fixant des valeurs arbitraires aux paramètres à estimer qui sont $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_G\}$, $\boldsymbol{\Sigma} = \{\Sigma_1, \ldots, \Sigma_G\}$, $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_G\}$ et $\mathbf{z} = \{z_{11}, \ldots, z_{1G}, z_{21}, \ldots, z_{N1}, \ldots, z_{NG}\}$. On rajoute que les valeurs $z_1, \ldots, z_N$ sont la réalisation des variables aléatoires $Z_1, \ldots, Z_N$ qui sont identiquement et indépendamment distribuées selon une densité multinomiale $Multi(\pi_1, \ldots, \pi_G)$. En réécrivant la densité sous la forme $f_Y^g(y_i|z_i) = \prod_{g=1}^G f_{Y_i}^g(y_i|\mu_g, \Sigma_g)^{z_{ig}}$, le logarithme de la vraisemblance avec les estimations à la

$j^{ième}$ itération est

$$\ell\left(\mathbf{Y}, \mathbf{Z}; \boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}^{(j)}, \boldsymbol{\pi}^{(j)}, \mathbf{z}^{(j)}\right) =$$
$$\sum_{i=1}^{N}\sum_{g=1}^{G} z_{ig}\left(\log \pi_g^{(j)} - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma_g^{(j)}|) + \frac{1}{2}(y_i - \mu_g^{(j)})'(\Sigma_g^{-1})^{(j)}(y_i - \mu_g^{(j)})\right) \quad (4.2.3)$$

En calculant l'espérance conditionnelle $\mathbb{E}\left[\ell\left(\mu_g, \Sigma_g, \pi_g, z_{ig}; \mathbf{Y}, \mathbf{Z}\right)|\mathbf{Y}, \boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}^{(j)}, \boldsymbol{\pi}^{(j)}, \mathbf{z}^{(j)}\right]$, on trouve que

$$\hat{z}_{ig} = \frac{\pi_g^{(j)} f_{Y_i}^g(y_i|\mu_g^{(j)}, \Sigma_g^{(j)})}{\sum_{l=1}^{G} \pi_l^{(j)} f_{Y_i}^l(y_i|\mu_l^{(j)}, \Sigma_l^{(j)})} \quad , \quad \forall\ i = 1, \ldots n \text{ et } \ g = 1, \ldots, G.$$

Par la suite, le logarithme de la vraisemblance complète (4.2.3) est maximisé selon les paramètres π_g, μ_g et Σ_g avec les $\hat{z}_{ig}$ trouvés à la dernière itération de l'étape E. Avec les calculs fait dans [**10**], cette maximisation retourne les valeurs suivantes :

$$\pi_g^{(j+1)} = \frac{n_g}{N}; \quad \mu_g^{(j+1)} = \frac{\sum_{i=1}^{N} \hat{z}_{ig} y_i}{n_g}; \quad n_g^{(j+1)} = \sum_{i=1}^{N} \hat{z}_{ig}.$$

Pour ce qui est du paramètre $\Sigma_g^{(j+1)}$, l'estimateur à l'étape M dépend de sa paramétrisation, ce qui complique les calculs. Le travail de Celeux et Govaert [**6**] traite de cette question. Un détail qui se doit être mentionné est qu'une décomposition en valeurs propres est utilisée pour les modèles ellipsoïdaux. Ceci n'est pas possible si le nombre d'observations N est inférieur au nombre de composantes n. On répète les deux dernières étapes jusqu'à convergence des estimations.

Le nombre de partitions et la paramétrisation forment l'ensemble des hyper-paramètres. Le programme fait les calculs pour tester toutes les combinaisons possibles, et retient le modèle avec un AIC et un BIC optimaux.

Finalement, on pourra classifier de nouvelles données y^* avec les estimations finales $\hat{\pi}_g, \hat{\mu}_g$ et $\hat{\Sigma}_g$, $g = 1, \ldots, G$, à l'aide du calcul de probabilité suivant :

$$\hat{z}_g^* = \frac{\hat{\pi}_g f_Y^g(y_i|\hat{\mu}_g, \hat{\Sigma}_g)}{\sum_{l=1}^{G} \hat{\pi}_l f_Y^l(y_i|\hat{\mu}_l, \hat{\Sigma}_l)}$$

pour chaque g. On attribue le groupe g à la variable y^* associée avec la plus grande probabilité z_g^*.

Encore une fois, la méthode présentée ne permet pas de prendre en compte la structure fonctionnelle des données. Pour cette raison, on passera à des méthodes plus complexes.

4.3. Méthodes fonctionnelles

Ces approches utiliseront les différents concepts fonctionnels élaborés dans les chapitres précédents. Elles utiliseront la reconstruction des données par B-splines, l'analyse en composantes principales et des estimations par l'algorithme EM. À moins que cela soit mentionné autrement, elles utilisent toujours une structure de base commune : les données peuvent être exprimées par une combinaison linéaire de fonctions de base telles que

$$\mathbf{Y} = \mathbf{C}\boldsymbol{\phi}(t) + \boldsymbol{\varepsilon}(t)$$

où $\mathbf{Y}(t)$ est la matrice des observations, C celle des coefficients, $\boldsymbol{\phi}(t)$ est le vecteur des fonctions de base et $\boldsymbol{\varepsilon}(t)$ est celui des termes d'erreur. Certaines méthodes utiliseront la formulation $\mathbf{Y} = \boldsymbol{\phi}(t)\mathbf{C} + \boldsymbol{\varepsilon}(t)$, où $\boldsymbol{\phi}(t) = (\phi_1(t), \ldots, \phi_q)$. Pour une seule observation, on a $Y_i(t) = c_i\boldsymbol{\phi}(t) + \varepsilon_i(t) = \sum_{j=1}^{q} c_{ij}\phi_j(t) + \varepsilon_i(t)$, où $c_i(t) = (c_{i1}, \ldots, c_{iq})$, $i = 1, \ldots, N$ et $\boldsymbol{\phi}(t) = (\phi_1(t), \ldots, \phi_q(t))$. Les fonctions $\phi_1(t), \ldots \phi_q(t)$ et $\varepsilon(t)$ sont des n-vecteurs de fonctions évaluées aux temps de mesures $t_1, \ldots, t_n$.

Les modèles se différencient principalement selon l'objet analysé. Les méthodes *fitfclust* et *funmbclust* se concentrent sur la structure des coefficients dans $\mathbf{C}$, alors que *funclust* et *funHDDC* eux se concentrent sur les scores issus des composantes principales.

4.3.1. fitfclust

James et Sugar [**15**] ont travaillé intensément sur le partitionnement des données fonctionnelles, et leur travail a inspiré plusieurs chercheurs autres par la suite. Ils ont proposé le modèle *fitfclust*, qui a été réutilisé dans le progiciel **R** *funcy* [**31**]. Passant par la modélisation des données, il utilise des effets mixtes pour les coefficients des fonctions de base, ce qui permet de répartir l'information disponible tout au long de la courbe et produit de meilleurs résultats pour les données incomplètes. Contrairement aux autres méthodes par modélisation, celle-ci ne réduit pas les données par ACP fonctionnelle, mais l'article [**15**] propose des outils pour obtenir une représentation des données en faible dimension.

Avec la structure générale $Y_i(t) = c_i\boldsymbol{\phi}(t) + \varepsilon_i(t)$, les variables aléatoires $Y_1, \ldots, Y_N$ sont identiquement et indépendamment distribuées telles que

$$c_i = \theta_\mu + \Theta\alpha_{z_i} + \gamma \text{ avec } \gamma \sim \mathcal{N}(0, \Gamma) \text{ et } \varepsilon_i \sim \mathcal{N}(0, \sigma^2 I)$$

où Z_i est la variable aléatoire représentant la partition de l'observation i. On rappelle que c_i est un q-vecteur de coefficients. La variable Z_i dernière suit une distribution multinomiale de probabilité $\pi_1, \ldots, \pi_G$. θ_μ est un q-vecteur, décrivant la moyenne globale et $\Theta\alpha_g$ est un autre q-vecteur décrivant la moyenne de la partition g avec Θ une matrice $q \times h$ et α_g

un h-vecteur. Ici, α_g est le centre de la partition g dans un espace restreint de dimension $h \leq \min(q, G-1)$, afin d'adapter le modèle aux observations incomplètes. Un effet aléatoire γ y est aussi ajouté, afin de pallier aux problèmes liés aux observations incomplètes. Cette variable est commune à toutes les observations, ce qui permet de répartir l'information au travers des courbes. En d'autres mots, les courbes incomplètes pourront, dans un certain sens, emprunter de l'information d'une autre.

En somme, le modèle de partitionnement fonctionnel mixte est tel que les variables aléatoires $Y_1, \ldots, Y_N$ sont identiquement et indépendamment distribuées, de sorte que

$$Y_i(t) = \boldsymbol{\phi}(t)\left(\theta_\mu + \Theta\alpha_{Z_i} + \gamma\right) + \varepsilon_i(t) \text{ avec } \gamma \sim \mathcal{N}(0, \Gamma) \text{ et } \varepsilon_i(t) \sim \mathcal{N}(0, \sigma^2 I),$$
$$\text{donc } Y_i(t) \sim \mathcal{N}\left(\boldsymbol{\phi}(t)\left(\theta_\mu + \Theta\alpha_{Z_i}\right), \sigma^2 I + \boldsymbol{\phi}(t)\Gamma(\boldsymbol{\phi}(t))'\right) \tag{4.3.1}$$

Pour que le modèle 4.3.1 soit complet, il faudra rajouter des contraintes sur θ_μ, Θ et α_g afin que ceux-ci ne soient pas confondus. Il sera donc nécessaire que

$$\sum_{g=1}^{G} \alpha_g = 0 \quad \text{et} \quad \Theta'(\boldsymbol{\phi}(t))'\left(\sigma^2 I + \boldsymbol{\phi}(t)\Gamma(\boldsymbol{\phi}(t))'\right)^{-1}\boldsymbol{\phi}(t)\Theta = I. \tag{4.3.2}$$

La première contrainte permettra d'interpréter $\theta_\mu\boldsymbol{\phi}(t)$ comme étant la courbe moyenne de l'ensemble des données. La seconde facilitera l'analyse des données et leur représentation dans un espace à basse dimension. Ceci mérite une explication, puisque James et Sugar prouvent dans [**15**] qu'il existe une relation entre la probabilité a posteriori que la $i^{ième}$ courbe se trouve dans la partition g et la distance au carré entre α_g et la projection de y_i sur le sous-espace moyen.

Dans un premier temps, soit $\Sigma = \sigma^2 I + \boldsymbol{\phi}(t)\Gamma(\boldsymbol{\phi}(t))'$ la matrice de covariance de Y_i. Il est possible de projeter y_i sur la base de splines de dimension q et d'obtenir

$$\hat{c}_i = \left((\boldsymbol{\phi}(t))'\Sigma^{-1}\boldsymbol{\phi}(t)\right)^{-1}(\boldsymbol{\phi}(t))'\Sigma y_i.$$

Ensuite, $\hat{c}_i$ est projetée sur l'espace engendré par les courbes moyennes μ_g de dimension h pour ainsi obtenir $\theta_\mu + \Theta\hat{\alpha}_i$ où

$$\hat{\alpha}_i = \left(\Theta'(\boldsymbol{\phi}(t))'\Sigma^{-1}\boldsymbol{\phi}(t)\Theta\right)^{-1}\Theta'(\boldsymbol{\phi}(t))'\Sigma^{-1}\boldsymbol{\phi}(t)\left(\hat{\eta}_i - \theta_\mu\right).$$

Alors, $\hat{\alpha}_i$ est la projection de y_i centrée sur l'espace des courbes moyennes. Il est maintenant possible de présenter le théorème suivant :

Théorème 4.3.1. *Soit y_i provenant de la densité définie-en 4.3.1. Puisque*

$$\log \mathbb{P}(Z_i = g|y_i) = c(y_i) + \log(\pi_g) - \frac{1}{2}\left(\hat{\alpha}_i - \alpha_g\right)' Cov(\hat{\alpha}_i)^{-1}\left(\hat{\alpha}_i - \alpha_g\right)$$

tel que $c(Y_i)$ est une constante, π_g la probabilité qu'une courbe quelconque soit dans la partition P_g et que $Cov(\hat{\alpha}_i) = (\Theta'(\boldsymbol{\phi}(t))'\Sigma^{-1}\boldsymbol{\phi}(t)\Theta)^{-1}$, alors

$$\underset{k}{arg\ max}\ \mathbb{P}(Z_i = g|y_i) = \underset{k}{arg\ min}\left((\hat{\alpha}_i - \alpha_g)'\, Cov(\hat{\alpha}_i)^{-1}\,(\hat{\alpha}_i - \alpha_g) - 2\log(\pi_g)\right)$$

Une preuve de ce théorème peut être retrouvée dans [**15**]. Ce théorème montre qu'il est possible de projeter les y_i dans un espace de dimension inférieure sans perdre de l'information sur leur partitionnement. En effet, la deuxième contrainte de 4.3.2 entraîne que $Cov(\hat{\alpha}_i) = I$, donc

$$\arg\max_g\ \mathbb{P}(Z_i = g|y_i) = \arg\min_g\left(\|\hat{\alpha}_i - \alpha_g\|_2^2 - 2\log(\pi_g)\right)$$

et permet d'utiliser la distance euclidienne pour l'analyse dans le sous-espace.

Pour obtenir l'adéquation du modèle 4.3.1, il faudra trouver une estimation des paramètres $\theta_\mu, \Theta, \Gamma$ et σ^2 de plus que α_g et π_g pour $g = 1, \ldots, G$. Les estimations seront obtenues en maximisant le logarithme de la vraisemblance mixte suivante :

$$\mathcal{L}\left(\theta_\mu, \Theta, \Gamma, \sigma^2, \alpha_1, \ldots, \alpha_G, \pi_1, \ldots, \pi_G | y_1, \ldots, y_N\right) = \prod_{i=1}^{N}\sum_{g=1}^{G}\pi_g f_g\left(Y_i|\ \theta_\mu, \Theta, \Gamma, \sigma^2, \alpha_g\right)$$

à l'aide de l'algorithme EM. Les détails de l'algorithme se retrouvent dans l'appendice de l'article [**15**].

Les hyper-paramètres q et h restent à être déterminés. En commençant avec q, l'algorithme présenté teste plusieurs valeurs de q et retient celle qui correspond aux meilleurs BIC ou aux meilleurs AIC. Finalement, la taille de h est telle que $h \leq min(q, G-1)$, G le nombre de groupes. L'article [**15**] ne suggère pas de procédure concrète pour obtenir un h optimal. Il admet que le choix de h n'a jamais causé de problème dans leur simulation, puisque le nombre de groupes total G était petit. Il propose alors une approche générale qui consiste à calculer le modèle avec $h = G - 1$ et calculer les $\alpha_1, \ldots, \alpha_G$. Lorsque que le calcul est complété, un test est conduit pour déterminer si les centres des partitions se retrouvent dans un plan ayant une dimension plus petite que h. Si c'est le cas, un nouveau modèle est calculé avec h fixé à cette dimension. Même si une ACP permettrait d'obtenir une réponse directement, cette procédure n'est pas utilisée dans la procédure *fitclust*.

Ce premier modèle fonctionnel est simple, comparé aux prochains qui suivent. Il modélise les courbes selon les coefficients des fonctions de base et réussit à pallier aux problèmes des données incomplètes, même si cela n'est pas notre cas ici. En plus, il est possible de représenter les données dans un espace à dimension moindre, et de faire de la discrimination dans celle-ci. Il est toutefois possible de complexifier la structure avec une approche bayésienne. C'est ce que fait le prochain modèle, *funmbclust*.

4.3.2. funmbclust

L'équipe de Murua et Folly a revisité les méthodes de partitionnement par modélisation dans [**1**] et propose l'approche *funmbclust*, qui se veut plus flexible. Cette approche fournit des résultats plus concluants, et le modèle s'adapte bien aux données irrégulières. Le modèle *fitfclust* suppose que les vecteurs des coefficients des splines suivent une distribution gaussienne. Le modèle *funmbclust* lui utilise les scores des composantes principales fonctionnelles. De plus, les termes d'erreurs suivent une distribution t de Student multivariée, contrairement à une gaussienne multivariée comme dans de nombreux écrits. Ceci permet de traiter des distributions avec des ailes plus lourdes. Finalement, une structure bayésienne est appliquée sur l'ensemble des paramètres.

Cette approche utilise une structure différente pour les données. En utilisant le fait que les données observées résident dans l'espace L^2 et proviennent d'une série de variables aléatoires indépendantes $Y_1, \ldots, Y_N$, il est supposé qu'il existe une fonction moyenne $\mu(t)$ et une base de fonction issue de l'espace propre $\varphi_1(t), \ldots, \varphi_k(t)$ telles que

$$Y_i(t) = \mu(t) + \sum_{j=1}^{k} \varphi_j(t)\lambda_{ij} + \varepsilon_i(t) = \mu(t) + \boldsymbol{\varphi}(t)'\boldsymbol{\lambda}_i + \varepsilon_i(t),\ i = 1, \ldots, N. \tag{4.3.3}$$

Les vecteurs $\boldsymbol{\lambda_i} = (\lambda_{i1}, \ldots, \lambda_{ik})$ contiennent les scores des composantes et $\boldsymbol{\varphi}(t) = (\varphi_1(t), \ldots, \varphi_k(t))$ les fonctions propres. Comme de fait, la valeur k est le nombre de composantes principales retenues. Contrairement aux modélisations plus standards, les termes d'erreurs $\varepsilon_i(t)$ ne sont pas directement issus d'une densité gaussienne. Ils suivent plutôt la statistique t de Student avec ν_0 degré de liberté et $\sigma^2 I_{2N}$ comme matrice de position. Pour faciliter l'estimation des paramètres, on traduit cette distribution par la convolution entre une densité gaussienne et une chi-deux inversée :

$$\varepsilon_i \sim \mathcal{N}\left(0, \sigma^2\nu_i I_{n_i}\right),\quad \nu_i \sim Inv - \chi^2(\nu_0),\ i = 1, \ldots, N.$$

Les paramètres $\nu_0, \nu_1, \ldots, \nu_N$ seront estimés par un algorithme EM. Cette densité est semblable à la loi gaussienne, mais elle possède des ailes plus lourdes. La densité sera portée à produire des valeurs plus éloignées de sa moyenne. Ceci permettra de regrouper des courbes qui sont plus distancées de la tendance générale.

Ce modèle de partitionnement admet que les scores des composantes sont des effets mixtes de sorte que $\boldsymbol{\lambda}_i = \mu_{Z_i} + \gamma_i^{Z_i}$, où Z_i est une variable aléatoire et qui prend la valeur de la partition de l'individu i entre $1, \ldots, G$. Les vecteurs μ_{z_i} et $\gamma_i^{z_i}$ correspondent à la courbe moyenne de la partition associée à i et à l'effet de l'individu i dans la partition respectivement. On peut remarquer que μ_{z_i} est équivalent au terme α_{z_i} dans l'approche *fitfclust*.

Les fonctions propres et la fonction moyenne sont, à leur tour, reconstruites à l'aide des B-splines $\boldsymbol{\phi}(t) = (\phi_1(t), \ldots, \phi_q(t))'$. Il est possible de reformuler $\mu(t) = \boldsymbol{\phi}(t)\theta_\mu$ avec $\theta_\mu \in \mathbb{R}^q$ et $\boldsymbol{\varphi}(t)' = \boldsymbol{\phi}(t)\Theta$ avec Θ une matrice réelle de taille $q \times k$. Cette nouvelle paramétrisation transforme l'équation 4.3.3 comme suit:

$$\begin{aligned} Y_i(t) &= \mu(t) + \boldsymbol{\varphi}(t)'\boldsymbol{\lambda}_i + \varepsilon_i(t) \\ &= \mu(t) + \boldsymbol{\varphi}(t)'\mu_{z_i} + \boldsymbol{\varphi}(t)'\gamma_i^{z_i} + \varepsilon_i(t) \\ &= \boldsymbol{\phi}_i(t)\theta_\mu + \boldsymbol{\phi}_i(t)\Theta\mu_{z_i} + \boldsymbol{\phi}_i(t)\Theta\gamma_i^{z_i} + \varepsilon_i(t) \end{aligned} \tag{4.3.4}$$

où $\boldsymbol{\phi}_i(t) = (\phi(t_{i1}), \ldots, \phi_i(t_{in_i}))$ est la matrice des fonctions de bases évaluées aux différents temps de mesure de l'individu i. Puisque les données seront balancées, $\boldsymbol{\phi_i}(t) = \boldsymbol{\phi}(t)$, pour tout $i = 1, \ldots, N$. Tout comme pour les autres méthodes qui passent par la reconstruction des données, il faudra estimer les coefficients associés des splines. Dans le cas présent, il s'agit du q-vecteur θ_μ associé avec la moyenne globale, et de la matrice Θ associée aux fonctions des composantes principales.

Les variables aléatoires Z_i, $i = 1, \ldots, N$ suivent une distribution Multinomiale avec les paramètres $(\pi_1, \ldots, \pi_G)$ de sorte que π_g est la probabilité qu'une observation se trouve dans la partition g. Puisque le modèle possède une structure bayésienne, le paramètre π_g possède une densité a priori, tout comme μ_{Z_i}, $\gamma_i^{Z_i}$ et σ^2. Les distributions a priori présentées sont pour le cas où les $Y_i(t)$ sont des données à deux dimensions. Le cas à une dimension peut être retrouvé dans le travail de Folly et Murua [**1**]. Les distributions a priori sont les suivantes:

$$\begin{cases} \mu_g & \sim & \mathcal{N}(0,\Gamma_\mu) \\ \gamma_i^g & \sim & \mathcal{N}(0,\Gamma_g) \\ \sigma^2 & \sim & \text{Gamma}(\alpha_0, \beta_0) \\ (\pi_1, \ldots, \pi_G) & \sim & \text{Dirichlet}(a_1, \ldots, a_G) \end{cases}$$

avec

$$\begin{cases} \Gamma_\mu & \sim & \text{InvWishart}(m,(m-2k-1)I_{2k}) \\ \Gamma_g & \sim & \text{InvWishart}(m,(m-2k-1)D) \\ D & \sim & \text{InvWishart}(m,(m-2k-1)I_{2k}) \end{cases}$$

On y retrouve la densité Wishart inversée, qui génère une matrice aléatoire et est souvent utilisée comme densité a priori pour la matrice de covariance des densités normales multivariées. Le travail dans [**1**] montre qu'il est préférable de prendre $(\alpha_0, \beta_0) = (3,1)$ et $a_g = 1/G$, pour $g = 1, \ldots, G$.

Pour obtenir un modèle avec une bonne structure, il reste certains détails à ajouter. La base sera choisie pour que $\boldsymbol{\phi}'\boldsymbol{\phi} = I_q$ et que $\Theta'\Theta = I_k$. Ceci permettra d'obtenir une approximation respectable de la contrainte d'orthogonalité sur les fonctions de base des

composantes principales. Ensuite, il faudra que

$$\sum_{g=1}^{G} \pi_g \mu_g = 0 \text{ et } \sum_{i=1}^{N} P_{ig} \gamma_i^g = 0, \text{ pour tout } g = 1, \ldots, G$$

pour s'assurer que $\boldsymbol{\phi}(t)\theta_\mu$ soit la moyenne globale, et que $\boldsymbol{\phi}(t)\Theta\mu_g$ soit la moyenne des courbes de la partition g.

L'estimation des paramètres se fera encore à l'aide de l'algorithme EM. Il considère la matrice des mesures observées $\mathbf{Y} = (y_1, \ldots, y_N)$ et l'ensemble des mesures non observées $\left\{\mathbf{Z}, \boldsymbol{\gamma}^{\boldsymbol{Z}}, \boldsymbol{\nu}\right\}$ tel que $\mathbf{Z} = (z_1, \ldots, z_N)$, $\boldsymbol{\gamma}^{\boldsymbol{Z}} = (\gamma_1^{z_1}, \ldots, \gamma_N^{z_N})$ et $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_N)$. L'ensemble des paramètres à estimer est $\{\boldsymbol{\mu}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda}\}$ où $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_G)$, $\boldsymbol{\Gamma} = (\Gamma_1, \ldots, \Gamma_G)$, et $\boldsymbol{\Lambda} = \{\theta_\mu, \Theta, D, \Gamma_\mu, \pi_1, \pi_2, \ldots, \pi_G, \nu_0, \sigma^2\}$. Pour utiliser l'algorithme EM, il faudrait calculer le logarithme de la vraisemblance complète et elle s'avère être très lourde en termes de calcul algébrique. Tous les détails se retrouvent dans la section 3.2.2. de [**1**]. Après la $j^{ième}$ étape de l'algorithme EM, les estimations $\boldsymbol{\mu}^{(j)}, \boldsymbol{\Gamma}^{(j)}, \boldsymbol{\Lambda}^{(j)}$ seront disponibles pour calculer la probabilité que l'objet i soit dans la partition g comme suit

$$\hat{\pi}_{ig}^{(j)} = \frac{f\left(Y_i | Z_i = g, \nu_i, \boldsymbol{\mu}^{(j)}, \boldsymbol{\Gamma}^{(j)}, \boldsymbol{\Lambda}^{(j)}\right) \pi_g^{(j)}}{\sum_{h=1}^{G} f\left(Y_i | Z_i = h, \nu_i, \boldsymbol{\mu}^{(j)}, \boldsymbol{\Gamma}^{(j)}, \boldsymbol{\Lambda}^{(j)}\right) \pi_h^{(j)}}$$

avec $f\left(Y_i | Z_i = g, \nu_i, \boldsymbol{\mu}^{(j)}, \boldsymbol{\Gamma}^{(j)}, \boldsymbol{\Lambda}^{(j)}\right)$ qui suit une densité gaussienne avec moyenne et variance

$$\boldsymbol{\phi}\theta_\mu^{(j)} + \boldsymbol{\phi}\Theta^{(j)}\mu_{z_i} \text{ et } \nu_i^{(j)}\sigma_{(j)}^2 \left(I_{2n} - \boldsymbol{\phi}\Theta^{(j)} \left(\nu_i^{(j)}\sigma_{(j)}^2 \left(\Gamma_{z_i}^{(j)}\right)^{-1} + (\Theta^{(j)})'\boldsymbol{\phi}'\boldsymbol{\phi}\Theta^{(j)}\right)^{-1} (\Theta^{(j)})'\boldsymbol{\phi}'\right).$$

Pour conclure, pour obtenir un modèle optimal les hyper-paramètres q, k et G doivent être trouvés à l'aide de différents critères de sélection. Il faudra tester plusieurs trios de paramètres (q,k,G), pour ensuite calculer les vraisemblances des modèles obtenus. Les critères BIC et AIC sont des choix naturels, cependant Murua et Folly proposent aussi l'approximation de la logarithme de vraisemblance marginale, aussi connu sous le nom de MLL de l'anglais « *Marginal log-likelihood* ». Cette quantité est obtenue en intégrant la vraisemblance estimée en conditionnant selon les paramètres du modèle. On trouvera au sujet du MLL dans [**1**].

Ceci conclut les approches par modélisation des coefficients des fonctions de base. On va maintenant discuter des modèles qui décident de s'intéresser plutôt aux scores issus de la décomposition par fonction propre.

4.3.3. funclust

Les méthodes précédentes posaient une densité sur la matrice des coefficients $\mathbf{C}$ de la combinaison linéaire $\mathbf{C}\boldsymbol{\phi}(t)$. Le nombre de calculs est alors lié à la dimension de la base q.

Les approches suivantes vont plutôt se concentrer sur la modélisation des scores obtenus grâce à l'analyse en composantes principales.

On rappelle que le but de l'ACPF est de trouver les fonctions propres qui satisfont l'équation

$$\mathcal{V}\varphi_i = \lambda_i\varphi_i, \quad \forall i \geq 1.$$

On avait expliqué dans la section 2.3.3 que résoudre cette dernière était équivalent à résoudre l'équation propre

$$N^{-1}\mathbf{W}^{1/2}\mathbf{C}'\mathbf{C}\mathbf{W}^{1/2}\mathbf{u}_j = \lambda_j\mathbf{u}_j, \text{ pour } j = 1,\ldots,N$$

où $\mathbf{u}_j = \mathbf{W}^{1/2}\mathbf{B}_j$. De ce fait, il est possible d'établir le *score* d'une composantes principales par $s_{ij} = \mathbf{c}_i\mathbf{W}\mathbf{B}_j$ avec $\mathbf{c}_i$ la $i^{ième}$ ligne de $\mathbf{C}$ et $\mathbf{W} = \int_0^T \boldsymbol{\phi}(t)\boldsymbol{\phi}(t)'dt$. Le score s_{ij} peut être vu comme étant les coordonnées de la courbe y_i selon la $j^{ième}$ composantes principales. Dans un sens plus large, $s_j = \mathbf{C}\mathbf{W}\mathbf{B}_j$.

Jacques et Preda [**14**] ont proposé ce qui serait, selon eux, le premier algorithme de partitionnement par modélisation pour les données fonctionnelles multivariées. Ils utilisent une substitution de densité en utilisant la troncature de l'expansion de Krahuenen-Loeve

$$Y^{(d)}(t) = \mu(t) + \sum_{j=1}^{d} s_j\varphi_j(t), \quad t \in [0,T]$$

où d est le nombre de composantes principales retenues pour l'analyse. On retient les d premières composantes associées aux d valeurs propres les plus élevées $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$. L'article de Delaigle et Hall [**8**] justifie cette substitution. La probabilité que la variable aléatoire Y_i soit dans une boule de rayon $\delta \geq 0$ centrée en y_i est

$$\log \mathbb{P}\left(\|Y_i - y_i\| \leq \delta\right) = \sum_{j=1}^{d} \log f_{S_j}(s_{ij}) + \xi\left(\delta, a(\delta)\right) + o(a(\delta))$$

pour des fonctions ξ et a. L'article explique que le nombre de composantes principales retenu d tend vers l'infini lorsque $\delta \to 0$, donc ajouter des composantes principales assure une convergence en probabilité. Puisque les fonctions ξ et a dépendent seulement de δ, on obtient la substitution de densité suivante :

$$f_{Y_i}^{(d)}(y_i) = \prod_{j=1}^{d} f_{S_j}(s_{ij}). \tag{4.3.5}$$

L'algorithme *funclust* utilise cette substitution pour la classification des courbes, et utilise une densité gaussienne multivariée pour $f_{S_i}(s_{ij})$.

Similairement aux méthodes précédentes, on suppose que les observations $y_1, \ldots, y_N$ sont issues de N variables aléatoires identiquement et indépendamment distribuées $Y_1, \ldots, Y_N$. Chacune de ces observations est associée à une variable latente z_i représentant le groupe de l'observation i et provenant d'une variable aléatoire Z_i. On cherche à estimer les densités des courbes d'une même partition g. Pour obtenir la substitution, une ACPF sera effectuée sur P_g. Ainsi, la densité de Y_i sachant que $z_i = g$ est

$$f^{(d_g)}_{Y_i|Z_i=g}(y_i) = \prod_{j=1}^{d_g} f_{S_j|Z_i=g}\left(s_{ijg}\right),$$

où d_g est le nombre de composantes principales retenues pour le groupe g et s_{ijg} est le scores de la $j^{ième}$ composante pour y_i au sein du groupe g. On supposera que la densité $f_{S_j|Z_i=g}$ est une gaussienne multivariée de taille d_g de moyenne 0 avec la matrice de covariance Σ_g. Cette matrice est diagonale, avec les valeurs propres de l'ACPF $(\lambda^g_1, \ldots, \lambda^g_{d_g})$ correspondant aux variances des composantes principales.

En reprenant la notation $\pi_g = \mathbb{P}(Z_i = g)$ et en définissant $d = (d_1, \ldots, d_G)$, la densité non conditionnelle de chaque Y_i est

$$f^{(d)}_{Y_i}(y_i) = \sum_{g=1}^{G} \prod_{j=1}^{d_g} \pi_g f_{S_j|Z_i=g}\left(s_{ijg}\right).$$

En passant par les scores des courbes, le but est de prendre en compte la dépendance entre ceux-ci. Le calcul des scores utilise l'ensemble des données et analyse leur variabilité. Nécessairement, le score s_{ij} sera affecté par la composition de l'échantillon obtenu.

Le modèle *funclust* établie, il faut à présent estimer l'ensemble des paramètres $\Pi = \{\pi_1, \ldots, \pi_G, \Sigma_1, \ldots, \Sigma_G\}$. Cependant, il faut aussi estimer les scores des composantes principales et les variables latentes $z_1, \ldots, z_N$. De plus, il faut déterminer les dimensions $d_1, \ldots, d_G$ des scores retenus pour chaque partition, mais ils seront obtenus empiriquement avec le test de Catell.

L'algorithme EM sera utilisé pour *funclust*, avec des étapes supplémentaires entre le calcul de l'espérance conditionnelle (l'étape E) et celle de la maximisation (l'étape M), qui permettront de mettre à jour les scores et de déterminer la taille des dimensions $d_1, \ldots, d_G$. Après avoir initié les paramètres Π à une valeur initiale, l'algorithme répète les quatre étapes suivantes jusqu'à convergence.

Étape E : Avec les vraies valeurs $z_1, \ldots, z_N$ inconnues, on commence par calculer l'espérance conditionnelle du logarithme de la vraisemblance pseudo-complète :

$$Q\left[\Pi|\Pi^{(j)}\right] = \mathbb{E}\left[\ell^{(q)}\left(\Pi|\mathbf{Y}\right)\right] = \sum_{i=1}^{N}\sum_{g=1}^{G} t_{ig}\left(\log \pi_g + \sum_{j=1}^{d_g} \log f_{S_j|Z_i=g}(s_{ijg})\right).$$

La variable t_{ig} est la probabilité que la courbe y_i appartienne au groupe g calculé avec le score observé s_{ijg}

$$t_{ig}^{(j)} \approx \frac{\pi_g \prod_{j=1}^{d_g} f_{S_j|Z_i=g}\left(s_{ijg}\right)}{\sum_{l=1}^{G} \pi_l \prod_{j=1}^{d_l} f_{S_j|Z_i=l}\left(s_{ijl}\right)}.$$

L'approximation vient du fait que les densités sont obtenues à l'aide d'une substitution.

Mise à jour des scores : Avec les probabilités t_{ig}, on est en mesure de mettre les scores des composantes à jour. Les probabilités permettront de donner un poids à chaque courbe y_i dans la construction de chaque score. Donc pour un groupe g, la matrice des coefficients $\mathbf{C}$ est remplacé par $\mathbf{C}_g = (I_N - \mathbb{1}_N(t_{1g}^{(j)}, \ldots, t_{Ng}^{(j)}))\mathbf{C}$ ce qui permet de calculer $s_{ijg} = \mathbf{c}_{ig}\mathbf{W}\mathbf{B}_j$ où $\mathbf{c}_{ig}$ est la $i^{ième}$ ligne de $\mathbf{C}_g$. De plus, on obtient une mise à jour de la variance des valeurs propres et par conséquent de celle de $\hat{\Sigma}_g^{(j)}$

Estimation des dimensions d_g : Selon Jacques et Preda [**14**], le problème des dimensions d_g est «un problème ouvert». L'algorithme *funclust* utilise le test d'éboulis de Cattell [**5**]. De plus, les dimensions initiales sont toutes 1 et elles peuvent seulement augmenter entre chaque itération de l'algorithme.

Étape M : Finalement, on maximise la quantité $Q\left[\Pi|\Pi^{(j)}\right]$, ce qui amène les estimations suivantes :

$$\hat{\pi}^{(j+1)} = \frac{1}{N}\sum_{i=1}^{N} t_{ig}^{(j)}$$

Enfin, l'hyper-paramètre de l'algorithme *funclust* est le nombre de groupes G. Naturellement, le meilleur modèle retenu sera celui avec les critères AIC et BIC optimaux. Le grand problème de cette approche est la formulation de t_{ig}, l'estimation de la probabilité qu'une courbe i soit dans la partition g. Cela nécessite le calcul du score s_{ijg}, qui dépend de de la dernière estimation t_{ig} lors de l'algorithme EM. Il n'est pas possible de prédire la probabilité qu'une nouvelle courbe y^* appartienne à une partition g, puisqu'on ne pourra pas calculer son score dans l'espace de la partition g selon l'approche donnée par *funclust*. Il n'y aura donc aucune prédiction faite à l'aide de ce modèle.

On a maintenant un algorithme de base qui passe par la modélisation des scores des courbes. Celui-ci retient les scores les plus importants. Or, le prochain modèle *funHDDC* propose un plus grand contrôle sur la densité des scores.

4.3.4. funHDDC

La prochaine méthode a été conçue par Jacques, Bouveyron et al. [**3**]. Comme le modèle *funclust*, elle suppose que les scores issus de l'ACP fonctionnelle multivariée sont des variables aléatoires. Cependant, elle ajoute une paramétrisation supplémentaire sur la matrice de covariance et sur la dimension de la densité.

Ce modèle suit les mêmes bases que les autres modèles. Le but est de séparer les observations $y_1, \ldots, y_N$ en G partitions. Soit la variable aléatoire latente Z_{ig} tel que $Z_{ig} = 1$ si y_i appartient à la partition P_g et 0 sinon. Posons n_g le nombre de courbes dans la partition g. Avec ce qui a été établi dans la section précédente, les scores seront désignés par $s_i = (s_{i1}, \ldots, s_{iq})'$ où q est le nombre de fonctions dans $\boldsymbol{\phi}(t) = (\phi_1(t), \ldots, \phi_q(t))'$.

Ensuite, les courbes d'une partition P_g auront leurs propres fonctions de base $\psi_1^g(t), \ldots, \psi_q^g(t)$ telles que

$$\psi_j^g(t) = \sum_{l=1}^{q} h_{gjl}\phi_l(t).$$

Ceci implique que les poids des fonctions de base pour chaque partition seront distincts.

La matrice H_g, qui contient les coefficients h_{gjl} associés aux fonctions de base, est de taille $q \times q$ et orthogonale. Les fonctions $\phi_1, \ldots, \phi_q$ sont, dans notre cas, obtenues à l'aide de B-splines. Pour des raisons qui vont devenir apparentes sous peu, H_g est séparée en deux parties tel que $H_g = [U_g, V_g]$. U_g et V_g sont de taille $q \times d_g$ et $q \times (q - d_g)$ respectivement, et puisque H_g est orthogonale, $U_g'U_g = I_{d_g}$, $V_g'V_g = I_{q-d_g}$ et $U_g'V_g = 0$.

Maintenant, la méthode *funHDDC* va modéliser les scores $s_1, \ldots, s_N$ au lieu des observations. Elle suppose que les scores des données d'une partition P_g sont issus des variables aléatoires $S_1^g, \ldots, S_{n_g}^G$ qui sont identiquement et indépendamment distribuées selon une distribution gaussienne multivariée, comme suit :

$$S_i^g \sim \mathcal{N}(\mu_g, \Sigma_g)$$

avec $\mu_g \in \mathbb{R}^q$ et Σ_g la matrice de covariance d'ordre q. Le cœur du modèle sera dans la modélisation de Σ_g. Le but est d'accorder une plus grande importance aux d_g premières composantes et de traiter les $q - d_g$ comme étant des composantes de bruit. Plus précisément, avec le d_k-vecteur $(a_{g1}, \ldots, a_{gd_g})$ et le $q - d_k$-vecteur $(b_g, \ldots, b_g)$, la matrice de covariance a

la forme suivante :

$$\Sigma_g = \left(\begin{array}{ccc|cc} a_{g1} & & 0 & & \\ & \ddots & & & \mathbf{0} \\ 0 & & a_{gd_g} & & \\ \hline & & & b_g & 0 \\ & \mathbf{0} & & & \ddots \\ & & & 0 & b_g \end{array}\right)$$

Comme mentionné dans l'article [**3**], cette paramétrisation prend le nom de $[a_{gj}b_gH_gd_g]$. On retrouve le modèle *funclust* lorsque $b_1 = \ldots = b_G = 0$ et qu'on centre la densité.

De là, cinq sous modèles peut être définis par différentes contraintes appliquées sur les a_{jg} et les b_g.

- $[a_{gj}b_gH_gd_g]$: Aucune contrainte imposée sur les paramètres.
- $[a_{gj}bH_gd_g]$: Les composantes de bruit sont les mêmes entre chaque partition.
- $[a_gb_gH_gd_g]$: Les premières composantes d'une même partition sont fixes, celles de bruit varient.
- $[a_gbH_gd_g]$: Les premières composantes d'une même partition et celles de bruit sont fixes.
- $[ab_gH_gd_g]$: Les premières composantes sont fixes entre les partitions, celles de bruit varient.
- $[abH_gd_g]$: Toutes les composantes sont fixes entre les partitions.

L'algorithme *funmbclust* détermine le meilleur modèle à l'aide de critères de sélection. Pour ce qui est de la taille de dimension d_g de chaque groupe, un test d'éboulis (*scree-test*) de Cattell est utilisé [**5**]. La taille de la dimension peut varier durant l'algorithme EM, et sera déterminée durant ce processus.

Puisque ce partitionnement est fait par un modèle statistique, l'estimation des paramètres est faite en maximisant la vraisemblance avec l'aide d'un algorithme EM. Le modèle $[a_{gj}b_gH_gd_g]$ sera considéré pour la présentation de la procédure étant donné qu'il est la forme la plus générale. Les scores de l'ACPF multivariée sont des variables aléatoires, mais leurs densités dépendent du groupe. Alors avec S la variable aléatoire d'un score quelconque,

$$f_S(s) = \sum_{g=1}^{G} \pi_g (2\pi)^{-\frac{q}{2}} |\Sigma_g|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(s-\mu_g)' \Sigma_g^{-1}(s-\mu_g)\right)$$

où π_g est la probabilité qu'une courbe soit dans la partition P_g.

En utilisant l'algortihme Em ici, le but est d'obtenir le vecteur de paramètre $\mathbf{\Pi} = \{\pi_1, \ldots, \pi_G, \mu_1, \ldots, \mu_G, \mathbf{a}_1, \ldots, \mathbf{a}_G, b_1, \ldots, b_G, H_1, \ldots, H_G\}$ où $\mathbf{a}_g = (a_{g1}, a_{g2}, \ldots, a_{gd_g})$. Les

variables latentes dans ce modèle sont les z_{ig}, qui déterminent si l'individu i appartient au groupe g. De plus, on définit la matrice $\mathbf{C}_g = \dfrac{1}{n_g}\sum_{i=1}^{N} z_{ig}\left(\mathbf{c}_i - \mu_g\right)'\left(\mathbf{c}_i - \mu_g\right)$.

Étape E: Calculer la probabilité a posteriori de faire partie du groupe g pour chaque courbe:

$$t_{ig}^{(j)} = \mathbb{E}\left[z_{ig}|\mathbf{c}_i, \mathbf{\Pi}^{(j-1)}\right] = \left[\sum_{l=1}^{G}\exp\left(\frac{1}{2}\left(L_g^{(j-1)}(\mathbf{c}_i) - L_l^{(j-1)}(\mathbf{c}_i)\right)\right)\right]^{-1} \tag{4.3.6}$$

tel que $L_g^{(j-1)}(x)$ est une fonction définie dans $\mathbb{R}^q$ comme suit :

$$\begin{aligned} L_g^{(j-1)}(x) = {} & \|\mu_g^{(j-1)} - \mathbf{P}(x)\|_{D_g}^2 + \frac{1}{b_g^{(j-1)}}\|c - \mathbf{P}_g(x)\|^2 \\ & + \sum_{k=1}^{d_g}\log\left(a_{gk}^{(j-1)}\right) + (q-d_g)\log(b_g^{(j-1)}) - 2\log\left(\pi_g^{(j-1)}\right) \end{aligned} \tag{4.3.7}$$

La fonction $L_g^{(j-1)}(x)$ utilise plusieurs notations et concepts propres au problème. D'abord, la norme $\|\cdot\|_{D_g}^2$ sur l'espace latent $\mathbf{E}_g$ est définie par $\|z\|_{D_g}^2 = z'D_g z$, $D_g = \tilde{H}_g\Sigma_g$ avec $\tilde{H}_g = \left[U_g, 0_{q-d_g}\right]$. De plus, $\mathbf{P}_g$ est l'opérateur de projection sur l'espace fonctionnel $\mathbf{E}_g$ défini par $\mathbf{P}_g(\theta) = \mathbf{W}U_g'U_g\mathbf{W}'(c-\mu_g) + \mu_g$. Les preuves de tous ces résultats peuvent être trouvées dans l'article [**3**].

M step: Les paramètres du modèle sont estimés en maximisant le logarithme de la vraisemblance complète, sachant les probabilités a posteriori $t_{ig}^{(j)}$. Les scores moyens de chaque groupe et les proportions mixtes sont mis à jour de la manière suivante :

$$\hat{\pi}_k^{(j)} = \frac{\eta_g^{(j)}}{N},\ \mu_g^{(j)} = \frac{1}{\eta_g^{(j)}}\sum_{i=1}^{N} t_{ig}^{(j)}\mathbf{c}_i,$$

où $\eta_g^{(j)} = \sum_{i=1}^{N} t_{ig}^{(j)}$. Les $t_{ig}^{(j)}$ finaux détermineront la classification des données. Il sera possible de réutiliser le meilleur modèle obtenu pour classifier un nouvel ensemble de données à l'aide de l'approche bayésienne. L'estimation des autres paramètres se retrouve dans [**3**]. Malgré que le calcul des $t_{ig}^{(j)}$ soit plus complexe que celui proposé par *funclust*, tous les paramètres ne dépendent pas des observations. Ceci implique qu'il sera possible de prendre les estimations $\hat{\mu}_g, \hat{a}_{gk}, \hat{b}_g, \hat{H}_g$ et $\hat{\pi}_g$ pour calculer $\mathbb{P}(y^* \in g)$ pour une nouvelle obsevation y^*.

Pour un nombre de fonctions de base q donné, l'algorithme va déterminer le meilleur modèle $[a_{gj}b_gH_gd_g]$ et le nombre de partitions g avec le BIC et l'AIC.

Plutôt que de modéliser les courbes $y_1, \ldots, y_N$ directement, l'approche *funHDDC* s'intéresse plutôt aux scores de celles-ci. Il va même déterminer quelle partie de ces scores a

une importance, et traiter l'autre part comme étant du bruit.

Chapitre 5

Simulation : résultats du partitionnement

Il est maintenant temps d'appliquer les méthodes présentées aux données de trajectoires de voitures. On rappelle que notre but est d'évaluer la capacité de ces algorithmes à faire de la classification dans une première phase d'entraînement. Les résultats seront utilisés pour faire de la prédiction sur un nouvel ensemble de données.

Ce chapitre portera sur les résultats de la classification obtenue par les six méthodes. Or, les données doivent être pré-traitées pour l'analyse. On discutera de l'interpolation LOESS [**7**] pour obtenir des données sur la même échelle de temps, et de l'échantillonnage pour la construction des ensembles d'entraînement. On présentera aussi l'indice ajusté de Rand [**25**], qui permettra d'évaluer la qualité de la classification. Il y aura également une courte analyse en composantes principales fonctionnelles. Enfin, les résultats du partitionnement et des prédictions seront présentés avec des explications des impacts de chacune des méthodes.

5.1. Préparation des données

Comme mentionné dans l'introduction, les données ont été placées dans un même intervalle de temps [0,1]. Le temps 0 représente l'entrée de la voiture dans le capteur vidéo et 1 représente sa sortie. Un premier problème est que les librairies *distclust*, *mclust* et *fitfclust* ne sont pas compatibles avec le cas multivarié. On a donc traité les données comme étant univariées, en mettant les abscisses et ordonnées en un même vecteur. En d'autres mots, la trajectoire

$$y_i = \left(y_i^1(t), y_i^2(t)\right) = \begin{pmatrix} y_i^1(t_1) & y_i^2(t_2) \\ \vdots & \vdots \\ y_i^1(t_n) & y_i^2(t_n) \end{pmatrix} v,$$

est transformée dans la courbe univariée

$$\tilde{y}_i(t) = \left(y_i^1(t_1), \ldots, y_i^1(t_n), y_i^2(t_1), \ldots, y_i^2(t_n)\right).$$

Il est clair que la rupture entre les points $y_i^1(t_n)$ et $y_i^2(t_1)$ crée un biais dans la modélisation. Cependant, le but premier est le partitionnement des données et cette rupture ne cause aucun problème à cet égard.

Pour réaliser cette étude, il a fallu attribuer manuellement une classe à 979 trajectoires prises des vidéos, puisqu'on veut évaluer la qualité de classification et de prédiction des six méthodes. L'information disponible était les coordonnées des voitures à chaque fréquence de temps. On rappelle que les mesures de temps données sont selon le temps dans la vidéo. Alors la première mesure de coordonnées pour une voiture correspond alors au temps de son apparition dans la vidéo, et la dernière mesure son temps de sortie. Ainsi, on a pu identifier visuellement chacune des voitures directement avec les vidéos. On a confirmé les points d'entrée et de sortie de chaque trajectoire, et on les a classifiés par conséquent.

5.1.1. Interpolation LOESS

Les données brutes ne sont pas régulières, c'est-à-dire l'information sur la position de chacune des voitures n'est pas fixée sur la même grille de temps entre chacune d'entre elles. En d'autres mots, soit n_i le nombre de points de position que la caméra a capté pour la voiture i et n_j celle de la voiture j. Alors n_i n'a pas nécessairement la même valeur que n_j, $i \neq j$. Il est quand même possible de travailler avec les données en tant que telles, mais la charge numérique va grandement s'alourdir. C'est pour cela qu'elles seront réajustées afin que les positions des voitures soient toutes sur la même grille, c'est-à-dire que $n_i = n$, pour toute les observations $i = 1, \ldots, N$, où N est le nombre total d'observations.

Chaque trajectoire a été mise sur la même grille de temps à partir d'un intervalle de temps commun $[0,T]$. Ceci a été fait au moyen d'une régression polynomiale locale, mieux connue sous le nom de *LOESS* (de l'anglais *LOcally Estimated Scatterplot Smoothing*), et élaborée par Cleveland [**7**]. Cette méthode non paramétrique permet de passer une courbe par un ensemble de points et a la particularité d'être robuste aux points aberrants. De plus, la régression se fait localement : elle s'exécute dans un voisinage pour chaque mesure de temps. On suppose que chaque courbe peut s'écrire sous la forme $y_i(t) = \beta(t) + \epsilon_i$, où ϵ_i est le terme d'erreur d'espérance nulle avec une variance constante σ^2. La méthode trouve un estimateur des moindres carrées pondérées $\hat{\beta}$ tel que $\hat{y}_i(t) = \hat{\beta}(t)$ à un point $t \in [0,T]$. L'estimateur utilise des poids $w_1, \ldots, w_{n_i}$ qui sont choisis en fonction du voisinage de chaque mesure de temps t_l. Plusieurs variables doivent être définies pour établir cette régression.

On parle d'une régression locale, puisqu'on a un voisinage de points pour chaque mesure de temps t_l. D'abord, $\alpha \in]0,1]$ est la proportion de mesure parmi toutes les mesures

possibles qui formeront le voisinage $\mathbf{V}(t_l)$. Ce paramètre est connu sous le nom de *paramètre de lissage*, puisqu'il contrôle la qualité de la régression. Ce sous-ensemble aura $r = \lfloor \alpha n_i \rfloor$ observations pour la régression locale. On en déduit que la population totale forme le voisinage $\mathbf{V}(t_l)$ lorsque $\alpha = 1$.

Maintenant, pour chaque $t_l \in \{t_1, \ldots, t_n\}$, soit $\delta_l(t) = |t - t_l|$ une mesure de distance entre le temps de mesure t_l et les autres mesures. Ensuite, soit $\delta_l^{(k)}(t)$ la $k^{ième}$ plus petite distance par rapport à t_l. De plus, on prend W la fonction de poids tricubique définie par

$$W(u,v) = \begin{cases} (1 - (u/v)^3)^3 & \text{si } 0 \leq u \leq v \\ 0 & \text{si } u > v \end{cases}$$

où $u,v \geq 0$. Pour un r fixé, les poids pour chaque paire $(y_i(t_l), t_l)$ sont donnés par $w_l(t) = W(\delta_l(t), \delta_l^{(r)}(t))$. Ces poids décroissent ou restent les mêmes lorsque la distance entre t_l et t augmente. Une fois les w_l calculés, il ne reste plus qu'à obtenir l'estimateur des moindres carrées pondérées polynomial $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_d)$ qui minimise la quantité

$$\sum_{l=1}^{n} w_l(t) \left(y_i(t) - \beta_0 - \beta_1 t - \ldots - \beta_d t^d \right)^2.$$

Il faut aussi faire le choix du degré polynomial d. Puisque $\hat{\beta}$ est l'estimateur des moindres carrées, une estimation quadratique est obtenue lorsque $d = 2$. Il n'est pas nécessaire de prendre $d > 2$, puisqu'on ne fait qu'augmenter la charge de calcul et que l'on a déjà les deux premières dérivées continues lorsque $d = 2$.

Ainsi, on a une estimation $\hat{y}_i(t)$ pour n'importe quel point $t \in [0,T]$ à partir de ceux observés. Cela permet d'établir une grille de temps $(t_1,t_2, \ldots, t_n)$, et d'interpoler chaque courbe $y_1,y_2, \ldots, y_N$ selon celle-ci, et ainsi équilibrer les données. Pour notre cas, on prend un paramètre de lissage $\alpha = 3/4$ et on applique la régression LOESS sur chacune des deux dimensions des coordonnées des trajectoires. Originalement, le nombre de mesures de temps captées par la caméra varie entre 100 et 1500 par trajectoire. Pour ne pas alourdir le calcul, on a donc fixé le nombre de mesures de temps à $n = 75$.

5.1.2. Construction des échantillons

Les six méthodes présentées dans le chapitre 4 sont entraînées en partitionnant des jeux de données issues de la grande totalité. Il faut que ces jeux de données représentent la population totale et qu'ils possèdent des courbes de chaque partition connue, sinon on obtiendra des partitionnements fautifs. On rappelle que les courbes représentent des trajectoires de véhicules dans une intersection à trois branches. Une classification suggérée est la direction

de ces véhicules, c'est-à-dire leurs entrées et sorties lorsqu'ils utilisent l'intersection, ce qui donne six partitions. Alors l'ensemble d'entraînement est alors construit en prenant n courbes de chaque partition, à l'aide d'un échantillonnage aléatoire simple sans remise [**21**]. Dans une même partition, chaque courbe possède la même probabilité de sélection. Notre échantillon a donc 20 trajectoires issues de chaque partition qui sont prédéterminées, ce qui donne un échantillon de taille $N = 120$. On a évité d'avoir un N plus grand, afin d'éviter une surcharge lors des calculs numériques; certains algorithmes, comme *funmbclust*, ont de la difficulté à converger lorsque N est grand. En revanche, les modèles ellipsoïdaux ne pourront pas être testés puisque la reconstruction de données en forme univariée fait en sorte que le nombre d'observations $N = 120$ sont inférieur au nombre de composantes $2n = 150$.

L'échantillon de 120 trajectoires formera l'ensemble d'entraînement. Il servira pour le partitionnement et l'entraînement des six méthodes. L'ensemble utilisé pour évaluer la qualité des prédictions des méthodes est les 859 trajectoires qui ne font pas parties de l'ensemble d'entraînement. Les deux ensembles n'ont aucune trajectoire commune.

5.1.3. Indice ajusté de Rand

Une mesure de qualité est utilisée pour évaluer le partitionnement d'un algorithme. Il s'agit de l'*indice de Rand*, qui mesure l'accord entre deux partitionnements d'un ensemble de données. Pour définir l'indice, soit la paire d'observations (y_i, y_j) provenant d'un ensemble de données et on suppose qu'il existe deux partitions $P_1, \ldots, P_G$, $G \in \mathbb{N}$, et $\hat{P}_1, \ldots, \hat{P}_{G'}$, $G' \in \mathbb{N}$. Dans notre cas, la première sera prédéterminée lors de la préparation des données et la deuxième sera obtenue à l'aide des méthodes de partitionnement fonctionnel. L'indice définit les variables indicatrices suivantes :

- $\Delta(i,j) = 1$ s'il existe des valeurs $g \in \{1, \ldots, G\}$ et $g' \in \{1, \ldots, G'\}$ telles que la paire $(y_i, y_j) \in P_g$ et que $(y_i, y_j) \in \hat{P}_{g'}$. On appellera ce cas des *paires en accord*.
- $\Delta(i,j) = 1$ si pour tous les $g \in \{1, \ldots, G\}$ et les $g' \in \{1, \ldots, G'\}$, on a $(y_i, y_j) \notin P_g$ et $(y_i, y_j) \notin \hat{P}_{g'}$. On appellera ce cas des *paires en désaccord*.
- $\Delta(i,j) = 0$ dans tout autre cas.

Puisqu'il existe $\binom{N}{2}$ paires d'observations possibles, l'indice de Rand est

$$R = \frac{1}{\binom{N}{2}} \sum_{i<j}^{N} \Delta(i,j).$$

Alors, R comptabilise toutes les paires qui n'ont pas changé dans leur relation de partition. Il prend les valeurs dans l'intervalle [0,1], 0 étant aucun accord entre les partitions et 1 étant un accord parfait.

Cependant, cet indice ne considère pas la distribution nulle de la partition obtenue. En d'autres mots, il n'y a aucune correction pour le hasard. Hubert et Arabie [**12**] proposent un indice de Rand ajusté qui normalise R. On construit le tableau de contingence suivant :

	$\hat{P}_1$	$\hat{P}_2$	$\cdots$	$\hat{P}_{G'}$	$Total$
P_1	N_{11}	N_{12}	$\cdots$	$N_{1G'}$	$N_{1.}$
P_2	N_{21}	N_{22}	$\cdots$	$N_{2G'}$	$N_{2.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
P_G	N_{G1}	N_{G2}	$\cdots$	$N_{GG'}$	$N_{G.}$
$Total$	$N_{.1}$	$N_{.2}$	$\cdots$	$N_{.G'}$	N

Ici, les $N_{ij} = |P_i \cap \hat{P}_j|$, $N_{.j} = \sum_{i=1}^{G} N_{ij}$ et $N_{i.} = \sum_{j=1}^{G'} N_{ij}$. Si les valeurs du tableau étaient générées complètement aléatoirement, les valeurs N_{ij} suivraient une distribution hypergéométrique généralisée. Alors l'espérance du nombre de paires d'objets en accord serait

$$\mathbb{E}\left[\sum_{i,j}\binom{N_{ij}}{2}\right] = \left[\sum_{i=1}^{G}\binom{N_{i.}}{2}\sum_{j=1}^{G'}\binom{N_{.j}}{2}\right] / \binom{N}{2}$$

De plus, le nombre total de paires en accord est $\sum_{i,j}\binom{N_{ij}}{2}$. Si on a un partitionnement parfait, $N_{ij} = N_{.j} = N_{i.}$ lorsque $i = j$ et $N_{ij} = 0$ si $i \neq j$. Le nombre total de paires en accord devient $\frac{1}{2}\left[\sum_{i=1}^{G}\binom{N_{i.}}{2} + \sum_{j=1}^{G'}\binom{N_{.j}}{2}\right]$. Ainsi, l'indice de Rand ajusté, noté «ARI» (de l'anglais *Ajusted Rand Index*), est

$$ARI = \frac{\sum_{i,j}\binom{N_{ij}}{2} - \left[\sum_{i=1}^{G}\binom{N_{i.}}{2}\sum_{j=1}^{G'}\binom{N_{.j}}{2}\right] / \binom{N}{2}}{\frac{1}{2}\left[\sum_{i=1}^{G}\binom{N_{i.}}{2} + \sum_{j=1}^{G'}\binom{N_{.j}}{2}\right] - \left[\sum_{i=1}^{G}\binom{N_{i.}}{2}\sum_{j=1}^{G'}\binom{N_{.j}}{2}\right] / \binom{N}{2}}.$$

On compare la répartition observée des paires d'objets en accord au cas d'une partition parfaite, tout en corrigeant chaque partie par l'espérance de la distribution nulle du tableau de contingence. On déduit un accord très satisfaisant entre deux partitions lorsque le ARI est proche de 1.

Le ARI sera utilisé pour évaluer la qualité du partitionnent obtenue par les différents algorithmes.

5.2. Choix de l'échantillon et des paramètres

C'est grâce au progiciels **R** et **Java** que les résultats sont obtenus par la suite et les librairies sont disponibles au grand public.

Pour chaque méthode, il faudra contrôler les hyper-paramètres, s'il y en a. Un hyper-paramètre commun à tous est le nombre de partitions G. Les algorithmes testeront les valeurs 3 à 12. La borne inférieure est choisie pour couvrir le cas où un algorithme ne fait aucune différence entre le sens des trajectoires. En d'autres mots, une voiture qui rentre dans l'intersection par l'entrée de gauche et sort par la droite pourrait être vue de la même manière qu'une voiture qui fait le chemin inverse. Toujours de par la nature des données, tester les algorithmes pour un partitionnement à 2 groupes serait inutile. La limite supérieure 12 est choisie pour représenter l'hypothèse que les conducteurs font soit des virages serrés, soit des virages plus amples à l'intersection. Autrement dit, certains véhicules passeront plus près du séparateur routier que d'autres, et il se peut qu'un algorithme détecte cette distinction.

Les autres hyper-paramètres dépendent des méthodes. Le nombre de fonctions de base q est un hyper-paramètre commun à *fitfclust*, *fumbclust*, *funclust*, et *funHDDC*. Il sera préférable de fixer ce paramètre pour toutes les méthodes. Un q élevé cause une surestimation et peut créer des BIC et AIC plus élevés, tout en augmentant la charge des calculs numériques. De plus, un q fixé permet une meilleure comparaison entre les résultats. Pour déterminer le meilleur q, on a procédé à une régression par splines bivariée sur l'échantillon $N = 120$ avec $q = 5, \ldots, 50$. Après, on a comparé les moyennes des sommes des erreurs aux carrés données par $N^{-1} \sum_{i=1}^{N} \|\hat{y}_i - y_i\|^2$. Cette comparaison se retrouve dans la figure 5.1. Il faut faire un choix avec parcimonie entre une moyenne des erreurs faible et un q assez bas. On a décidé de prendre $q = 25$, puisque la moyenne des erreurs se stabilise autour de ce point.

Finalement, le nombre de composantes principales fonctionnelles retenues est un hyper-paramètre pour *funmbclust*. Les valeurs entières testées vont de 1 à 8. Le choix sera expliqué dans la prochaine section lors de l'analyse en composantes principales fonctionnelles.

5.3. Analyse des résultats

5.3.1. Résultats de l'analyse en composantes principales fonctionnelles

Avant de parler des résultats principaux, une analyse en composantes principales fonctionnelles (ACPF) a été faite sur les coordonnées des abscisses et des ordonnées.

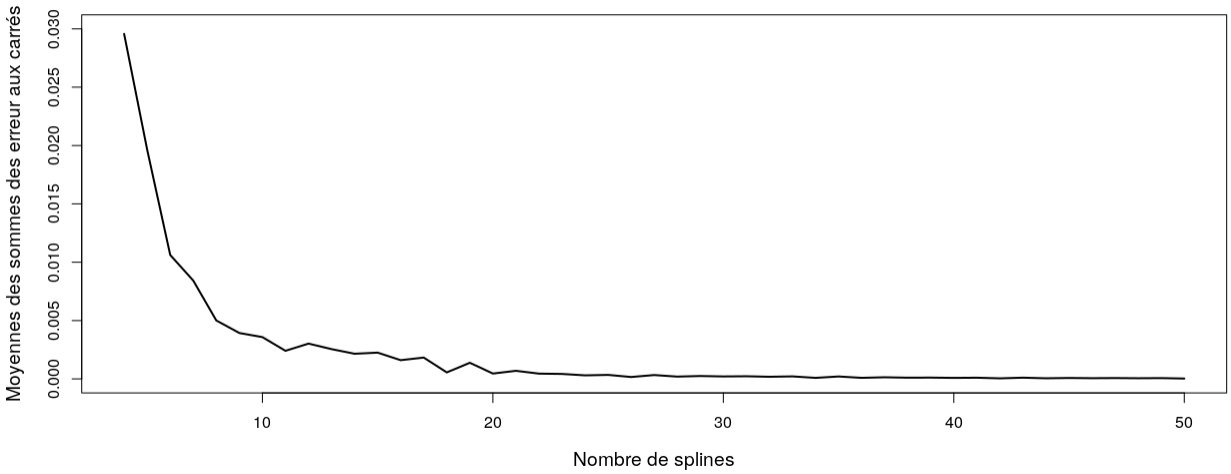

Fig. 5.1. Les moyennes des sommes des erreurs aux carrés selon le nombre de splines utilisé suite à la régression bivariée sur l'ensemble d'entrainement.

Cela pour donner une idée du comportement des données, de comment les algorithmes comme *funHDDC* ont formé leurs dimensions et du nombre de composantes à tester pour l'algorithme *funmbclust*.

On a d'abord reconstruit les données à l'aide de 25 B-splines définies sur l'intervalle de temps [0,1]. On procède avec l'ACPF sur chacune des dimensions. L'analyse des 4 premières composantes principales peut être observée dans la figure 5.2. Lorsqu'on compare les courbes associées aux deux dimensions, on remarque des symétries réflectives presque parfaites des courbes des composantes principales selon un axe horizontal au temps médian $t = 0{,}5$, sauf exception pour la quatrième composante. La proportion de variance expliquée est similaire pour les deux, avec les deux premières composantes principales expliquant 96,6% et 97,5% de la variance des coordonnées x et y respectivement. Ces deux composantes attribuent un poids important aux coordonnées à l'entrée et la sortie des trajectoires dans le capteur caméra. Ce n'est pas surprenant, car c'est la branche d'arrivée et de départ de la voiture à l'intersection qui définit son parcours. Les troisième et quatrième composantes attribuent une importance autour du centre de l'intervalle de temps [0,1], qui est sensiblement le moment où les voitures effectuent leur virage.

Pour l'algorithme *funmbclust*, la dimensions de l'ACPF est un hyper-paramètre. Les résultats de Folly et Murua [**1**] suggèrent que *funmbclust* performe bien lorsque $k = 6$, on a donc choisi de tester les valeurs 1 à 8.

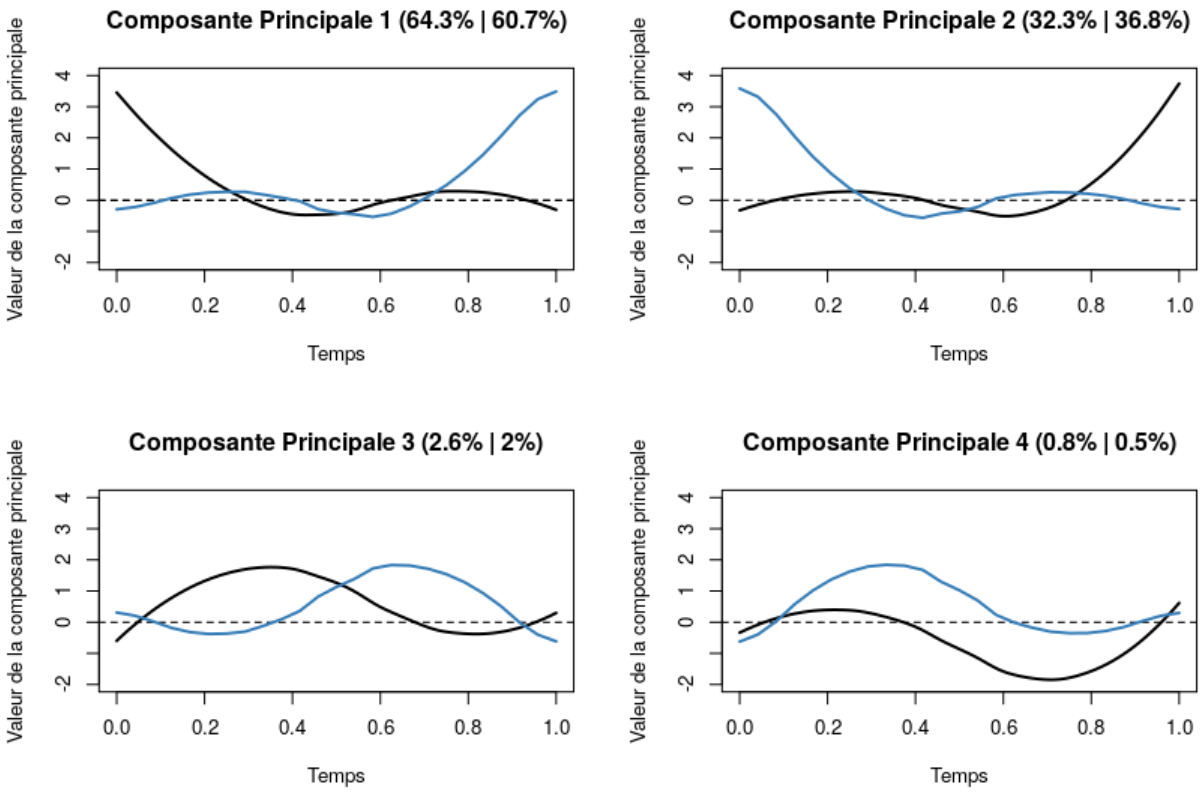

Fig. 5.2. Analyse en composantes principales sur les coordonnées. Les composantes principales de l'analyse faite sur les abscisses sont représentées par les courbes *noires*. Les composantes principales de l'analyse faite sur les ordonnées sont représentées par les courbes *bleu*.

5.3.2. Résultats des partitionnements

On passe maintenant au cœur des résultats. Les ARI, BIC et AIC seront analysées pour chacune des méthodes, à l'exception de la méthode *distclust* : puisqu'elle est basée sur la notion de distance, il ne sera pas possible d'obtenir un BIC et un AIC. On préserve les algorithmes entraînés, associés à chacun des critères optimaux. Dans les rares cas où plus d'une combinaison de paramètres résultent en un même ARI, on analysera les BIC et AIC ensuite. Pour chaque modèle et critère de sélection, des représentations graphiques des trajectoires moyennes de chaque partition seront présentées comme guide de référence.

Les résultats avec la méthode par distance *distclust* présentent des ARI forts. En effet, le tableau 5.1 montre des ARI au-dessus de 0,8 pour toutes les partitions allant de 6 à 11. On atteint même un ARI de 0,922 lorsque $G = 6$, ce qui est très conclusif. Cependant, l'ARI

est subjectif à notre choix prédéterminé du partitionnement de l'échantillon. Pour le positionnement multidimensionnel, les distances ont été projetées dans un espace à dimension 4.

G	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
ARI	0,316	0,631	0,773	**0,922**	0,875	0,844	0,866	0,808	0,806	0,762

Tableau 5.1. Résultats du partitionnement avec l'algorithme *distclust*.

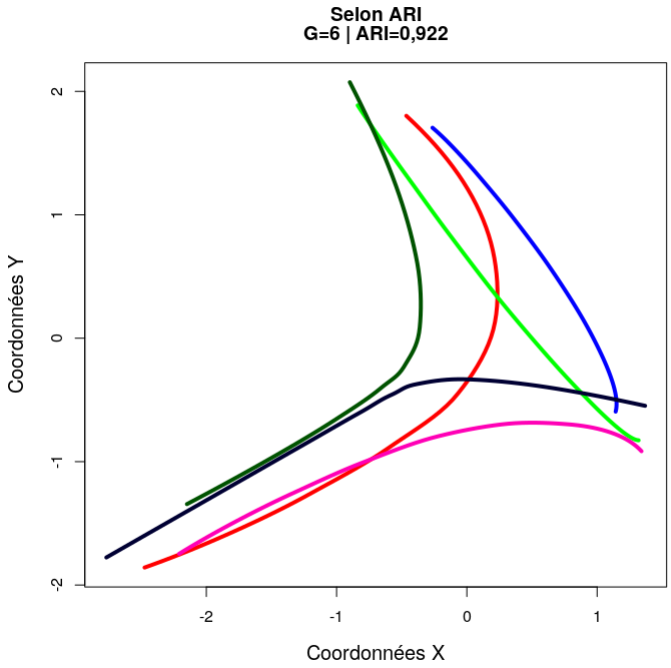

Fig. 5.3. Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *distclust*.

L'algorithme de mélange de densité gaussienne *mclust* présente aussi des ARI satisfaisants, allant même jusqu'à 0,98. Les modèles ellipsoïdaux *EEE, EEV, VEV et VVV* ne pouvaient pas être testés, dû à la structure des données. Une chose qui inquiète lors de l'analyse du tableau 5.2 est que les BIC et AIC croissent lorsque G augmente. Il est possible que trop de G aient été testés. Cependant, le G optimal selon ces critères est de 10 pour des modèles différents. Avec le BIC, c'est le modèle *VII* qui correspond à des matrices de covariance distincte entre les groupes de la forme $\Sigma_g = \lambda_g I$. Puisqu'elle est diagonale, la covariance entre les temps de mesure des courbes est nulle, ce qui va à l'encontre de la définition des données longitudinales. De l'autre côté, le AIC a choisi le modèle *VVI*, qui est aussi associé à une matrice de covariance diagonale. Or, on en déduit que la méthode préfère les modèles avec des covariances diagonales parce qu'ils ont moins de paramètre.

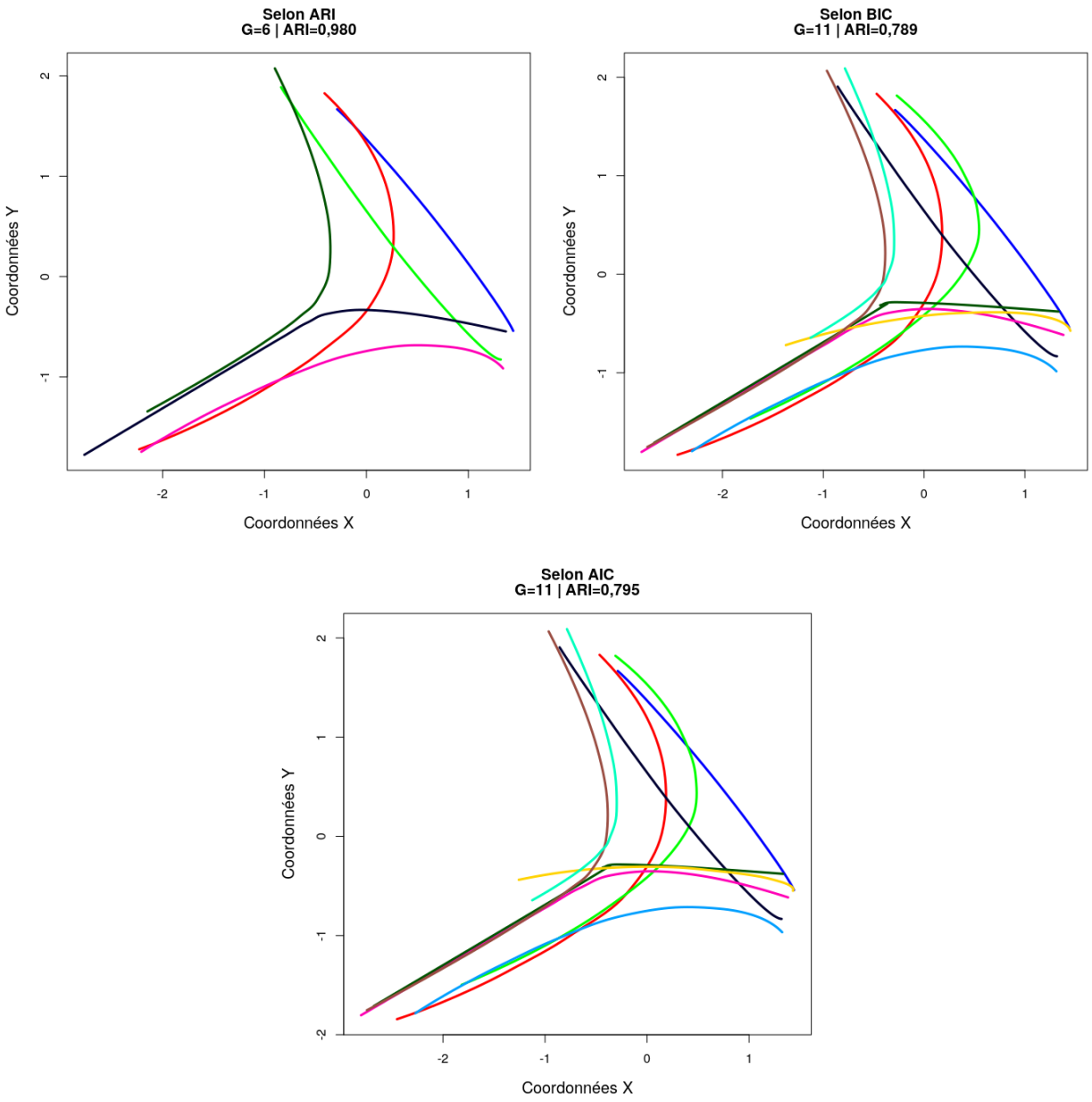

Fig. 5.4. Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *mclust*.

Ceci est un point important puisqu'on travail avec des données de grande dimension. La matrice de covariance est paramétrisée par $\Sigma_g = \lambda_g D_g$, alors les variances des positions aux différents temps de mesure varient. Finalement, les ARI associés aux BIC et AIC sont de 0,789 et 0,795 respectivement. Ce n'est pas l'idéal, mais cela reste toutefois satisfaisant. Les graphiques dans la figure 5.4 montrent que les partitions supplémentaires sont justement les virages différents, ce qui appuie notre hypothèse sur le nombre de G testé.

G	Model	ARI	BIC	AIC	G	Model	ARI	BIC	AIC
3	*EII*	0,477	-36939,19	-35659,73	**8**	*EII*	0,916	-2241,87	1170,02
	VII	0,465	-35238,99	-33953,95		*VII*	0,892	5386,04	8817,45
	EEI	0,495	-30579,85	-28879,48		*EEI*	0,916	-1245,53	2587,27
	EVI	0,510	-28685,58	-26143,39		*EVI*	0,913	-426,93	6352,25
	VEI	0,465	-29105,31	-27399,36		*VEI*	0,888	5087,70	8940,01
	VVI	0,494	-25781,39	-23233,62		*VVI*	0,897	4626,72	11425,41
4	*EII*	0,631	-24668,94	-22962,99	**9**	*EII*	0,866	-759,40	3078,98
	VII	0,675	-13346,68	-11632,38		*VII*	0,844	6533,41	10394,08
	EEI	0,633	-19268,42	-17141,56		*EEI*	0,866	174,17	4433,45
	EVI	0,631	-14687,65	-11298,06		*EVI*	0,866	809,93	8436,51
	VEI	0,653	-12164,61	-10029,39		*VEI*	0,839	6409,89	10691,47
	VVI	0,675	-9749,67	-6351,72		*VVI*	0,849	5539,57	13188,45
5	*EII*	0,748	-17826,02	-15693,59	**10**	*EII*	0,816	4310,27	8575,13
	VII	0,767	-4771,54	-2627,96		*VII*	0,789	**9056,32**	13346,27
	EEI	0,733	-12217,67	-9664,33		*EEI*	0,812	5431,84	10117,61
	EVI	0,817	-13944,82	-9707,83		*EVI*	0,813	3700,27	12174,25
	VEI	0,817	-5617,77	-3053,27		*VEI*	0,784	9023,39	13734,25
	VVI	0,817	-4075,78	172,35		*VVI*	0,795	6491,95	**14991,01**
6	*EII*	0,922	-7033,45	-4474,53	**11**	*EII*	0,814	4735,90	9427,25
	VII	0,922	1232,62	3805,48		*VII*	-	-	-
	EEI	0,922	-5507,13	-2527,30		*EEI*	0,783	3918,77	9031,03
	EVI	**0,980**	-5889,57	-805,19		*EVI*	-	-	-
	VEI	**0,980**	-893,65	2100,12		*VEI*	-	-	-
	VVI	**0,980**	583,40	5681,72		*VVI*	-	-	-
7	*EII*	0,894	-5621,17	-2635,77	**12**	*EII*	0,812	7310,84	12428,68
	VII	0,940	2270,69	5272,82		*VII*	-	-	-
	EEI	0,894	-3986,61	-580,29		*EEI*	0,808	7313,67	12852,42
	EVI	0,949	-5384,62	547,16		*EVI*	-	-	-
	VEI	0,940	2073,69	5496,73		*VEI*	-	-	-
	VVI	0,949	2470,24	8418,75		*VVI*	-	-	-

Tableau 5.2. Résultats du partitionnement avec l'algorithme *mclust*.

Passons maintenant aux modèles probabilistes fonctionnels. Le premier algorithme *fitfclust* présente des résultats moins satisfaisants que les derniers, avec un ARI optimal de 0,799. Celui-ci n'est pas associé au partitionnement prédéfini avec $G = 5$, ce qui est assez surprenant. Le BIC optimal provient du modèle où $G = 7$ avec un ARI 0,785; proche du meilleur cas. Ensuite, le meilleur AIC vient du cas où $G = 10$, un résultat semblable à ceux obtenus avec la méthode *mclust*, mais avec des partitions distinctes selon les graphiques de la figure 5.5. Les résultats sont présentés dans le tableau 5.3.

G	ARI	BIC	AIC
3	0,434	-123990,26	-58316,96
4	0,606	-119857,31	-53760,31
5	**0,799**	-118985,72	-52465,01
6	0,786	-119440,29	-52495,89
7	0,785	**-116915,25**	-49547,15
8	0,745	-117255,87	-49464,07
9	0,683	-117877,36	-49661,86
10	0,667	-117825,78	**-49186,59**
11	0,662	-118981,91	-49919,02
12	0,675	-119662,63	-50176,03

Tableau 5.3. Résultats du partitionnement avec l'algorithme *fitclust*.

La méthode *funmbclust* a été développée récemment, et elle possède encore des lacunes numériques. Malgré qu'on ait testé pour plusieurs dimensions de l'analyse en composantes principales k, peu ont convergé numériquement. On a tout de même des modèles avec des ARI presque parfaits, notamment le modèle où $k = 2$ et $G = 6$ avec $ARI = 0{,}9$. Les BIC et AIC optimaux coïncident avec $k = 1$ et $G = 7$ pour un ARI assez fort de 0,835. Cet algorithme est une nette amélioration par rapport à *fitfclust*. Les résultats des partitionnements sont dans le tableau 5.4.

k	G	ARI	BIC	AIC
1	6	0,156	-28135,90	-27486,41
	7	0,839	**8613,23**	**9265,51**
2	6	**0,900**	-131778,51	-130961,78
	9	0,835	-111254,87	-110421,41
	10	0,816	-175992,77	-175153,73

Tableau 5.4. Résultats du partitionnement avec l'algorithme *funmbclust*.

L'approche *funclust* est la moins performante de toutes. D'abord, l'algorithme a de la difficulté à partitionner correctement les données. À l'exception du cas $G = 3$, tous les résultats avaient une partition vide. Il semble que l'approche soit mal adaptée d'une part pour les données multivariées, mais aussi pour le partitionnement de grande taille. En effet, les analyses ont été effectuées dans [**14**] sur des données avec 2 ou 3 partitions. Ceci étant dit, le ARI maximal est de 0,623 avec $G = 6$. Même si les meilleurs BIC et AIC indiquent une meilleure adéquation lorsque $G = 3$, c'est aussi le seul qui n'a aucune partition vide. Les graphiques dans 5.7 sont peu concluants. Sur l'ensemble, le nombre de composantes

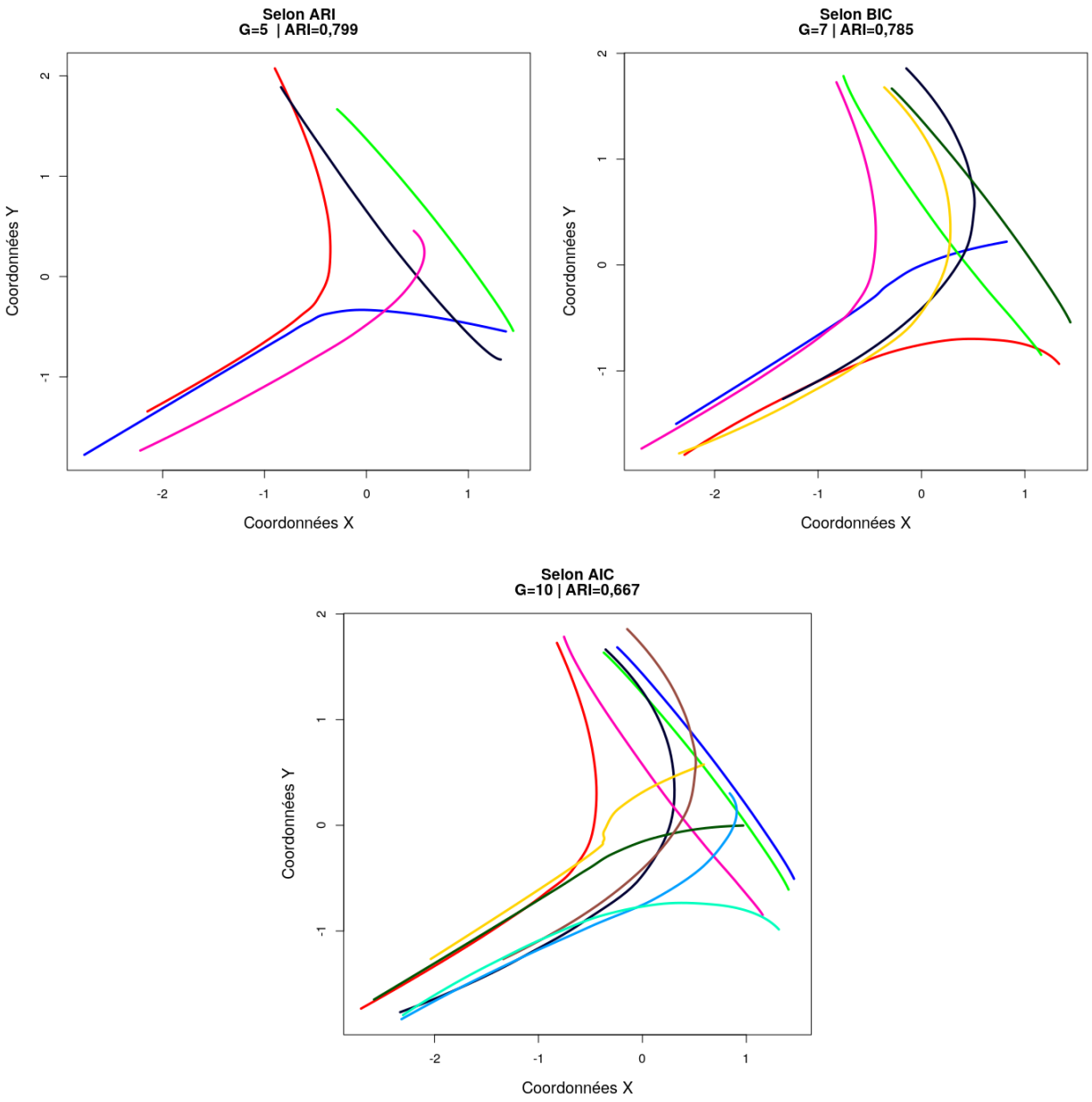

Fig. 5.5. Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *fitfclust*.

principales retenues est de 4, soit 2 par dimension. Les résultats sont dans le tableau 5.5.

Finalement, l'approche *funHDDC* démontre aussi des partitionnements satisfaisants. Tous les modèles n'ont pas convergé, et ce problème survient davantage lorsqu'on teste pour un G grand. Lorsque $G = 10{,}11{,}12$, seul les modèles $a_{kj}b_kH_kd_k$ et $a_kb_kH_kd_k$ convergent. De plus, les BIC et AIC ont tendance à croître lorsque G augmente. Malgré tout, un des modèles a réussi à obtenir un partitionnement exact (ARI = 1), avec le modèle $a_{kj}b_kH_kd_k$ et $G = 6$.

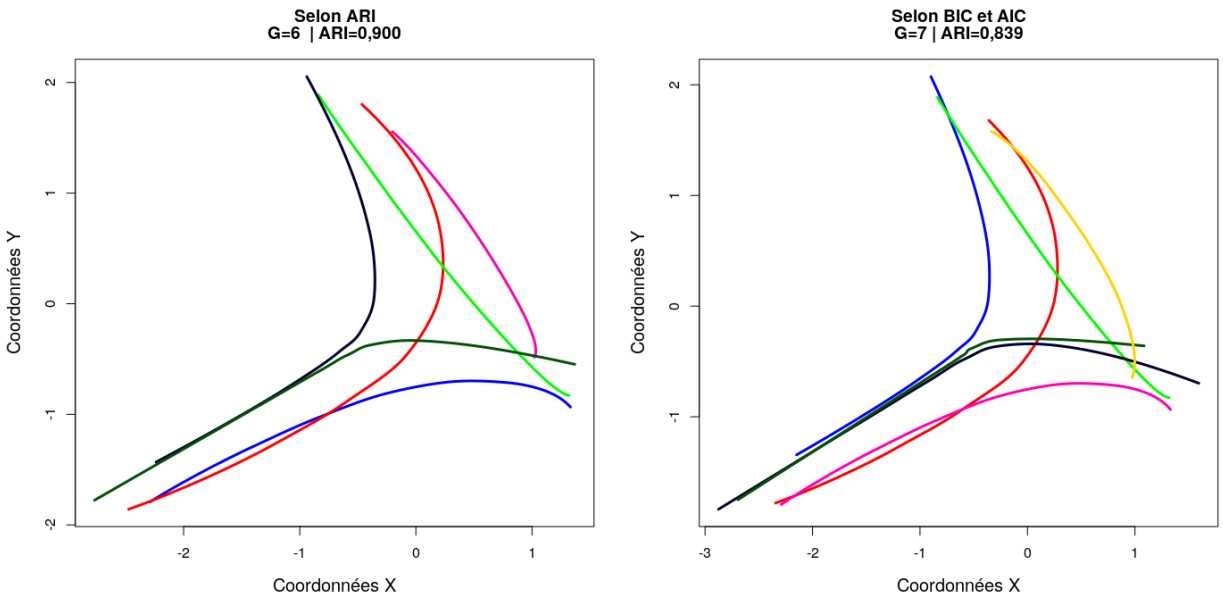

Fig. 5.6. Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *funmbclust*.

G	Dim.	ARI	BIC	AIC
3	4	0,046	**1108,09**	**1147,11**
4	4	0,394	860,75	913,72
5	4	0,150	919,23	983,34
6	4	0,498	810,05	890,89
7	4	0,531	786,13	880,91
8	4	0,602	769,51	878,22
9	4	**0,623**	828,53	951,18
10	4	0,529	842,72	979,31
11	4	0,592	707,56	858,08
12	4	0,496	682,27	846,73

Tableau 5.5. Résultats du partitionnement avec l'algorithme *funclust*. Dim.: La dimension du sous-espace propre fonctionnel par groupe.

Selon les critères BIC et AIC, le meilleur choix de paramètres est $G = 11$ avec le modèle $a_{kj}b_kH_kd_k$. Finalement, le nombre de composantes principales retenues est majoritairement 1, mais peut aller jusqu'à 3 comme on peut le remarquer dans le tableau 5.6. Le fait qu'une bonne majorité des dimensions des sous-espaces sont de 1 et 2 vient complémenter l'ACPF fait précédemment. Ces composantes représentaient autour de 95% de la variance et attribuaient d'importants poids à l'entrée et à la sortie des voitures à l'intersection. L'analyse complète se retrouve dans le tableau 5.6.

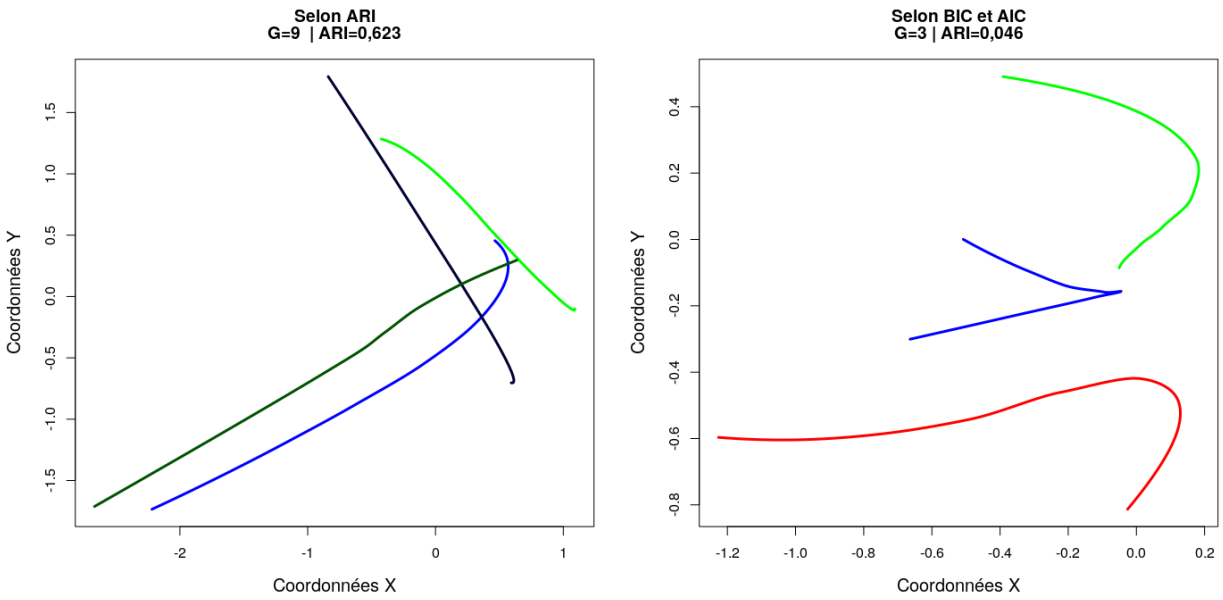

Fig. 5.7. Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *funclust*.

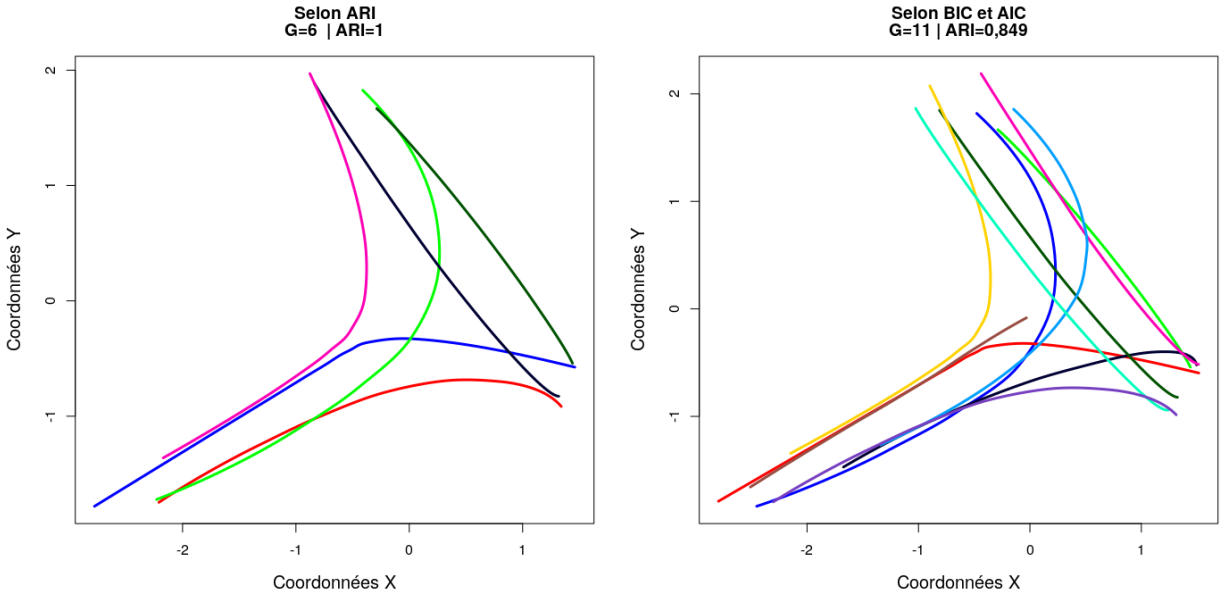

Fig. 5.8. Représentation graphique des trajectoires moyennes de chaque partition obtenue avec *funHDDC*.

5.4. Prédictions

On peut maintenant passer à la prédiction des directions des trajectoires. On a prédit les classes des trajectoires avec l'ensemble de prédictions à 859 observations selon les partitionnements obtenus par les méthodes. La qualité des prédictions sera donnée par

G	Model	Dim.	ARI	BIC	AIC	G	Model	Dim.	ARI	BIC	AIC
3	$a_{kj}b_kH_kd_k$	3,2,1	0,032	-5386313,35	-5385056,19	**8**	$a_{kj}b_kH_kd_k$	1,1,1,1,1,1,2,1	0,924	-1713570,60	-1711162,21
	$a_{kj}bH_kd_k$	1,2,2	0,451	-5596925,08	-5595807,29		$a_{kj}bH_kd_k$	2,1,1,1,1,1,1,1	0,891	-1683992,23	-1681603,35
	$a_kb_kH_kd_k$	-	-	-	-		$a_kb_kH_kd_k$	-	-	-	-
	$a_kbH_kd_k$	2,1,1	0,519	-4576682,64	-4575704,23		$a_kbH_kd_k$	2,1,1,2,1,1,1,3	0,842	-1644736,76	-1641952,05
	$ab_kH_kd_k$	1,2,3	0,032	-5386469,97	-5385226,75		$ab_kH_kd_k$	2,1,1,2,1,1,2,2	0,747	-2507161,16	-2504373,66
	abH_kd_k	2,1,2	0,451	-5596975,18	-5595868,54		abH_kd_k	1,1,1,3,2,1,1,2	0,842	-1644748,63	-1641983,44
4	$a_{kj}b_kH_kd_k$	1,1,1,2	0,675	-3936362,13	-3935091,04	**9**	$a_{kj}b_kH_kd_k$	1,1,1,1,1,1,1,1,1	0,881	-1683633,77	-1681077,64
	$a_{kj}bH_kd_k$	1,2,2,1	0,631	-3816853,18	-3815453,86		$a_{kj}bH_kd_k$	1,1,1,1,1,2,2,2,1	0,866	-1787663,98	-1784720,39
	$a_kb_kH_kd_k$	1,1,2,2	0,649	-4065939,12	-4064537,01		$a_kb_kH_kd_k$	-	-	-	-
	$a_kbH_kd_k$	1,1,2,2	0,631	-3816853,58	-3815459,84		$a_kbH_kd_k$	1,2,1,2,1,1,1,1,2	0,863	-1879140,12	-1876204,89
	$ab_kH_kd_k$	1,1,2,2	0,649	-4065945,74	-4064551,99		$ab_kH_kd_k$	-	-	-	-
	abH_kd_k	1,1,1,1	0,659	-4671567,19	-4670449,40		abH_kd_k	2,1,1,2,2,2,1,1,2	0,846	-2151842,84	-2148662,31
5	$a_{kj}b_kH_kd_k$	-	-	-	-	**10**	$a_{kj}b_kH_kd_k$	-	-	-	-
	$a_{kj}bH_kd_k$	1,1,1,1,2	0,764	-2486571,29	-2485027,02		$a_{kj}bH_kd_k$	1,3,1,2,1,1,2,1,1,1	0,847	-1906470,42	-1903111,49
	$a_kb_kH_kd_k$	1,1,2,1,1	0,817	-2336016,51	-2334463,87		$a_kb_kH_kd_k$	-	-	-	-
	$a_kbH_kd_k$	1,1,1,2,1	0,750	-4430522,53	-4428981,05		$a_kbH_kd_k$	1,3,1,1,1,1,1,2,1,1	0,843	-1644577,50	-1641363,53
	$ab_kH_kd_k$	1,1,1,1,2	0,817	-2336049,78	-2334508,29		$ab_kH_kd_k$	-	-	-	-
	abH_kd_k	1,1,1,1,1	0,530	-1960655,86	-1959259,33		abH_kd_k	-	-	-	-
6	$a_{kj}b_kH_kd_k$	2,1,1,1,1,1	**1,000**	-1730066,71	-1728226,97	**11**	$a_{kj}b_kH_kd_k$	-	-	-	-
	$a_{kj}bH_kd_k$	2,1,2,2,2,1	0,726	-4543434,88	-4541199,31		$a_{kj}bH_kd_k$	1,1,1,1,1,1,1,1,1,2,1	0,849	**-1436530,28**	**-1433296,79**
	$a_kb_kH_kd_k$	-	-	-	-		$a_kb_kH_kd_k$	-	-	-	-
	$a_kbH_kd_k$	1,1,1,2,1,1	0,940	-1817717,82	-1815894,80		$a_kbH_kd_k$	1,3,1,1,1,1,1,2,1,1,1	0,842	-1778991,40	-1775495,88
	$ab_kH_kd_k$	-	-	-	-		$ab_kH_kd_k$	-	-	-	-
	abH_kd_k	2,1,1,1,1,1	0,940	-1817719,51	-1815910,43		abH_kd_k	-	-	-	-
7	$a_{kj}b_kH_kd_k$	-	-	-	-	**12**	$a_{kj}b_kH_kd_k$	-	-	-	-
	$a_{kj}bH_kd_k$	1,2,1,1,1,1,1	0,945	-1899544,70	-1897437,35		$a_{kj}bH_kd_k$	1,2,1,1,2,3,2,1,2,1,2,1	0,731	-1857367,70	-1853035,94
	$a_kb_kH_kd_k$	1,1,1,1,2,1,1	0,966	-1717887,11	-1715765,83		$a_kb_kH_kd_k$	-	-	-	-
	$a_kbH_kd_k$	1,1,1,1,2,1,1	0,945	-1899543,30	-1897438,74		$a_kbH_kd_k$	1,2,1,1,1,1,1,1,1,1,1,3	0,769	-1499359,13	-1495582,07
	$ab_kH_kd_k$	2,1,1,1,1,1,2	0,820	-2235764,63	-2233526,28		$ab_kH_kd_k$	-	-	-	-
	abH_kd_k	1,2,1,2,1,2,1	0,723	-4055329,71	-4052974,28		abH_kd_k	-	-	-	-

Tableau 5.6. Résultats du partitionnement avec l'algorithme *funHDDC*. Dim.: La dimension du sous-espace propre fonctionnel par groupe.

les ARI qui comparent la prédiction faite par chacune des méthodes et la classification manuelle lors de la préparation des données. La méthode *distclust* a calculé les trajectoires moyennes de chaque partition. Donc on associera à chaque nouvelle trajectoire la partition de la moyenne la plus proche. Pour ce qui est des méthodes fonctionnelles, on a calculé des ensembles de paramètres estimés pour chacune et elles permettront de calculer les probabilités qu'une courbe soit dans une partition, comme expliqué dans le chapitre 4.

Avec les partitionnements optimaux de chaque modèle et selon chaque critère de sélection et le ARI, les prédictions sont satisfaisantes. On rappelle qu'il n'est pas possible de faire de la prédiction avec le modèle *funclust*, et que seul le critère ARI est disponible avec *distclust*. On discutera des méthodes avec les résultats trouvés dans le tableau 5.7.

L'approche par les distances *distclust* est très performante, avec un ARI de 0,968. La méthode *mclust* reste consistante dans ses prédictions. Même si le nombre de groupes choisi était de 10 selon le BIC et le AIC, la prédiction atteint de tout même des scores de 0,890

et de 0,932 respectivement. *Fitfclust* réussit à obtenir un ARI de 0,940 avec son modèle choisi selon le BIC, et un de 0,836 avec celui choisi selon le AIC. La même chose peut être dite pour *funmbclust*. Enfin, la méthode funHDDC, qui avait obtenu une partition parfaite lors du partitionnement, obtient une prédiction avec des ARI de 0,961 et de 0,706 selon les critères d'information qui sont les prédictions les plus faibles.

	Selon ARI		**Selon BIC**		**Selon AIC**	
Méthode	**ARI**	**G**	**ARI**	**G**	**ARI**	**G**
distclust	0,968	6	-	-	-	-
mclust	0,986	6	0,890	10	0,932	10
fitfclust	0,955	5	0,940	7	0,836	10
funmbclust	0,941	6	0,898	7	0,898	7
funclust	-	-	-	-	-	-
funHDDC	0,961	6	0,706	11	0,706	11

Tableau 5.7. Résultats de la prédiction selon le résultat avec le meilleur ARI, BIC et AIC pour chaque méthode.

Chapitre 6

Conclusion

Le but de cet ouvrage est d'évaluer la capacité des algorithmes de partitionnement à classifier les trajectoires dans le domaine du transport routier. L'idée était de traiter des trajectoires de voitures à une intersection à trois branches comme des données fonctionnelles longitudinales. Pour cela, on a exploré des outils importants de l'analyse fonctionnelle et du partitionnement.

Avec un total de six méthodes de partitionnement, une de distance, une de mélange de densité et quatre fonctionnelles, on a pu effectuer cette tâche. On a conclu que la majorité d'entre elles sont capables de classifier les courbes, et même de déterminer un autre partitionnement que celui qu'on avait prédéterminé. Avec les algorithmes entraînés, la prédiction sur un nouvel ensemble de données était satisfaisante. Même si une des méthodes probabilistes avait déterminé une classification autre que celle établie, elles ont démontré être conséquentes dans leurs règles.

En somme, les résultats du partitionnement des trajectoires à l'aide de méthodes fonctionnelles probabilistes sont semblables à ceux obtenus avec la méthode par distance. Cependant, même si l'algorithme des K-moyennes et ses dérivés, comme l'algorithme *distclust*, sont efficaces, ils ne procurent aucune modélisation statistique des données. Pour mesurer la qualité du partitionnement, on doit comparer celui estimé par l'algorithme avec un préétabli sur les données, ce qui amène une subjectivité dans nos analyses et aussi, au fait que ces méthodes ne peuvent pas être appliquées pour déterminer les nombres de classes dans les données. On préfère l'utilisation des modèles fonctionnelles pour ces raisons. D'abord, parce qu'une modélisation des données est disponible, ce qui permet l'étude de la structure fonctionnelle des trajectoires. D'une autre part, on peut utiliser des critères de sélection objectives comme le AIC et le BIC, et ainsi faire une sélection du nombre de partitions sans avoir à comparer les résultats avec une partition préétablie. En d'autres mots, on peut laisser les données parler d'eux-mêmes et découvrir les partitions au travers

des différentes modélisations.

Par la suite, les résultats étaient aussi satisfaisants pour la prédiction des directions des trajectoires. Les modèles fonctionnels sont restés cohérents dans leurs partitions estimées. La seule exception est la méthode *funHDDC* : il pourrait être pertinent d'étudier cette approche de nouveau pour voir si un ajustement est possible.

Finalement, on s'est concentré sur une partie de la grande problématique de la classification des trajectoires routières. Il serait intéressant d'évaluer la capacité des algorithmes à classifier les trajectoires avec de nombreux types d'usagers mélangés, comme des cyclistes et des piétons. La difficulté sera de traiter toutes ces trajectoires, qui sont de natures distinctes, dans un même cadre statistique. Pour cela, il faudra revisiter et peut être adapter certaines méthodes discutées dans ce mémoire.

Références bibliographiques

[1] Adjobo Folly Dzigbodi Adjogou : Analyse statistique de données fonctionnelles à structures complexes. 2018.

[2] Hirotugu Akaike : A new look at the statistical model identification. *In Selected Papers of Hirotugu Akaike*, pages 215–222. Springer, 1974.

[3] Charles Bouveyron et Julien Jacques : Model-based clustering of time series in group-specific functional subspaces. *Advances in Data Analysis and Classification*, 5(4):281–300, 2011.

[4] George EP Box et Norman R Draper : *Empirical model-building and response surfaces.* John Wiley & Sons, 1987.

[5] Raymond B Cattell : The scree test for the number of factors. *Multivariate behavioral research*, 1(2):245–276, 1966.

[6] Gilles Celeux et Gérard Govaert : Gaussian parsimonious clustering models. *Pattern recognition*, 28(5):781–793, 1995.

[7] William S Cleveland : Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[8] Aurore Delaigle, Peter Hall *et al.* : Defining probability density for a distribution of random functions. *The Annals of Statistics*, 38(2):1171–1193, 2010.

[9] Arthur P Dempster, Nan M Laird et Donald B Rubin : Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[10] Chris Fraley et Adrian E Raftery : Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.

[11] Jerome Friedman, Trevor Hastie et Robert Tibshirani : *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[12] Lawrence Hubert et Phipps Arabie : Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[13] Julien Jacques et Cristian Preda : Functional data clustering: a survey. *Advances in Data Analysis and Classification*, 8(3):231–255, 2014.

[14] Julien Jacques et Cristian Preda : Model-based clustering for multivariate functional data. *Computational Statistics & Data Analysis*, 71:92–106, 2014.

[15] Gareth M James et Catherine A Sugar : Clustering for sparsely sampled functional data. *Journal of the American Statistical Association*, 98(462):397–408, 2003.

[16] Johan Ludwig William Valdemar Jensen *et al.* : Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30:175–193, 1906.

[17] Joseph B Kruskal : Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[18] Brian G Leroux : Consistent estimation of a mixing distribution. *The Annals of Statistics*, pages 1350–1360, 1992.

[19] Roderick JA Little et Donald B Rubin : *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[20] Stuart Lloyd : Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[21] Sharon L Lohr : *Sampling: Design and Analysis: Design and Analysis*. Chapman and Hall/CRC, 2019.

[22] Jie Peng, Hans-Georg Müller *et al.* : Distance-based clustering of sparsely observed stochastic processes, with applications to online auctions. *The Annals of Applied Statistics*, 2(3):1056–1077, 2008.

[23] James O Ramsay : Functional data analysis. *Encyclopedia of Statistical Sciences*, 4, 2004.

[24] James O Ramsay et Bernard W Silverman : *Applied functional data analysis: methods and case studies*. Springer, 2007.

[25] William M Rand : Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[26] Gideon Schwarz *et al.* : Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[27] Robert Roth Stoll : *Set theory and logic*. Courier Corporation, 1979.

[28] Yoshio Takane, Forrest W Young et Jan De Leeuw : Nonmetric individual differences multidimensional scaling: An alternating least squares method with optimal scaling features. *Psychometrika*, 42(1):7–67, 1977.

[29] Read D Tuddenham : Physical growth of california boys and girls from birth to eighteen years. *University of California publications in child development*, 1:183–364, 1954.

[30] Fang Yao, Hans-Georg Müller et Jane-Ling Wang : Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100(470):577–590, 2005.

[31] Christina Yassouridis, Dominik Ernst et Friedrich Leisch : Generalization, combination and extension of functional clustering algorithms: The r-package funcy. *J. Stat. Softw*, 2016.

Annexe A

Annexe

A.1. Identité de Parseval

Théorème A.1.1. *Soit H un espace de Hilbert avec une base orthonormée formée par la suite $\{v_k\}_{k=1}^{\infty}$. On suppose que chaque élément $y \in H$ peut s'écrire par*

$$y = \sum_{k=1}^{\infty} \langle y,v_k \rangle \, v_k,$$

avec $\langle \cdot, \cdot \rangle$ un produit scalaire défini sur l'espace H. Alors

$$\sum_{k=1}^{\infty} | \langle y,v_k \rangle |^2 = \|y\|^2, \quad \forall y \in H.$$

Démonstration. Puisque les éléments de la suite $\{v_k\}_{k=1}^{\infty}$ forment une base orthonormée de l'espace H, alors

$$\langle v_k,v_l \rangle = \begin{cases} 1 & \text{si } k = l \\ 0 & \text{si } k \neq l \end{cases}$$

Avec l'hypothèse faite sur y, on a que

$$\begin{aligned} \|y\|^2 &= \left\langle \sum_{k=1}^{\infty} \langle y, v_k \rangle \, v_k, \sum_{k=1}^{\infty} \langle y, v_k \rangle \, v_k \right\rangle \\ &= \sum_{k=1}^{\infty} \sum_{l=1}^{\infty} \langle y, v_k \rangle \, \langle y, v_l \rangle \, \langle v_k, v_l \rangle \\ &= \sum_{k=1}^{\infty} | \langle y,v_k \rangle |^2. \end{aligned}$$

□